Long-term Robot Mapping in Dynamic Environments

by

Aisha Naima Walcott

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2011

©Massachusetts Institute of Technology 2011. All rights reserved.

Author .....
Department of Electrical Engineering and Computer Science
May 26, 2011

Certified by .....
John J. Leonard
Professor of Mechanical and Ocean Engineering
Thesis Supervisor

Accepted by .....
Professor Leslie A. Kolodziejski
Chairman, Department Committee on Graduate Students

Long-term Robot Mapping in Dynamic Environments

by

Aisha Naima Walcott

Submitted to the Department of Electrical Engineering and Computer Science
on May 26, 2011, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Computer Science

Abstract

One of the central goals in mobile robotics is to develop a mobile robot that can construct a map of an initially unknown dynamic environment. This is often referred to as the Simultaneous Localization and Mapping (SLAM) problem. A number of approaches to the SLAM problem have been successfully developed and applied, particularly to a mobile robot constructing a map of a 2D static indoor environment. While these methods work well for static environments, they are not robust to dynamic environments which are complex and composed of numerous objects that move at wide-varying time-scales, such as people or office furniture.

The problem of maintaining a map of a dynamic environment is important for both real-world applications and for the advancement of robotics. A mobile robot executing extended missions, such as autonomously collecting data underwater for months or years, must be able to reliably know where it is, update its map as the environment changes, and recover from mistakes. From a fundamental perspective, this work is important in order to understand and determine the problems that occur with existing mapping techniques for persistent long-term operation.

The primary contribution of the thesis is Dynamic Pose Graph SLAM (DPG-SLAM), a novel algorithm that addresses two core challenges of the long-term mapping problem. The first challenge is to ensure that the robot is able to remain localized in a changing environment over great lengths of time. The second challenge is to be able to maintain an up-to-date map over time in a computationally efficient manner. DPG-SLAM directly addresses both of these issues to enable long-term mobile robot navigation and map maintenance in changing environments. Using Kaess and Dellaert's incremental Smoothing and Mapping (iSAM) as the underlying SLAM state estimation engine, the dynamic pose graph evolves over time as the robot explores new areas and revisits previously mapped areas.

The algorithm is demonstrated on two real-world dynamic indoor laser data sets, demonstrating the ability to maintain an efficient, up-to-date map despite long-term environmental changes. Future research issues, such as the integration of adaptive exploration with dynamic map maintenance, are identified.

Thesis Supervisor: John J. Leonard
Title: Professor of Mechanical and Ocean Engineering

Acknowledgments

I want to first thank my family, my mother, my father, my three sisters, my brother, and my adorable sweet niece. They have inspired me and been there in both good times and not so good times. I especially want to thank my husband Reginald for all his love and support, and for our fun technical conversations between nano-electromechanical systems and mobile robots.

I want to thank my advisor, Professor John Leonard, for his unwavering support and patience through my tenure at MIT, and spontaneous crazy robot ideas one of which inspired the work in this thesis! I'd like to thank my committee members, Professor Leslie Kaelbling and Professor Tomas Lozano-Perez for their sage advice and technical feedback. I would like to thank Dr. Michael Kaess for providing us with his iSAM code. I would also like to thank Dr. Peter Biber for providing us with the University of Tubingen data set.

To my research group members and lab colleagues, Hordur, Michael, Maurice, Georgios, Jacques, Rob, Sam, Andrew, Beenie, Mike B, Muyiwa, Finale, Javier, and many others. Also, at CSAIL I'd like to thank TIG (The Infrastructure Group) who have helped me in crazy computer situations and have supported service projects that I have worked on to improve developing communities through education and technology, namely the Laare, Kenya project.

At MIT there were a number of great and supportive people, groups, and offices. In particular I want to extend a heartfelt thank you to my colleagues and friends Dr. Robbin Chapman, Dr. Ishara Mills-Henry, Dr. Danielle Hinton, Dr. James McLurkin, Dr. Lincoln Chandler, Dr. Muyiwa Olubuyide, Dr. Shaundra Daily, Dr. Amon Millner, Dr. Nicole Love, Dr. Sekou Remy, Mrs. Legena Henry, Mrs. Vernella Vickerman, Dr. Cortina McCurry, Mr. Eric Brittain, and Mr. Melvin Henry.

My thanks to Professors Seth Teller, Wesley Harris, Latanya Sweeney, and Deans Blanche Staton, Ike Colbert, Mrs. Marilyn Peirce, and the ODGE and EECS offices. Thanks to all the members of ACME, BGSA, the Fab Lab, MIT-AITI, both past and present. And, of course, to the undergrads of New House 2.

This thesis is dedicated to my grandma Rivera, grannie Walcott, auntie Walcott, and my late sister, Rashida Walcott. Each of these amazing women have shown me undying love and support over my lifetime. They started this PhD journey with me and I will miss their still being present at the finish.

This research has been supported by the Lucent CRFP, the MIT GSO-Lemelson Fellowship, the MIT Graduate Students Office Fellowship, the Stanley N. (1942) and Thelma L. Golembe Fellowship Fund. Also, this thesis is partially the result of research sponsored by the MIT Sea Grant College Program, under NOAA Grant Number NA06OAR4170019, MIT SG Project Number 2007-R/RCM-20, and by the Office of Naval Research under research grants N00014-05-10244 and N00014-07-11102.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

One of the central goals in mobile robotics is placing a robot in an unknown, dynamic environment with the mission to autonomously construct a map of the environment. Achieving this goal requires the robot to localize while constructing a map of the environment, and is referred to as the Simultaneous Localization and Mapping (SLAM) problem. A number of successful approaches have been developed particularly for static environments. However, environments are generally not static and are complex with objects moving at wide-varying time-scales.

This thesis presents a novel method called the Dynamic Pose Graph SLAM (DPG-SLAM) that directly addresses the problem of long-term mobile robot navigation and map maintenance in dynamic environments. The robot is assumed to operate in such environments for extended periods, eg. weeks, months, or even years.

(a) (b) (c)

Figure 1-1: Example of different real-world environments for potential long-term robot mapping. (a) shows one of the two robots, Spirit and Opportunity, that have been operating on Mars for several years. Figure (b) shows a robot patrolling a public space. Figure (c) shows an image of the aftermath of the Japan earthquake.

Environments are constantly changing, both in the short term and the long term. The dynamics of a complex environment directly affect what the robot senses and ultimately incorporates into its map. The focus of this thesis is on environments that change slowly over time, and often abruptly and seemingly at random, such as furniture moving. Additionally, robots are being used on longer term missions, or in scenarios where they have a persistent presence in our every-day lives (see Figure 1-1). For example, consider a mobile robot navigating in an airport and maintaining a model of the dynamics of the environment. The robot must be able to adapt to changes in the environment such as in an airport terminal that is undergoing remodeling, and at the same time, discern when a change oc-

curs such as an unexpected package placed in a prohibited area. Our DPG-SLAM method provides a robot with the ability to adapt to long-term changes as well as detect new objects added to the environment.

The long-term SLAM problem presents a number of challenges. For example, vast amounts of data are collected, objects move according to different time scales, and coping with failures inherent in long-term autonomous operation. To confront some of these issues a number of methods to SLAM in populated and dynamic environments have been proposed. Many of these methods approximate a static map, and use techniques to filter out transient obstacles, such as people walking [3, 32, 54, 66, 75].

1.1 Long-term Mapping Overview

Traditional SLAM and methods for SLAM in dynamic environments are not robust to environment changes that occur at wide-varying time scales. A robot covering a large environment will navigate and revisit parts of the environment at different points in time. One of the main difficulties the robot faces is dealing with objects that move infrequently and do not follow a continuous trajectory, such as sofas, chairs, and desks. To maintain an accurate and up-to-date map, the robot must be able to detect environment changes and repair the map. Revisiting a room where furniture has been moved, for example, results in different measurements of the same room. Both, people walking and furniture being moved, are examples of what makes an environment dynamic. However, the two have significantly different time-scales. People walking are in continuous motion while within the robot's field-of-view. In contrast, a sofa is generally stationary while within the robot's field-of view. The sofa can be moved or removed outside of the robot's FOV and after the sofa has be included in the map. The only way for the robot to be able to detect this type of change is to revisit the location where the sofa initially was located, take new measurements and repair the map.

While different approaches have been proposed to represent long-term dynamic maps [9, 10, 52, 80], the long-term mapping in dynamic environments remains an open research problem in robotics.

This thesis presents a novel solution to the problem of long-term mapping in dynamic environments called DPG-SLAM (Dynamic Pose Graph SLAM). A low-dynamic environment is one that consists of static and low-dynamic objects. The term low-dynamic objects was introduced in [52] to describe objects that appear stationary within the robot's field-of-view, and move while outside the robot's field-of-view. To enable a mobile robot to navigate and maintain an accurate and up-to-date map while operating in a low-dynamic environment, the DPG-SLAM method focuses on achieving four critical goals listed below.

Long-term Mapping Goals:

1. *Continuously incorporate new information.* As the robot navigates, information about the environment obtained by the laser range finder should be continuously incorporated into the DPG model. This includes computing constraints, updating the robot's trajectory, and maintaining an up-to-date map.

2. *Representation of the environment should include the history of the map as changes occur with the passage of time.* The representation of the environment should store information on where changes occurred and where changes did not occur. By maintaining

the environment changes, the model can be used to represent the temporal dynamics of the environment. That is, it would be possible to determine how often changes occur in various parts of the environment based on the laser data collected by the robot. In addition to keeping track of the changes, it is also important to maintain the areas of the environment that do not change. These areas can be considered as "more static" than other areas. By incorporating this information into the map of the dynamic envisagement, such applications like path planning can exploit the model by planning paths through parts of the environment that are more static. In these areas it is more likely that the robot can accurately localize.

3. *Detect changes and update the map online.* To maintain an accurate and up-to-date map the robot should detect changes relative to its current pose and repair its current representation of the environment online. This process should be done incrementally at the pose level; i.e., as the robot navigates and adds poses to the DPG.

4. *Address the problem of tractability for growing DPG.* At the heart of the Dynamic Pose Graph model is a pose graph [28] representation. A pose graph consists of nodes and edges that make up the robot's trajectory (see Chapter 3 for details). The size of the pose graph grows with time as the robot navigates around its environment. The continuous addition of nodes and constraints to the DPG directly affects the computational tractability when pose-graph optimization is applied. Therefore, a mobile robot navigating for weeks, months, or even years, will generate an ever increasing DPG. To address the problem of tractability, a framework for reducing the DPG size including nodes, edges, and ranges should be a part of DPG-SLAM.

1.2 Summary of Contributions

The primary contributions of this thesis are threefold. First, we develop the Dynamic Pose Graph model, an extension to pose graphs that has additional information stored at the nodes in order to cope with the dynamic environments. Second, we propose a unified environment representation called the *active* and *dynamic* maps. The third contribution is the DPG-SLAM method which has two parts. The first part is maintaining an up-to-date representation in the face of environment changes, referred to as DPG-SLAM-NR. The second part is addressing the tractability problem by reducing the size of the DPG model. Figure 1-2 presents an example of our work as compared to traditional pose graph SLAM methods applied to a dynamic environment. We demonstrate our work on a B21 mobile robot (shown in Figure 1-3.) equipped with a laser range finder in the plane providing 180^o coverage.

1.3 Thesis Outline

The layout of this thesis is as follows. Chapter 2 describes related work in long-term robot mapping and SLAM in dynamic environments. Chapter 3 presents the foundation for this work, namely pose graph SLAM. Chapter 4 details our formulation of the long-term mapping problem. Next, Chapter 5 describes the DPG-SLAM method, its core algorithms, and concludes with an example of DPG-SLAM applied to the CSAIL Reading Room Data set. Chapter 6 presents the analysis and results illustrating the efficacy of our DPG-SLAM

methods applied to the CSAIL Reading Room data set and the University of Tubingen data set [9]. Finally, Chapter 7 concludes with a summary of the work presented in this thesis, limitations of our methods, as well as potential future research.

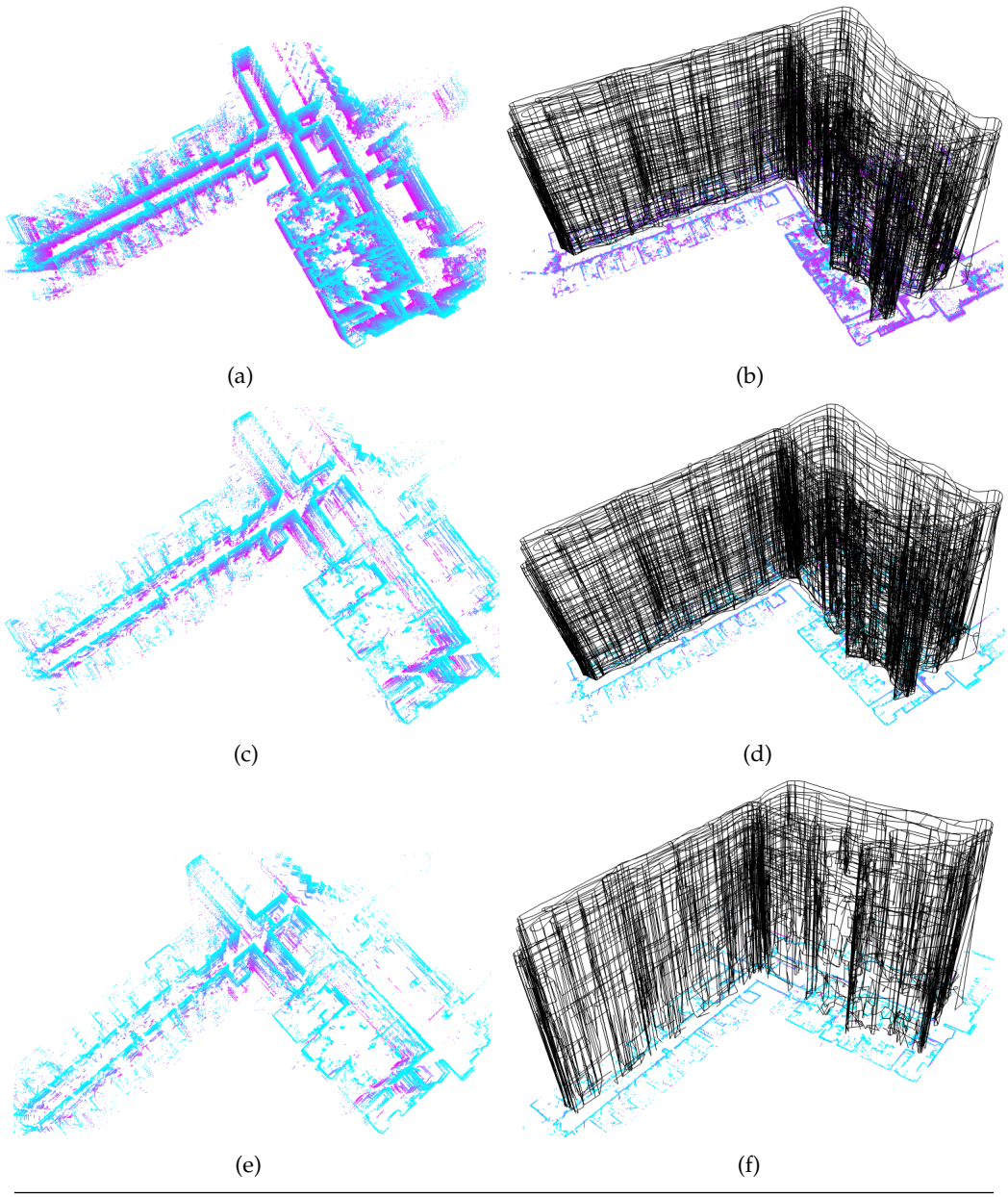


Figure 1-2: Images of pose graph SLAM, DPG-SLAM-NR, and DPG-SLAM on 60 passes through the University of Tubingen, Robot Lab. Figures (a) and (b) are the results of applying pose graph SLAM. Figures (c) and (d) are the result of applying DPG-SLAM-NR (our method for maintaining an up-to-date map). Figures (e) and (f) are the results of both maintaining an up-to-date map and reducing the size of the DPG (ie. DPG-SLAM).

Figure 1-3: B21 robot used in our experiments.

Chapter 2

Related Work

This thesis presents Dynamic Pose Graph SLAM (DPG-SLAM) a method for maintaining an up-to-date map in a slowly changing environment. There has been a lot of attention on the problem of SLAM in dynamic environments particularly where the robot operates for short periods, or covers the environment and generates a map. Typical methods where a robot performs SLAM in a dynamic environment for short periods, filter out dynamic objects, detect and classify static objects versus dynamic objects, or track moving objects [3,32,54,66,75].

More recently, the problem of continuous, long-term SLAM, also known as the life-long SLAM problem, has received much attention. A critical part of the life-long SLAM problem is the ability to construct and maintain an up-to-date map while operating in dynamic environments. The robot is repeatedly covers the environment while objects move at wide-varying time scales. More specifically, [52] describes dynamic environments as those composed of static *low-dynamic* and *high-dynamic* objects. High-dynamic objects are those that appear in motion while within the robot's field-of-view (FOV), such as transient objects like people and vehicles [52]. Low-dynamic objects are those that are stationary within the robot's FOV and move at random outside the robot's FOV, such as sofas and desks. While there have been some recent successful approaches have been proposed [9,10,52,80] there remain a number of open challenged.

This chapter summarizes related work from traditional SLAM in static environments towards life-long SLAM in dynamic environments. At the heart of the life-long SLAM problem is the long-term mapping problem. We discuss recent work on long-term SLAM in dynamic environments and the various models for representing the environment.

2.1 Summary of SLAM in Static and Dynamic Environments

In traditional SLAM the robot is placed in an initially unknown and static environment and is tasked with constructing a map of the environment. For example, Figure 2-1(a) shows a room comprised of static objects i.e., walls. To construct a map the robot makes one pass through the room to cover the space. To cover the space autonomously the robot must follow an exploration strategy as well as localize and construct a map. As the robot navigates, errors in its odometry measurements accrue. At the same time sensor measurements are corrupted by noise. Consequently, a core part of the general SLAM problem is the ability for the robot to cope with uncertainty in its motion and sensor measurements.

To construct a map of a static environments numerous approaches to the traditional

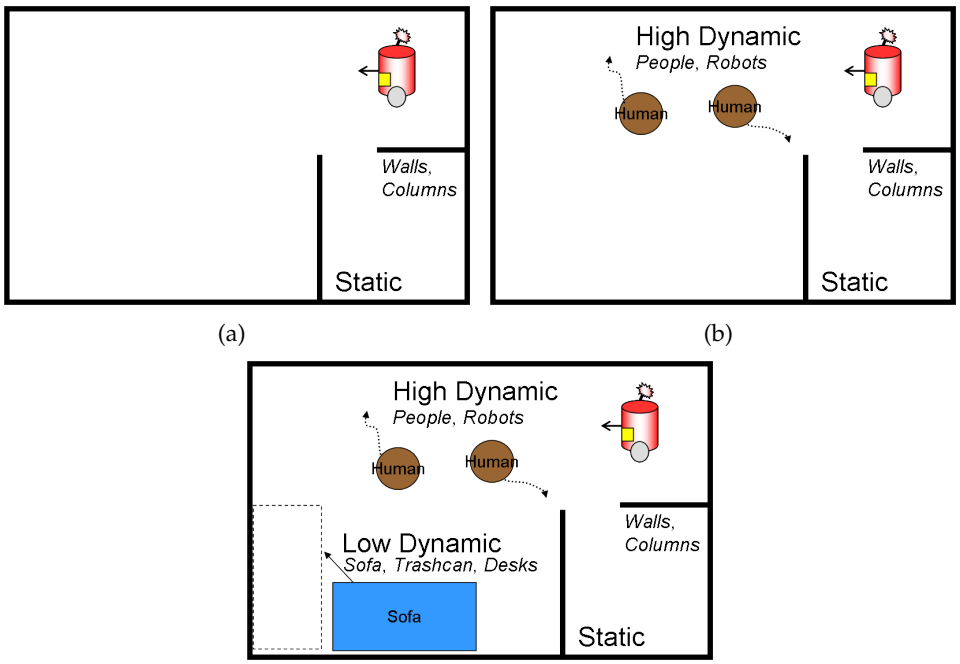


Figure 2-1: Example of static, high-dynamic, and low-dynamic entities (objects) in an environment.

SLAM problem have been proposed. The two premier approaches apply smoothing, which using both the past and the future measurements, and filtering, which used the past measurements. Smoothing methods compute the full posterior of the robot poses along the trajectory and include [16, 20, 28, 30, 36, 41, 53, 58]. Filtering methods maintain the current position of the robot and include the Extended Kalman Filter (EKF), and the Extended Information Filter (EIF), and Particle Filters (PFs) [12, 27, 65, 72].

To construct a map autonomously the robot must be able to explore and cover the environment. A number of exploration strategies have been proposed and tested in indoor environments both with single and multiple robots [23, 24, 43, 59, 71]. Exploration strategies alone are not sufficient in more realistic scenarios in which objects, such as people or furniture, are moving at various times scales.

SLAM in high-dynamic environments includes environments with objects that are moving within the FOV of the robot (see Figure 2-1(b)). Many of the approaches to the SLAM problem in high-dynamic environments use methods to filter-out or track moving objects [15,31,46,54,73,74]. Examples include robots operating in populated environments such as museum tour guide robots [64, 68] or robots assisting the elderly in nursing homes [60, 61]. Also, recent success with the DARPA grand challenges demonstrated autonomous vehicles operating safely in high-dynamic environments [1].

2.2 Long-term Mapping in Dynamic Environments

A core part of the life-long SLAM problem is the robot's ability to maintain an up-to-date map while operating in an environment for an extended period, e.g. weeks, months, or years. In general, the environment would consist of static, low-dynamic, and high-

dynamic objects. Recently there has been some work focusing on the two important areas of long-term mapping environment representation and long-term robot operation.

2.2.1 Representation

To represent a dynamic environment both spatial and temporal data can be included in the map. That is, the location of an object (represented by the sensor measurements) and the time the robot detected the object (or set of measurements referring to an object). Three common approaches to representing a dynamic environment are grid-based, map aging, and multiple maps.

Grid based methods extend the original occupancy grid, first proposed by Moravec [55] and Elfes [18], which represent free, occupied, and unknown space with a discrete grid of cells. The cells are updated probabilistically as new sensor data is acquired and as the robot moves through the environment. A variety of techniques have been proposed for updating grid cells based on new sensor data, such as fuzzy logic and Dempster-Shafer probabilities [76].

While standard occupancy grid approaches are designed for static environments, a number of extensions have been proposed to use occupancy grids for dynamic environments. Arbuckle *et al.* [3] developed the Temporal Occupancy Grid (TOG) to classify static versus dynamic objects within the local FOV of the robot. Mitsou and Tzafestas [52] proposed an extension to the TOG, called the extended TOG, where each cell maintains all sensor measurements over time. More specifically, the state of each cell at different points in time is represented in a time interval data structure, *B+ Tree*. The extended TOG representation is memory intensive, and thus, not suitable for storing and maintaining long-term map data. In addition, Wolf and Sukhatme [78] and Biswas *et al.* proposed maintaining a static occupancy grid and a dynamic occupancy grid to represent a dynamic environment.

Another approach is to use multiple maps, distributed across different times, with the maps aging (or fading) at different rates. Yamauchi and Beer [83] developed an early approach to temporal mapping using topological representations. Additionally, Andrade-Cetto and Sanfeliu [2] developed a feature-based map where features in the map aged and were removed from the map.

More recently, important work that addresses the long-term mapping problem in dynamic environments was developed by Biber and Duckett [9,10]. Their method employed multiple, sample-based maps (from laser range scans), where different sets of samples are represented on different time scales. Many of the proposed methods for dynamic environments combine more than one representation, such as multiple maps and grid-based maps such as with [77–79].

2.2.2 Long-term Mobile Robot Operation

A key objective of the life-long SLAM problem is for the mobile robot to be able to operate for extended periods, such as weeks, months or even years. The robot should have little or no human intervention. Long-term operation can be based on distance covered by the robot or long-term operation can be based on the amount of time the robot spends operating in an environment. Robots that have covered long-distance include autonomous vehicles, for example from the DARPA Grand Challenges [1], [47], [39] [40] [51] [81]. However covering long-distances does not imply that the robot revisits several locations. In our

work, we assume a large part of long-term mapping includes revisiting areas a number of times.

In addition, there has been quite a bit of work that achieved long-term localization in an environment mapped *a priori*. One early example was the Minerva museum tour guide robot developed by Thrun and colleagues at CMU [68]. Another large-scale, long-term mobile robot installation was performed by Siegwart and colleagues at the Swiss National Exhibition in 2002 [64].

In this work we are specifically interested in the robot operating in an environment for an extended amount of time. The robot will repeatedly revisit areas and changes in the environment will occur at wide-varying time scales. The Mars rovers [33, 50] Spirit and Opportunity provide a compelling example of long-term mobile robot operations. Another example is RatSLAM, where a robot operated for several hours autonomously in an indoor environment, using a biologically inspired approach to long-term operation [51]. While these long-term robot deployments are impressive, none of them entailed persistent long-term map maintenance in dynamic environments.

2.3 Discussion

There are a number of challenges to enable a mobile robot to operate for extended periods while the environment changes slowly and at random over time. Below is a summary of the challenges addressed by our DPG-SLAM method, in the context of related work (see Table 2.1).

Table 2.1: Key challenges of long-term SLAM in low-dynamic environments, and related work.

Related Work	Continuous SLAM	Change Detection	Low-Dynamic Objects	Removing Data	Long-term Model	Long-term Data
A-Cetto [2]	yes	no	yes	yes	no	no
Wolf [77–79]	yes	yes	yes	no	no	no
Mitsou [52]	yes	no	yes	no	no	no
Biber [9, 10]	no	no	yes	yes	yes	yes
Konolige [39]	yes	no	yes	yes	yes	no
Our Work	yes	yes	yes	yes	yes	yes

In comparison to the previous research described above, our proposed approach is unique in several respects. First, it utilizes state-of-the-art pose graph optimization techniques for SLAM state estimation. Second, we focus on long-term changes to the environment (low dynamic objects). Third, we aim to continuously maintain a representation over time, keeping a history of the dynamics instead of letting them fade out over time. Most importantly, we keep a current up-to-date map at all times, identifying the more static components of the world, which will facilitate path planning, localization, and exploration strategies to keep the map up-to-date.

Chapter 3

Foundation

This chapter reviews the core SLAM methods that provide the foundation for this thesis. We begin with an overview of the robot model and notation. Then we provide a review of a graphical model for SLAM, called pose graph SLAM [4,28,36,69]. Finally, we review the non-linear optimization problem to find the maximum likelihood estimate of the position of the pose graph nodes given the robot's sensor measurements.

3.1 Robot Model

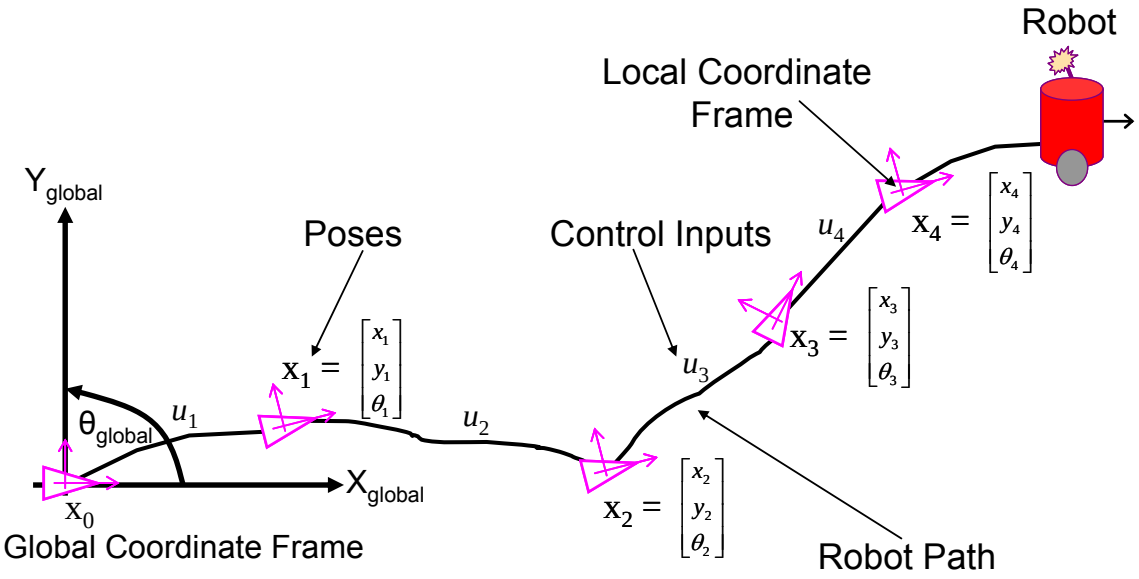


Figure 3-1: Example of a robot following a path. The poses, x_i. are shown with triangles and the control inputs, u_j, are shown on edges between the pose triangles. Each pose has its own local coordinate frame from which laser ranges are measured.

A mobile robot creates a trajectory by applying control inputs, u_i, over time as shown in Figure 3-1. The robot's positions are defined by *poses*, x_i, which specify the position (translation in the x and y directions) and the orientation (or heading) of the robot. In this thesis, a pose at time-stamp i is denoted by the vector $\mathbf{x_i} = [x_i \ y_i \ \theta_i]^T$, where x_i and y_i are

the position in a global coordinate frame and θ_i is the robot's heading also in the global reference frame. Each pose defines its own local coordinate frame shown as the small coordinate axis with each pose in Figure 3-1. A trajectory is represented by the history of poses from start time 0 to time i - defined as the set $X = \{\mathbf{x_0}, \mathbf{x_1}, ..., \mathbf{x_i}\}$, where we assume the initial position $\mathbf{x_0}$ is known. The set of all applied control inputs is $U = \{u_1, u_2, ..., u_i\}$. As the robot drives it takes sensor measurements of the environment, denoted z_i, at each pose. The set of all sensor measurements is denoted $Z = \{z_1, z_2, ..., z_i\}$

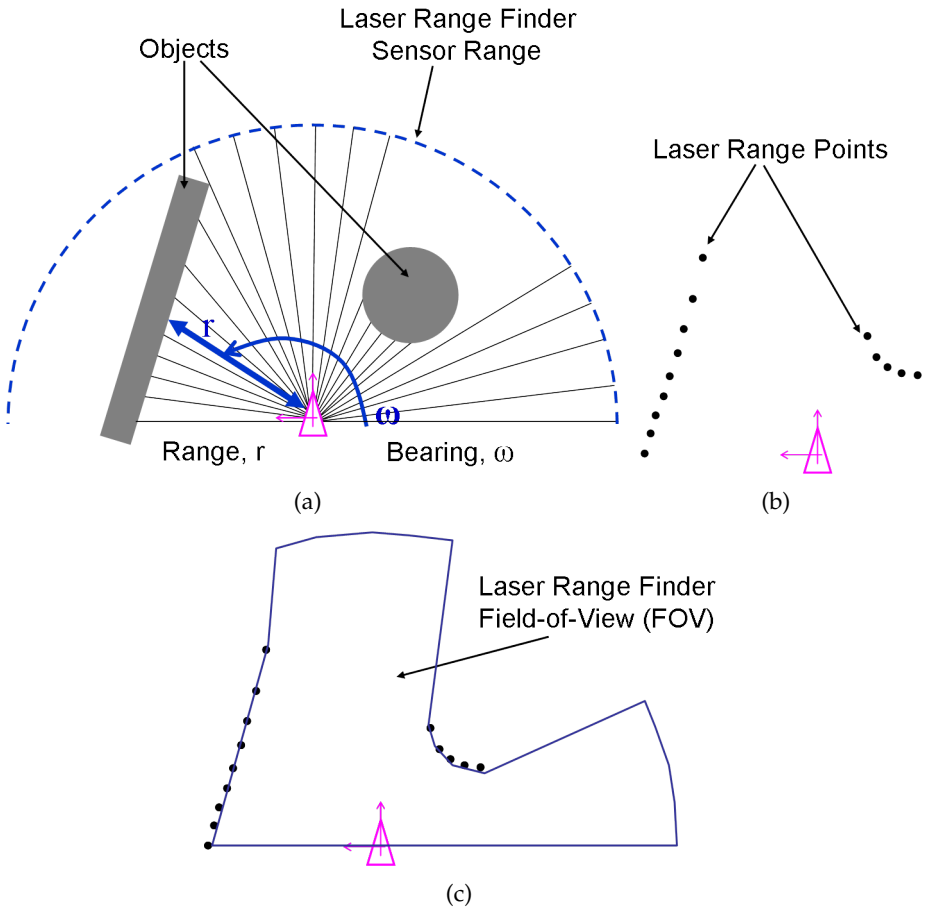


Figure 3-2: Images of the laser range finder sensor range, range points, and field-of-view (FOV).

In this thesis, we adopt existing methods to estimate the robot's trajectory given the measurements and control inputs. Each measurement is acquired by robot odometry and a laser range finder. Due to robot motion noise (eg. motors, odometry, wheel slippage) and other uncertainties caused by sensing, actuation, and environment irregularities, the true values of the poses on the trajectory are hidden or unknown. As a result, there is uncertainty in the robot poses, referred to as *pose uncertainty*. In addition, there is uncertainty or noise in the robot's sensor measurements. As mentioned, we use a laser range finder in this work. A laser range finder returns the range and bearing in a two-dimensional planar field of view to the sensor location. The collection of range and bearing values from 0^o to 180^o is called a laser scan. An example of a laser range scan is shown in Figure 3-2.

3.2 SLAM Overview

The map is a spatial representation of the environment, constructed using sensor data collected as the robots navigates through the space. A difficulty of this mission is that sensing and navigation errors grow unbounded with the passage of time. As the uncertainty in the robot's estimate of its current position increases, the accuracy of the map degrades over time. Constructing an accurate map in spite of these errors is a well-known problem in robotics known as the Simultaneous Localization and Mapping (SLAM) problem. Numerous approaches to the SLAM problem have been very effective particularly for a mobile robot operating in a static, 2D indoor environment.

A number of approaches to the SLAM problem have been applied to accurately estimate the trajectory of the robot and the map of the environment. The two premier techniques are smoothing and filtering. Smoothing techniques compute the robot trajectory using past and future measurements [25]. Filtering techniques use only the past measurements to estimate the robot position and the map [25].

Many of the traditional approaches to the SLAM problem relied on filtering methods such as the Extended Kalman Filter, or Particle Filters [6, 29, 65]. These methods maintain an estimate of the current pose of the robot and the map [4, 12, 44, 45, 65, 67, 72]. Filtering methods immediately incorporate sensor information into the current pose estimate; no additional information is propagated back to pose estimates generated at earlier points in time.

Recently, graphical methods that use smoothing have been successfully applied to the SLAM problem [16, 20, 28, 36, 37, 49, 53, 58, 65]. These approaches are graph-based and continuously update an estimate of all the poses on the trajectory by incorporating new information, eg. laser range scans, as the robot navigates. This is done at a cost of computing the full posterior over the entire robot trajectory, also referred to as smoothing [36]. Pose graph SLAM is a smoothing method and estimates the full trajectory of the robot. At present graph-based SLAM methods are considered state-of-the art in solving the SLAM problem in computation time and map accuracy [28]. In this thesis we extend pose graph SLAM methods to solve the problem of SLAM in low-dynamic environments.

3.3 Pose Graph SLAM

Pose graph methods address the SLAM problem by computing the robot's trajectory. The map is an implicitly represented by the collection of measurements, such as laser scans or images, taken at each pose along the trajectory [28, 36, 39, 49, 58]. The pose graph models the trajectory of the robot and the position of features/landmarks as a graph (see [69] for more details). A graph consisting only of poses is called a *pose graph*. The nodes denote the robot poses at different points in time and the edges denote relative spatial constraints between the poses [28]. In this work we construct the map from laser scans stored at each pose; thus our pose graph is constructed only from the set of poses that constitute the robot trajectory. This is referred to as *pose only* SLAM in [36]. Note, pose graph methods assume the environment is static. In this section we review the pose graph model, relative spatial constraints, and the trajectory estimation problem.

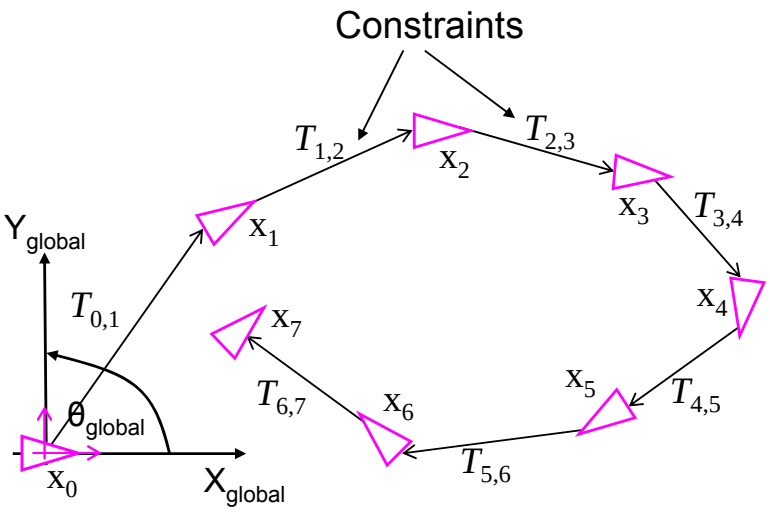


Figure 3-3: Example of a pose graph composed of a sequence of poses, $\mathbf{x_i}$, with a constraint (relative spatial transformation), $T_{i,j}$, on edges between poses.

3.3.1 Pose Graph Formulation

A pose graph, as shown in Figure 3-3, is a network of spatial relations between pairs of poses. It is defined as a connected graph $G = \langle N, E \rangle$, where $n_i \in N$ is a node in the set of all nodes $N = \{n_1, n_2, ..., n_t\}$, and $e_{i,j} \in E$ is a directed edge that connects a pair of nodes n_i and n_j. More specifically, a pose graph is defined as follows:

Pose Graph, *PG* = $\langle N, E \rangle$:

Node $n_i \in N$, where $n_i = \langle \mathbf{x}, z \rangle$,

- *Pose* $\mathbf{x} = \begin{bmatrix} x_i & y_i & \theta_i \end{bmatrix}^T$,
- *Laser range scan* $z = \{r_1, r_2, ..., r_m\}$, where m is the number of ranges in the laser scan.

Edge $e_{i,j} \in E$, where $e_{i,j} = \langle T, \Sigma \rangle$

- *Constraint* $T_{i,j}$ is computed by matching scans between two nodes n_i and n_j, where the spatial geometric transform is $T_{i,j} = \begin{bmatrix} tx_{ij} & ty_{ij} & t\theta_{ij} \end{bmatrix}^T$,
- *Covariance* Σ of the constraint $T_{i,j}$, where $\Sigma = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{x\theta} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{y\theta} \\ \sigma_{\theta x} & \sigma_{\theta y} & \sigma_{\theta\theta} \end{bmatrix}$.

An example of a pose graph is shown in Figure 3-3. Each spatial constraint is given in the local coordinate frame of a pose $\mathbf{x_i}$, that is, $T_{i,j}$ is the position of $\mathbf{x_j}$ relative to $\mathbf{x_i}$'s frame. A spatial constraint between two poses $\mathbf{x_i}$ and $\mathbf{x_j}$ is the estimate of the relative translation in the x and y directions, denoted t_x and t_y, and the relative rotation, t_θ, between the two poses (see Figure 3-5).

We use one of two types of spatial constraints for each edge in a pose graph. The two constraints are *odometry* constraints, generated by odometry measurements, and *scan*

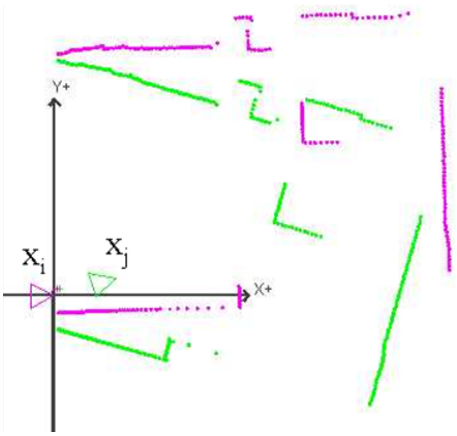


Figure 3-4: Example of two poses, x_i and x_j, and scans of the world relative to each pose.

matching where the relative transform between two scans is computed by attempting to align the scans and estimating the relative translation and rotation between the two scans. Both constraints represent the same measurement though odometry constraints tend to be more noisy. This process is also referred to as the *registration* process (see Figure3-4). It is assumed that the spatial constraint that is returned from scan matching follows a Gaussian distribution (as given in the pose graph definition). In this work, we use a common method for scan matching known as the Iterative Closest Point (ICP) method [7, 48, 62, 82]. In this work, we seed the ICP algorithm with an initial guess derived from odometry. If a sufficient registration cannot be found then the odometry measurement is used as the constraint.

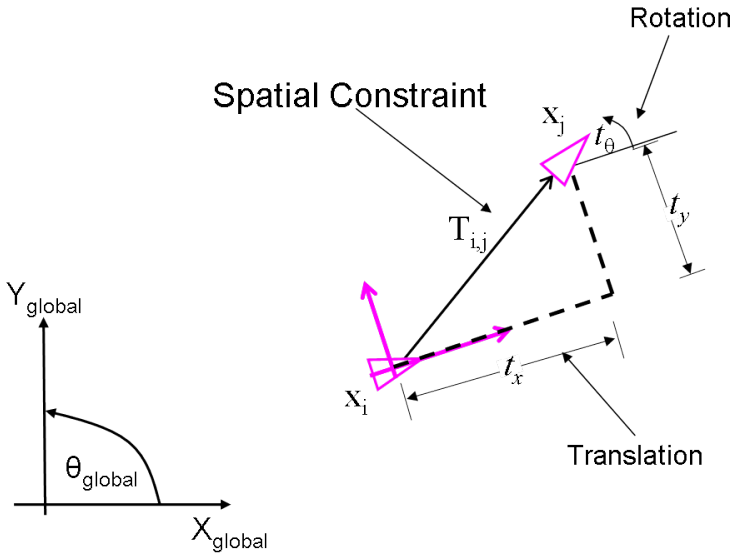


Figure 3-5: Two poses x_i and x_j with a constraint $T_{i,j}$ between them. The values of the constraint t_x, t_y, and t_θ are illustrated in the figure and show the location of pose x_j in the coordinate frame of pose x_i.

Given a set of relative constraints between a sequence of nodes (ie. a pose graph with no loop edges that induce a cycle), the position of each node can be referenced in the global coordinate frame by applying the *compound operator*, denoted "$\oplus$". The compound operator is described in [65] as follows,

$$T_{i,k} = T_{i,j} \oplus T_{j,k} = \begin{bmatrix} tx_{jk} \cos\ t\theta_{ij} - ty_{jk} \sin t\theta_{ij} + tx_{ij} \\ tx_{jk} \sin\ t\theta_{ij} - ty_{jk} \cos t\theta_{ij} + ty_{ij} \\ t\theta_{ij} + t\theta_{jk} \end{bmatrix} \tag{3.1}$$

The compound operator computes the relative constraint from pose $\mathbf{x_i}$ to pose $\mathbf{x_k}$ through pose $\mathbf{x_j}$. For example, to compute the position of pose x_3 relative to pose x_0 in Figure 3-3,the compound operator would be applied as follows $T_{0,3} = T_{0,1} \oplus T_{1,2} \oplus T_{2,3}$. Similarly, the reference frame of a pose $\mathbf{x_j}$ given in the reference frame of another pose $\mathbf{x_i}$ can be inverted to get the position of $\mathbf{x_i}$ in the frame of $\mathbf{x_j}$ by applying the *reverse operator*, $T_{j,i} = \ominus T_{i,j}$ [65]. The reverse operator is defined as follows,

$$T_{j,i} = \ominus T_{i,j} = \begin{bmatrix} -tx_{ij} \cos\ t\theta_{ij} - ty_{ij} \sin t\theta_{ij} \\ tx_{ij} \sin\ t\theta_{ij} - ty_{ij} \cos t\theta_{ij} \\ -t\theta_{ij} \end{bmatrix} \tag{3.2}$$

In general, combinations of the compound and reverse operators can be applied to change coordinate frames between pairs of poses. The approximate covariances of these functions are also given in [65]. Note that these operators are non-linear functions as a result of the robot poses including a heading (the robot's orientation).

3.3.2 Loop Constraints and Pose Graph Optimization

In the previous section we described how the compound operator can be used to compute the position of a pose in the global frame. Using the compound operator results in the best estimate of the pose when the pose graph is a sequence of poses and contains no cycles. When an edge is added to the pose graph that introduces a cycle it is referred to as a *loop constraint*. An example of pose graph with a loop constraint is shown in Figure 3-6, where the constraint $T_{7,1}$ is a loop constraint connecting poses $\mathbf{x_7}$ and $\mathbf{x_1}$.

A loop constraint is a constraint that is inserted as a result of the robot navigating to a part of the environment it has previously visited (see [19, 26, 49, 57] for more detail). For example, in Figure 3-6 the robot drives around in an ellipse, and returns to the part of the environment where it started. Thus, poses x_7 and x_1 are approximately at the same location, at least some or all of their sensor ranges overlap and as a result the robot is able to "recognize" common features. When a loop constraint is added, a pose estimate can be derived from more than one node in the pose graph. If it is known that x_7 is at least close to x_1, then the robot can attempt to compute the relative spatial constraint between the two poses using scan matching. Then that constraint can be inserted into the pose graph between nodes x_1 and x_7, as a loop constraint.

Generally, as the robot navigates, errors in its motion measurements and transforms accumulate. We assume these measurements are corrupted by zero-mean Gaussian noise. Small errors in the constraint estimates and sensor noise accrue with the addition of each pose in the trajectory. As a result, the uncertainty in the robot poses along the trajectory

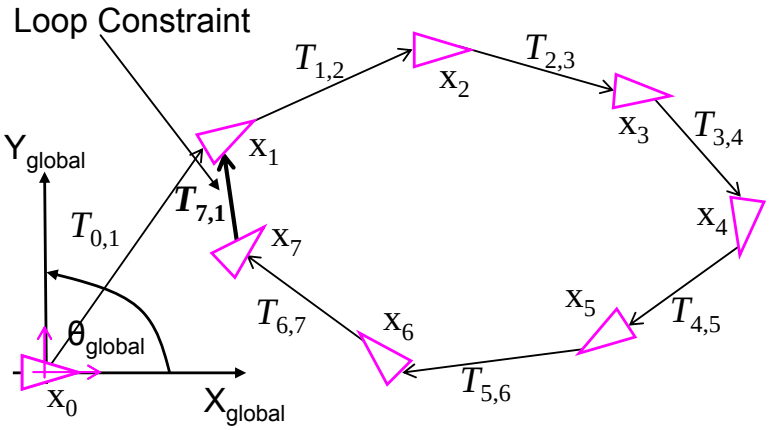
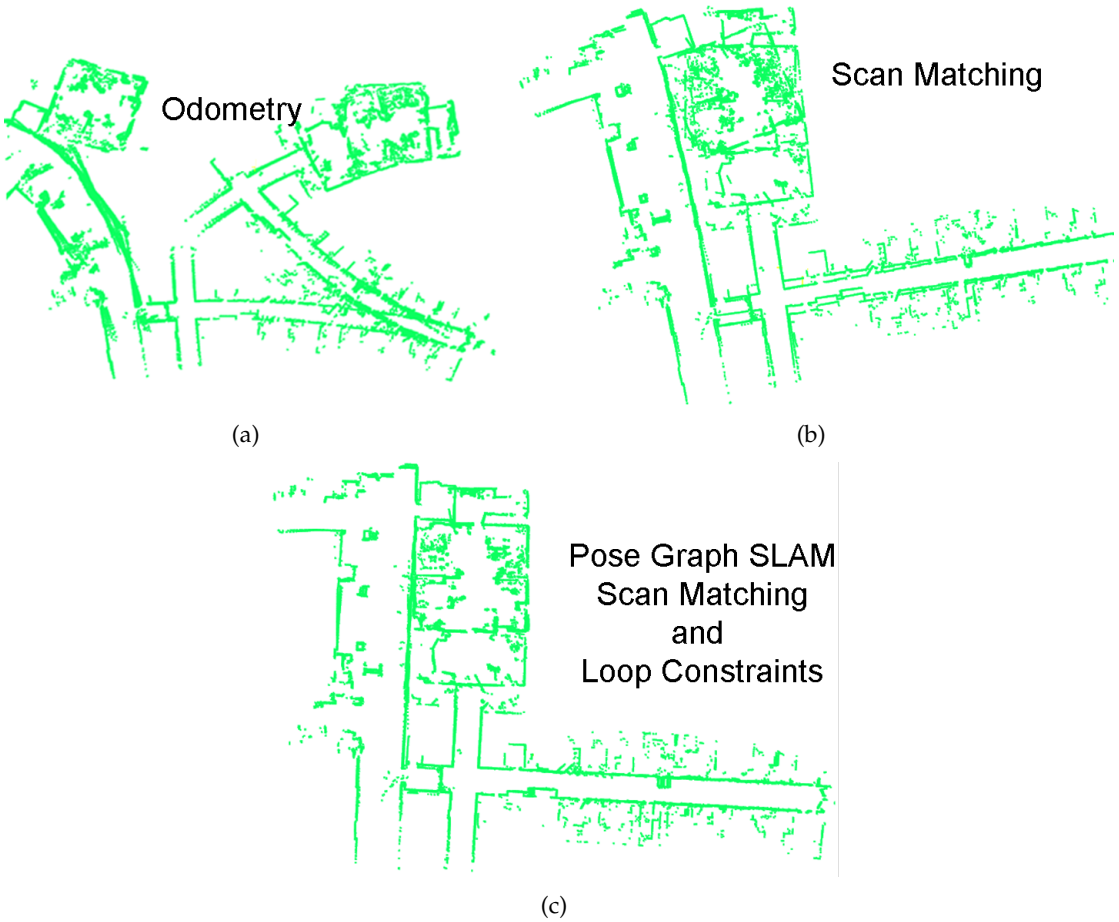


Figure 3-6: Example of a pose graph where a cycle is introduced as a result of a loop constraint, $T_{7,1}$, between poses $\mathbf{x_7}$ and $\mathbf{x_1}$.



(a) (b) (c)

Figure 3-7: Example maps generated from pure odometry, scan matching, and pose graph SLAM using iSAM (incremental smoothing and mapping) [36].

cause the trajectory estimate to drift from the ground truth, the actual robot path, over time. The covariances of the pose estimates can be reduced if a loop constraint is inserted into the pose graph. In this work we compute a loop constraint between nodes n_i and n_j by matching the two scans, z_i and z_j, and inserting the edge representing the relative transform [8, 48, 57].

3.3.3 Trajectory Estimation Problem

We summarize the probabilistic problem formulation of computing the full SLAM trajectory for pose graphs with any number of loop constraints. A pose graph contains a collection of poses and relative constraints that represent the trajectory X. The poses are referred to as variables that are estimated by various pose graph optimization methods. The general probabilistic problem formulation for pose graph SLAM, ie. SLAM without features/landmarks, is given by as $P(X, U, T)$. This formulation represents the joint probability over the robots poses X, the control inputs U, and the set of all constraints T.

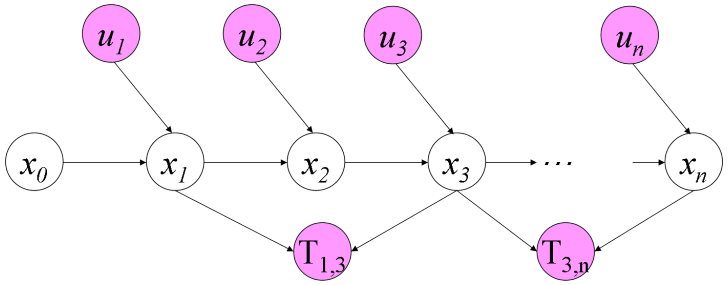


Figure 3-8: In the figure, the hidden variables are the robot poses, x_i. The transformation $T_{i,j}$ is a derived measurement computed from two laser range scans, z_i and z_j.

The Bayesian network of the trajectory estimation problem is shown in Figure 3-8. The robot applies control inputs u_i from pose to pose and scan matching constraints, denoted $T_{i,j}$, are computed for some m pairs of poses. Given the Bayesian network model, Eq. (3.3) factors as follows,

$$P(X, U, T) = P(\mathbf{x_0})P(u_1)...P(u_n)\prod_{i=1}^{n} P(\mathbf{x_i}|\mathbf{x_{i-1}}, u_{i-1})\prod_{k=1}^{m} P(T_{j,q}^{k}|\mathbf{x_j^k}, \mathbf{x_q^k}), \tag{3.3}$$

where pose $\mathbf{x_j}$ and pose $\mathbf{x_q}$ are referred to as $\mathbf{x_{q_k}}$ and $\mathbf{x_{q_k}}$ to denote that the k^{th} constraint depends on the two poses, and j and q are indexes referring to any pose along the trajectory.

We adopt the common SLAM assumption to formulate the robot's motion model [4, 69], resulting from odometry measurements. The robot's motion is represented as the following Gaussian noise model, where w_t is zero-mean Gaussian noise with covariance Q_t,

$$\mathbf{x_t} \leftarrow f(\mathbf{x_{t-1}}, u_t) + w_{t-1}, \tag{3.4}$$

where u_t is the control input applied from pose $\mathbf{x_{t-1}}$ to pose $\mathbf{x_t}$.

Oftentimes the relative rotation and translation between two poses can be computed more accurately by matching laser range scans taken at each pose. It is assumed that there is sufficient overlap between the two scans, z_i and z_j, and scan matching results in the

transformation, $T_{i,j}$, between two poses x_i and x_j. This process is also assumed to be corrupted by zero-mean Gaussian noise b_t, with covariance Σ as given in Eq. (3.5). The function $h(\mathbf{x_i}, \mathbf{x_j}) = \mathbf{x_i} \ominus \mathbf{x_j}$.

$$T_{i,j} \leftarrow h(\mathbf{x_i}, \mathbf{x_j}) + b_t \tag{3.5}$$

Given noise in the motion model and measurement models, nodes in a pose graph also include the covariance of the motion model, $n_i = \langle \mathbf{x_i}, Q_i, z_i \rangle$. As mentioned earlier pose graph edges contain the covariance of the constraint as well, $e_{i,j} = \langle T_{i,j}, \Sigma_{i,j} \rangle$. A general approach to compute the trajectory is to compute the maximum likelihood (or maximum a posteriori in some cases [36]) estimate of the trajectory of the joint probability as given below,

$$X^* = \arg\max_X P(X, U, T) \tag{3.6}$$

To compute the maximum likelihood estimate of the joint probability we minimize the negative log of Eq. (3.6) using the squared Mahalanobis distance, resulting in the non-linear least squares problem given in Eq. (3.7), where the priors are dropped for simplicity (see [16, 28, 42] for details).

$$\begin{aligned}
X^* &= \arg\max_X P(X, U, T) \\
&= \arg\min_X -\log P(X, U, T) \\
&= \arg\min_X -\log(\prod_{i=1}^{n} P(\mathbf{x_i}|\mathbf{x_{i-1}}, u_{i-1}) \prod_{k=1}^{m} P(T_{j,q}|\mathbf{x_j}^k, \mathbf{x_q}^k) \\
&= \arg\min_X (\sum_{i=1}^{n} -\log P(\mathbf{x_i}|\mathbf{x_{i-1}}, u_{i-1}) \sum_{k=1}^{m} -\log P(s_k|\mathbf{x_j}^k, \mathbf{x_l}^k) \\
&= \arg\min_X [\sum_{i=1}^{n} ((\mathbf{x_i} - f(\mathbf{x_{i-1}}, u_{i-1})) Q_i^{-1} (\mathbf{x_i} - f(\mathbf{x_{i-1}}, u_{i,1}))) \\
&\quad + \sum_{j=1,k=1}^{m} (T_{i,j}^k - h(\mathbf{x_i^k}, \mathbf{x_j^k})) \Sigma_k^{-1} (T_{i,j}^k - h(\mathbf{x_i^k}, \mathbf{x_j^k}))]
\end{aligned} \tag{3.7}$$

To solve the non-linear least squares problem in Eq. (3.7), a number of different pose graph optimization methods have been applied [16, 36, 58]. These methods compute the positions of all the nodes in the pose graph according to a cost function. The methods minimize the difference between the measurement $T_{i,j}$ and the prediction $\mathbf{x_t}$. More specifically, maximizing the likelihood in this model is equivalent to minimizing the square Mahalanobis distances. Thus, the goal is to compute the trajectory estimate that simultaneously minimizes the costs. This thesis uses iSAM (incremental smoothing and mapping [36]) for pose graph optimization in order to compute the best trajectory estimate X^*.

Chapter 4

Long-Term Mapping Problem

We aim to develop algorithms for a mobile robot to operate in an environment for long periods. In particular, our research focuses on equipping the robot with the ability to navigate in, and maintain an up-to-date map of, a dynamic environment. A mobile robot navigating for long periods in a dynamic environment will encounter objects that move at varying time-scales. In this thesis, objects are not extracted from the environment as in the recognition problem. Instead, this thesis focuses on how moving objects cause changes in the robot's representation or map of the environment, and implications for robot navigation in such environments. These map differences are referred to as changes. Thus, changes occur at varying time scales.

Much of the work in traditional SLAM assumes that the environment is made up of static, or static and transient objects, such as people or vehicles [31, 64, 68]. These approaches also assume the robot is operating for shorter periods of time [3, 32, 54, 66, 75]. While these methods have proven to be successful in indoor and some outdoor settings, the problem of life-long SLAM remains open. This thesis explores the important and critical open research problem of continuous long-term mobile robot navigation in dynamic environments. In this chapter we present the problem formulation and assumptions, and define a novel model called the Dynamic Pose Graph (DPG) to represent the trajectory and map of a dynamic environment, and outline our goals for long-term mapping.

4.1 Problem Formulation

For a mobile robot to be able to navigate in a dynamic environment and maintain an up-to-date map, it must be able to continuously localize and detect changes. We refer to this process as concurrent localization and change detection. To determine if the environment has changed when a robot revisits a location, poses at that location must be well localized relative to each other. Subsequently, older scans at that location may be compared to the most recent scans to determine whether or not a change has occurred, and if so, where the changes have occurred. Therefore, it is necessary to know the pose estimate along the trajectory as well as be able to detect if a change has occurred.

In this section, we detail three key components of the long-term mapping problem explored in this thesis. First is general probabilistic change detection and trajectory estimation formulation. The second is our dynamic environment model. Lastly, we describe the set of assumptions used in our approach.

4.1.1 Change Detection and Trajectory Estimation Problem

Our aim is to estimate the trajectory and the changes given the constraints, control inputs, and laser scans. Recall from Chapter 3, X represents the robot trajectory, Z is the collection of laser scans that make up the map, U is the set of control inputs, and T is the set of constraints between poses. The set of all changes $C = \{c_1, c_2, ..., c_r\}$ corresponds to each pose, and indicates if a change was detected at that pose. Thus, the concurrent change detection and trajectory estimation problem can be described by the expression $P(X, C|Z, U, T)$. The expression $P(X, C|Z, U, T)$ is expanded in Eq. (4.1) as follows,

$$P(X, C|Z, U, T) = P(C|X, Z, U, T)P(X|Z, U, T). \tag{4.1}$$

The term $P(X|Z, U, T)$ has constraints T that are derived from matching scans in Z. Given the constraints T, the set of laser scans Z is independent in the expression because it provides no additional information. Therefore we write $P(X|Z, U, T) = P(X|U, T)$. In the term $P(C|X, Z, U, T)$, given the trajectory X the control inputs U and the constraints T are independent. The laser scans Z are needed in order to detect change and compute C. As a result, $P(C|X, Z, U, T) = P(C|X, Z)$. Eq. (4.1) is re-written in (Eq. (4.2)) as follows,

$$P(X, C|Z, U, T) = P(X|U, T)P(C|X, Z). \tag{4.2}$$

In this work, we approximate solutions to the two distributions in Eq. (4.2) separately. Pose graph optimization using iSAM is applied to compute the trajectory, $P(X|U, T)$. Then given the best estimate of the trajectory, we use a binary function to compute whether changes have occurred (see Chapter 5) to approximate $P(C|X, Z)$.

4.1.2 Dynamic Environment Model

The process by which the environment changes over time is independent of how the robot navigates and is not directly known. The dynamic environment model includes the types of objects, both moving and stationary. These objects move at wide-varying time scales from seconds (such as people walking) to weeks (such as a piece of furniture moved from one location to another). The model is a composition of objects of three types: static, high-dynamic, and low-dynamic. The terms high-dynamic and low-dynamic were presented in [52] to describe general dynamic environments. Below is a description of each type of object.

Dynamic Environment Model, $Env = S \cup F \cup D$**:**

Static Objects, S: A static object, s_i, is one that remains stationary; ie. never moves. Such examples include walls and columns. The set of static objects is $S = \{s_1, s_2, ..., s_j\}$.

High-dynamic Objects, F: A high-dynamic object, f_i,is a transient object, such as a person walking, vehicles, or continuous moving artifacts like soccer balls. These objects appear in motion while in the robot's field-of-view. The set of all high-dynamic objects is $F = \{f_1, f_2, ..., f_m\}$.

Low-Dynamic Objects, D: A low-dynamic object, d_k, are those objects that move much slower than the robot navigates, and therefore their motion cannot be sensed immediately. That is, they do not appear in motion while in the robot's field-of-view. Often, these objects are stationary most of the time and, when moved, they appear to

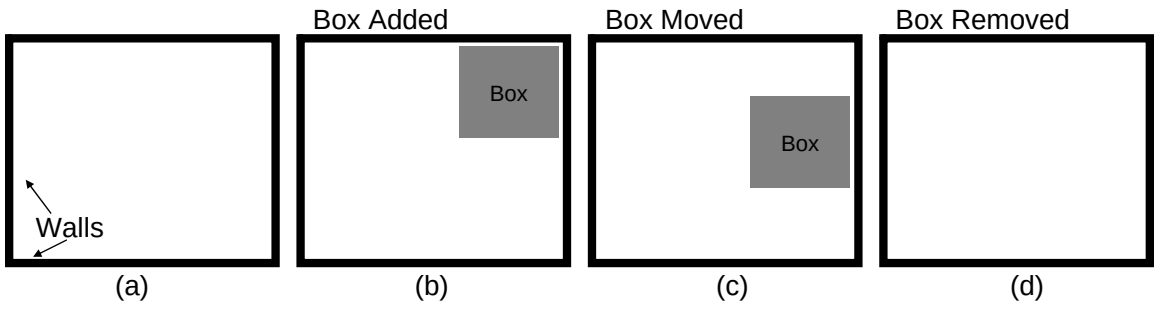


Figure 4-1: Illustration of the three types of environment change that can occur between passes. Four snapshots of the same room are shown, where a box is added, moved, and then removed. To move the box it is added to its new location, and removed from its previous location.

move abruptly and seemingly at random. Examples include trash cans, desks, and temporary installations. The set of low-dynamic objects is $D = \{d_1, d_2, ..., d_k\}$.

4.1.3 Assumptions

For long-term operation, it is assumed that the robot makes numerous *passes* through the environment to maintain an up-to-date map as well as capture the environment dynamics. A *pass* describes the robot navigating and covering either part of the environment or covering the entire environment. A robot executes a pass from a known start (or home) location, and completes the pass by returning to the start location. The general assumptions applied throughout this thesis are listed below.

Assumption 1: Dynamic Environment Model. The robot operates in an environment where the dynamics, places where objects are added, moved, or removed, are unknown. We assume that the environment contains static and low-dynamic objects, ie., $Env = S \cup D$. There are no high-dynamic objects F included in our environment model. high-dynamic objects can be filtered by existing methods including [17,22,31,74,79]. The space is a bounded two-dimensional indoor, office environment. In addition, changes are a result of low-dynamic objects being added, removed, or moved (see Figure 4-1). In order for an object to move it must be removed from its current position and added to its new position. We assume that if changes occur, then they occur between passes.

Assumption 2: Robot Navigation and Sensing. The robot navigates over the long-term by executing a number of passes. Each pass starts and ends at the same known start location, x_{start}. The robot is equipped with a two-dimensional, planar forward-facing laser range scanner, where each scan returns ranges values measured from angles between 0^o to 180^o. While navigating, the robot takes laser range measurements to construct and maintain an accurate map. The trajectory that the robot follows on different passes can overlap. Thus, the robot may take several measurements of the same location at different points in time (eg., measurements taken during different passes). An example of passes that overlap in the same location is given in the pose graph in Figure 4-2.

Assumption 3: Trajectory Estimation. The pose graph model is used to represent the robot's trajectory. To estimate the location of poses in the pose graph, pose graph optimization with incremental smoothing and mapping (iSAM) is used [36]. Additionally, generating loop closing constraints is a critical procedure to pose graph SLAM. We assume that our loop closing method generates constraints that, in turn, are used to improve pose estimates.

4.2 Example of Four Passes Through An Indoor low-dynamic Environment

In traditional SLAM, the robot makes one pass and low-dynamic objects are assumed to be static and can be included in the map. If the robot encounters high-dynamic objects such as people, several methods have been proposed to filter out or track these transient objects while constructing a static map. However, traditional SLAM methods that make only one pass are not sufficient to discern low-dynamic objects.

We present an example of a robot making four passes through an indoor environment, the MIT CSAIL Reading Room. Figure 4-2 shows four individual maps generated after each pass through the Reading Room. One or two low-dynamic objects, boxes denoted, d_1^p and d_2^p where p is the pass number, are removed, are added or moved after each pass.

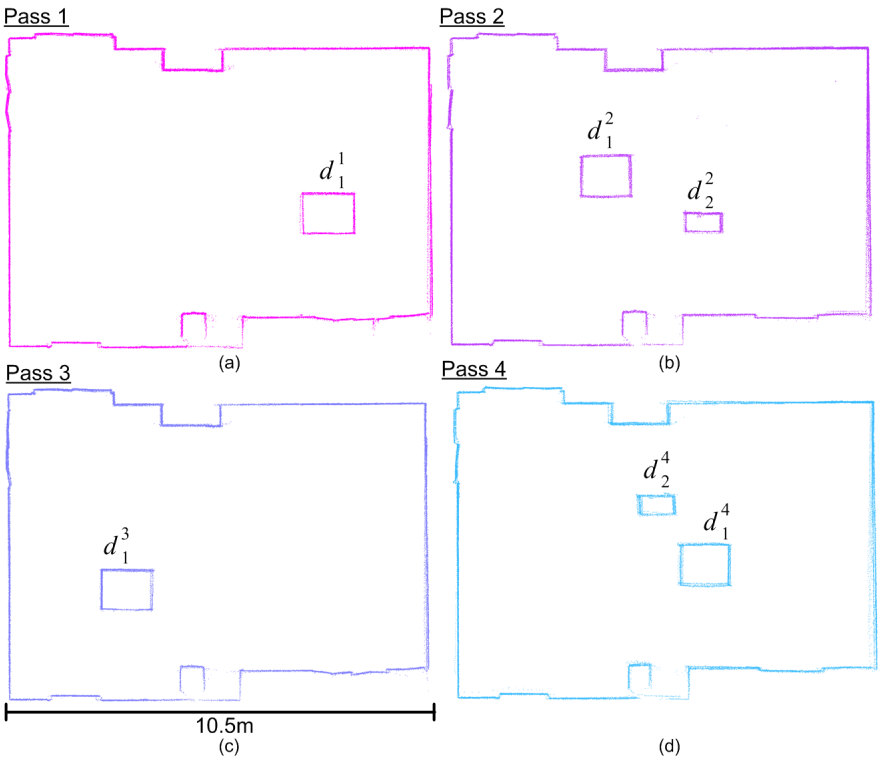


Figure 4-2: Example maps of four individual CSAIL Reading Room maps after four passes. Subscripts of each box (low-dynamic object) are used for identification and superscripts are used to denote the pass number.

4.2.1 Pose Graph SLAM Representation

Consider a scenario in which a mobile robot equipped with a laser range finder continuously navigates in an indoor room. The robot contains on-board computational capabilities to perform pose-graph SLAM while it navigates. The robot is tasked with exploring and maintaining an up-to-date map of the room. It computes pairwise and loop constraints using scan matching and applies pose graph optimization methods to obtain an accurate trajectory estimate and a map. At the same time, low-dynamic objects are added, moved, or removed as the robot makes a number of passes through the room.

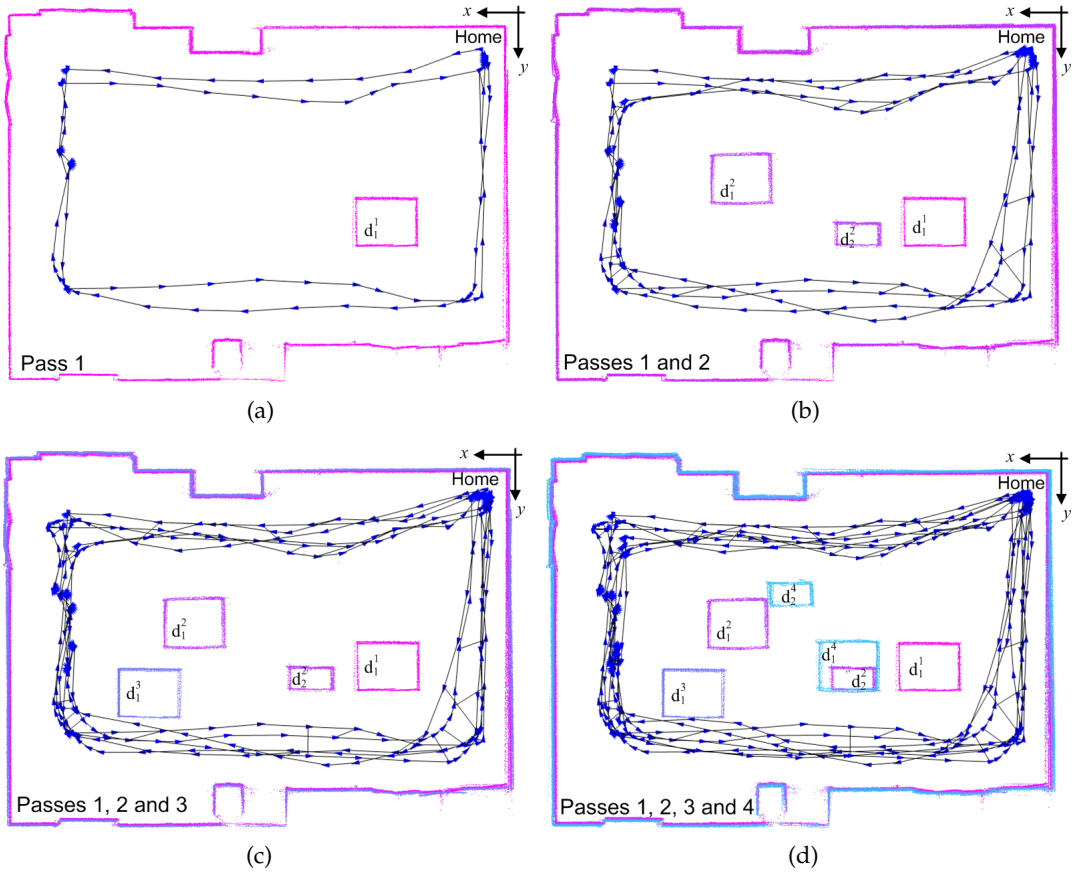


Figure 4-3: The pose graph and maps generated after each of the 4 passes through the CSAIL Reading Room. The images shows that with pose graph SLAM the low-dynamic objects from various locations are all included in the map. The room contains two boxes which are labeled d_1 and d_2 and are low-dynamic objects.

Figure 4-3(a) shows the pose graph and the map from the robot's first pass through the environment. There is one low-dynamic object d_1^1 (the superscript is the pass number that the box is at the location). Before the second pass the low-dynamic object d_1 is moved to where d_1^2 is located, and a second object is added d_2^2 (see Figure 4-3(b)). After the second pass, d_1 is moved and d_2 is removed. Finally, after the third pass both boxes are in the room, and are moved to different locations than previous passes (see Figure4-3(d)). A three-dimensional side view of the pose graph SLAM map generated after 4 passes is shown in Figure 4-4, where submaps from each pass are shown on the z-axis.

The example illustrates that in a dynamic environment, the set of all laser scans pro-

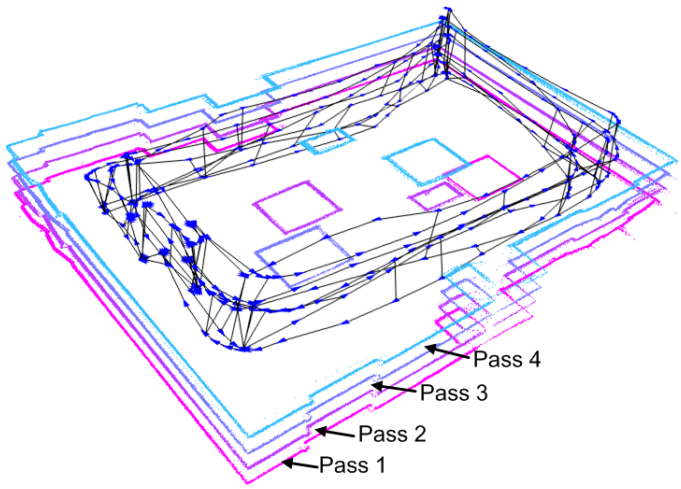


Figure 4-4: A three-dimensional side-view of the pose graph SLAM map highlighting the individual passes. The figure depicts pose graph SLAM maps generated after the robot makes 1, 2, 3, and then 4 passes through the CSAIL Reading Room.

duced by pose graph SLAM yield an inaccurate representation of the environment. The inaccuracies in the map arise as a result of changes caused by low-dynamic objects. When low-dynamic objects move the previous (older) scans of the environment become *stale* and do not reflect the current map. The example also illustrates how the pose graph grows as a function of the length of the robot trajectory and not as a function of the size of the room.

4.2.2 Pose Graph Representation of Low-dynamic Environments

One of the aims of the DPG-SLAM method is to be able to identify stale data and remove it from the current map. Figure 4-5 illustrates the best-case map after the robot makes four passes through the environment shown in Figure 4-2. In the figure the low-dynamic objects from passes 1-3 have been removed, leaving the ranges from d_1^4 and d_2^4. If d_1^3 was at the same position as d_1^4 then the ranges of d_1 from both passes would be included in the best-case map. Another aim of the DPG-SLAM method is to retain data from static objects which ensures that the map representation captures not only the most recent map but the "more static" parts of the environment as well.

The resulting map should represent both the static and the dynamic parts of the environment. By maintaining the points from ranges of low-dynamic objects that have been added, moved, or removed, the map can also represent temporal dynamics where parts of the environment change and the frequency of the changes. Figure 4-6 depicts the best-case dynamic map of the boxes as they are added and removed from pass to pass. Our goal is to develop a method where the robot incrementally constructs a representation of the environment online. The representation (or map) should contain the ranges of the static and dynamic parts of the environment as depicted in Figure 4-5 and Figure 4-6.

In addition, it is important to consider the pose graph from our example described above. In the Reading Room data set the robot is driven along an "out-and-back" trajectory as shown in Figure 4-7. The robot travels out for some distance and returns home along the same trajectory. As a result, redundant range measurements from objects such as those that are parallel to the robot's path are collected. An example of redundant mea-

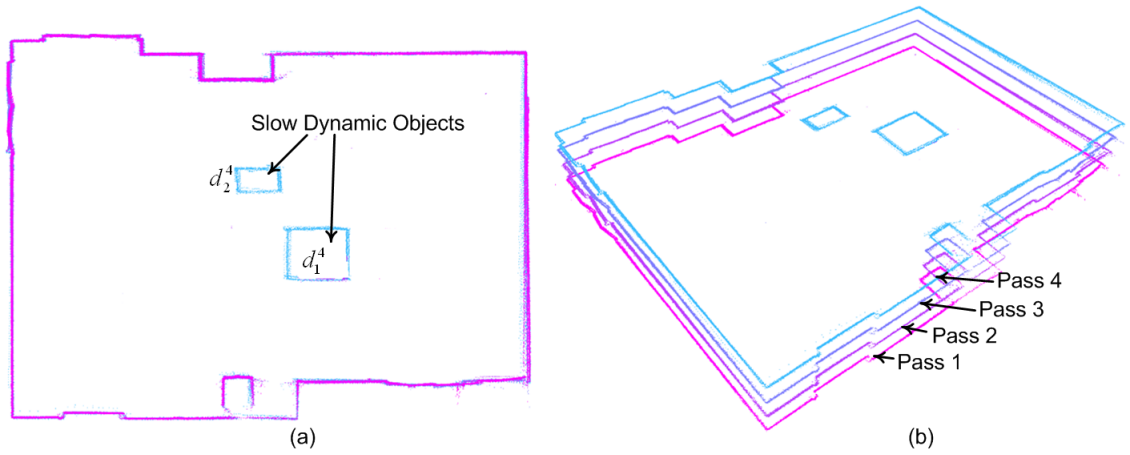


Figure 4-5: Illustration of the best-case map after four passes through the CSAIL Reading Room. The most recent state of the environment can be seen in (a), where range measurements from objects that moved during previous passes are removed from the map. (b) shows a side-view of the best-case map where laser range measurements from the static parts of the environment (the walls) remains in the map.

surements taken at poses along the trajectory are measurements from the walls that are collected as the robot goes out and returns along the same trajectory. Thus, the pose graph shown in Figure 4-7 could be potentially reduced to half of its size over time (after the robot completes its current pass). On the other hand, registrations (matching scans) across different passes contain additional information.

Also, from pass to pass there will be nodes with ranges from objects that have since moved, thus the ranges should not be included in the current map. That is, when the robot revisits an area, pose graph nodes containing old data, for example, could be removed as new nodes with current information (range measurements) are added to the pose graph. These observations about pose graph SLAM applied to a mobile robot operating in low-dynamic environments served as motivation to develop a sophisticated model that is robust to changing environments and can generate an accurate and up-to-date map.We call this model the Dynamic Pose Graph.

4.3 The Dynamic Pose Graph Model

This section presents a novel model called the Dynamic Pose Graph (DPG) to address the problem of long-term mapping in dynamic environments. The DPG is an extension of the SLAM pose graph model. A suite of algorithms are applied to a DPG that enable a mobile robot to continuously navigate and maintain an accurate and up-to-date map.

A Dynamic Pose Graph is a connected graph, denoted $DPG = \langle N, E \rangle$, with nodes $n_i \in N$ and edges $e_{i,j} \in E$, is defined as follows:

Dynamic Pose Graph, *DPG* = $\langle N, E \rangle$:

Node $n_i \in N$, where $n_i = \langle \mathbf{x}, c, a, p, z \rangle$,

- *Pose* $\mathbf{x} = \begin{bmatrix} x_i & y_i & \theta_i \end{bmatrix}^T$,

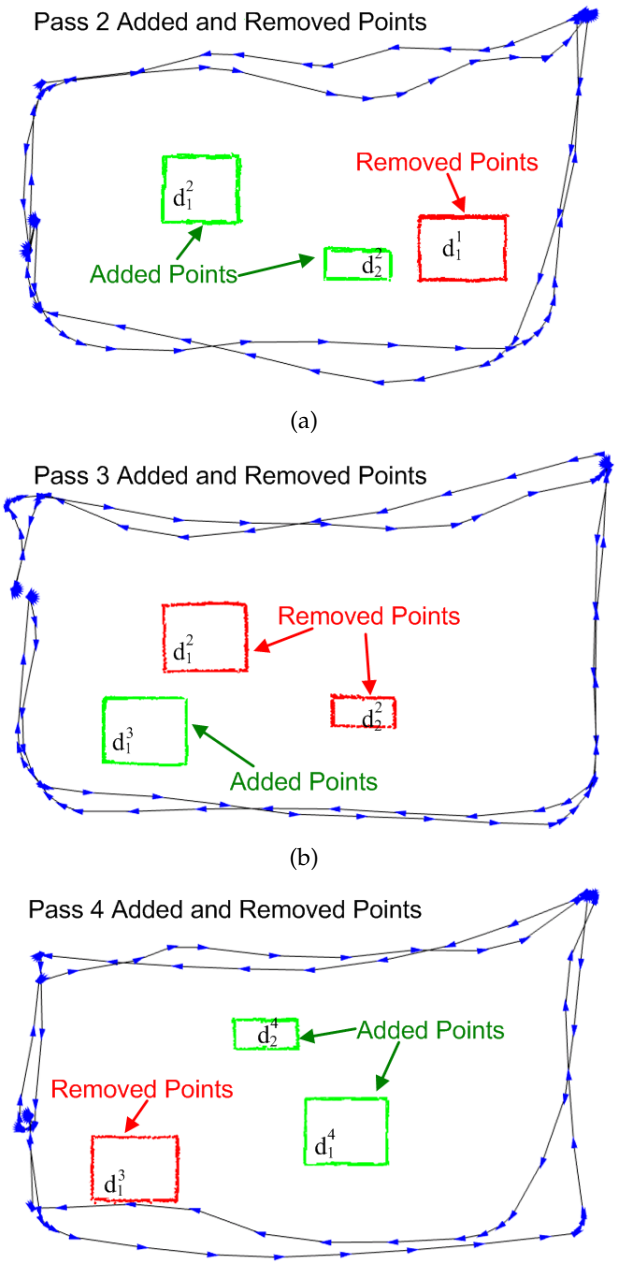


Figure 4-6: Illustration of the points that change from pass 1 to pass 2, pass 2 to pass 3, and from pass 3 to pass 4. Our goal is to include this information of the dynamics of the environment in our best-case map.

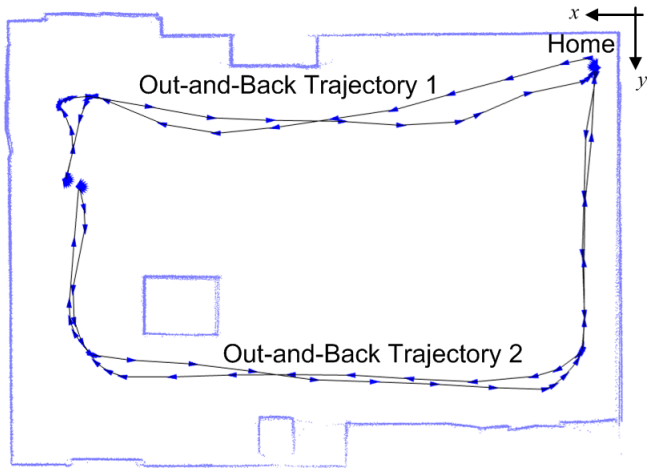


Figure 4-7: Example of the two out-and-back trajectories that the robot followed to map the room.

- *Change node indicator* $c = \begin{cases} 1, & \text{if change detected} \\ 0, & \text{if no change detected} \end{cases}$,
- *Active node indicator* $a = \begin{cases} active, & \text{if one or more } \textit{sectors} \text{ (see definition below) are on} \\ inactive, & \text{otherwise} \end{cases}$,
- *Pass number* p, an integer representing the pass at which the node was created and added to the dynamic pose graph,
- *Laser range scan* $z = \langle r, \alpha \rangle$,
  - *Range values and labels* $\psi = \{\langle r_1, l_1 \rangle ..., \langle r_m, l_m \rangle\}$, where m is the number of ranges in the laser scan. *Range labels* $l_k = \{static, added, removed\}$ are used to denote which laser range measurements are derived from a low-dynamic object or a static object. Recall that a low-dynamic object is either added removed, or a combination of the two to an environment. Each measurement is labeled according to whether or not it was derived from a static, added, or removed object, while the robot navigates and updates its map (see Chapter 5).
  - *Sectors* α, divide the laser scan into equal-sized partitions based on angle. The purpose of sectors is to retain as many accurate ranges as possible by dividing the laser scan and removing parts of the scan that are inaccurate and should not be included in the map. Each laser scan has a set of sectors, $A = \{ \alpha_1, \alpha_2, ..., \alpha_b \}$, where b is the number of sectors. For example, if a laser scan is divided into 10 sectors and has 181 range measurements from 0^o to 180^o, then the ranges are divided by angle of 18.1^o. Additionally, each sector can be in one of two states, either *on* or *off*. Once a sector is in the *off* state it cannot transition back to the *on* state. When all sectors of a laser range scan are *off*, the node it corresponds to is labeled inactive.

Edge $e_{i,j} \in E$, where $e_{i,j} = \langle T, \Sigma \rangle$

- *Constraint* $T_{i,j}$ is computed by matching scans between two nodes n_i and n_j,

where the spatial geometric transform is $T_{i,j} = \begin{bmatrix} tx_{ij} & ty_{ij} & t\theta_{ij} \end{bmatrix}^T$,

- *Covariance* Σ of the constraint $T_{i,j}$, where $\Sigma = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{x\theta} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{y\theta} \\ \sigma_{\theta x} & \sigma_{\theta y} & \sigma_{\theta\theta} \end{bmatrix}$.

An example of a dynamic pose graph is given in Figure 4-8. In the figure, the DPG is shown originating from the known start at each pass. An example of a node in a DPG is shown in Figure 4-9, where (a) shows a node from a typical pose graph. Images (b) and (c) show additional properties included in a DPG node.

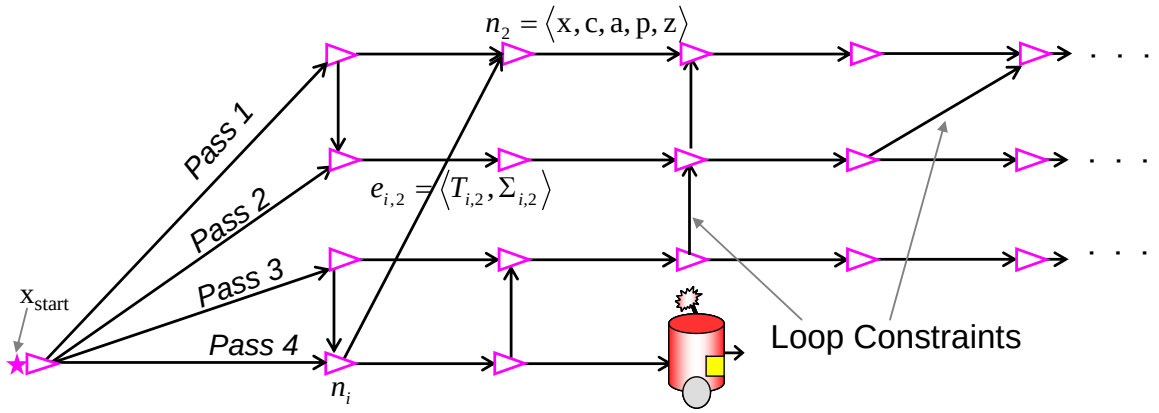


Figure 4-8: Example of a Dynamic Pose Graph. Two nodes $\mathbf{n_2}$ and $\mathbf{n_i}$ and a loop constraint between them are shown with their DPG notation.

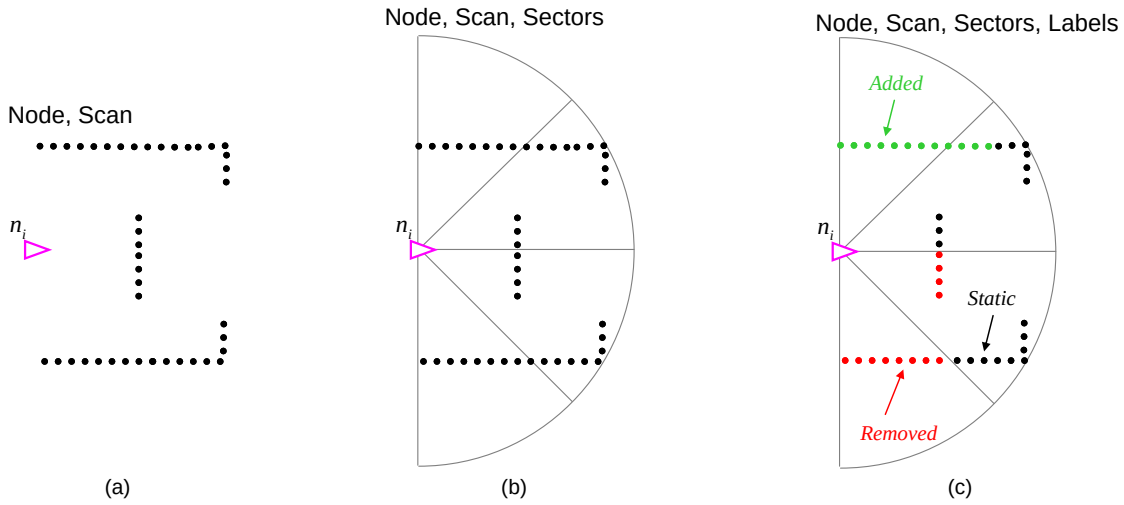


Figure 4-9: Illustration of a DPG Node and its contents. The node in (c) is a complete DPG Node with sectors and a laser range scan where the points on the scan are labeled static, added, or removed.

4.3.1 DPG Long-term Map Representation

Recall from Chapter 3 that a mobile robot equipped with a laser range scanner can navigate a static environment and compute its trajectory and map by applying pose-graph SLAM methods. Given a set of relative spatial constraints between poses along the robot's trajectory, denoted X, pose graph optimization methods yield an estimate of the poses. The set of poses is referred to as the robot's trajectory, and is denoted X. Then, with the updated pose estimates, the robot can generate a map by projecting the laser scans taken at each pose into a single global coordinate frame. Also, in traditional pose graph SLAM it is assumed that the environment is static. As a result, the map Z can be represented by the set of all laser scans, $z_i \in Z$. In Dynamic Pose Graph SLAM an accurate map is constructed from parts of scans that accurately reflect the environment. This is illustrated in the map shown in Figure 4-7b.

To represent a changing environment, the DPG maintains two maps an *active map* and a *dynamic map*.

Active Map: Z_{active} = { r_i | r_i is a range point from the scan z_j, and $\alpha(r_i)$ = on, and l_i is the range label and $l_i \in \{static, added\}$ and node(z_j) = active }.
The set of range measurements from laser scans that correspond to active nodes; the corresponding sector of each range measurement must be *on*, $\alpha(r_i) = on$; the label for each range is either static or added.

Dynamic Map: $Z_{dynamic}$ = { r_i | r_i is a range point from the scan z_j, and l_i is the range label and $l_i \in \{removed, added\}$ }.
The set of ranges deriving from active or inactive nodes and are labeled either added or removed. Note that added points are also included in the dynamic map because they represent a change in the environment. The dynamic map is a representation the history of changes that have been detected.

The combination of the active and dynamic maps, the set of points derived from the static parts of the environment can be determined by computing the set difference between the active and dynamic maps. The active map represents the most current state of the environment as well as retains the parts of the environment that have not changed. The dynamic map contains a representative sample of laser range points from parts of the environment that have changed over time.

4.4 Discussion

There have been a number of successful approaches to the SLAM problem in static environments and environments with high-dynamic objects. However, the problem of mapping environments with low-dynamic objects has not received as much attention. This is, in part due to the difficulties of detecting and/or tracking low-dynamic objects, as well as operating a mobile robot for long-periods (eg., days, weeks, or months). In this chapter we presented our problem formulation for mobile robot long-term mapping in dynamic environments. Three core components of the problem were outlined including the dynamic environment model, the general probabilistic problem formulation of concurrent change detection and trajectory estimation, and the assumptions. Additionally, we proposed a novel model called the Dynamic Pose Graph used to compute and store an accurate map of the environment at any point in time, while capturing the environment dynamics. The

following chapter presents Dynamic Pose Graph SLAM, our method to enable a mobile robot to remain localized and detect changes while operating in a dynamic environment.

Chapter 5

Dynamic Pose Graph SLAM

A mobile robot operating in a dynamic environment over long periods must be able to continuously localize and maintain an up-to-date map as the map changes with the passage of time. The robot makes a number of passes through the environment, navigating part or all of the environment during each pass, and returning to a known home location. This chapter presents the Dynamic Pose Graph SLAM method to enable a mobile robot to maintain an up-to-date and accurate map, while operating in an environment composed of static and slow-dynamic environments (see Chapter 4). To equip a mobile robot with the ability to have sustained long-term persistent operation in dynamic environments, existing approaches that deal with transient, high-dynamic objects can be integrated with novel methods presented in this thesis that address slow-dynamic and static objects.

Developing a method for a robot to update and repair its representation is important to addressing the life-long SLAM problem. Also, it is important to be able to detect differences or changes, particularly in security applications such as ship hull inspection and airports. In addition, the DPG-SLAM method is used to investigate the efficacy of state-of-the-art pose graph SLAM methods, such as incremental smoothing and mapping (iSAM) [36] used in this thesis, applied to low-dynamic environments.

To develop the DPG-SLAM method a number of challenges must be addressed. For example, if the covariance of the robot's pose estimate becomes too large, then the robot is unsure of its current position. Consequently, the robot is unable to reliably detect change and update its map. At the same time, if the robot believes its position is fairly accurate and the environment has changed since the previous visit, then the robot is not able to reliably determine its relative position because the previous and the current scans of the area are different. Thus, the robot frequently asks the question *Am I lost, or has the world changed*? Another challenge is to address the size of the pose graph as it grows as a function of the number of poses or how much the robot travels rather than the size of the environment.

This chapter describes the DPG-SLAM method for continuous long-term operation in low-dynamic environments. Recall from Chapter 1 four long-term mapping goals were detailed. The first goal is to enable the robot to continuously incorporate new information as it navigates. The second goal is to create a representation the incorporates the history of the map over time. The third goal is to be able to detect and react to changes online. Finally, the fourth goal is to address the problem of tractability because the DPG grows as the robot navigates. These goals are addressed in this chapter by our DPG-SLAM method. The core algorithms of DPG-SLAM applied to the DPG model are presented. Finally, we detail the results of this method as it applies to the 4-pass CSAIL Reading Room example

described in 4.

5.1 DPG-SLAM Overview

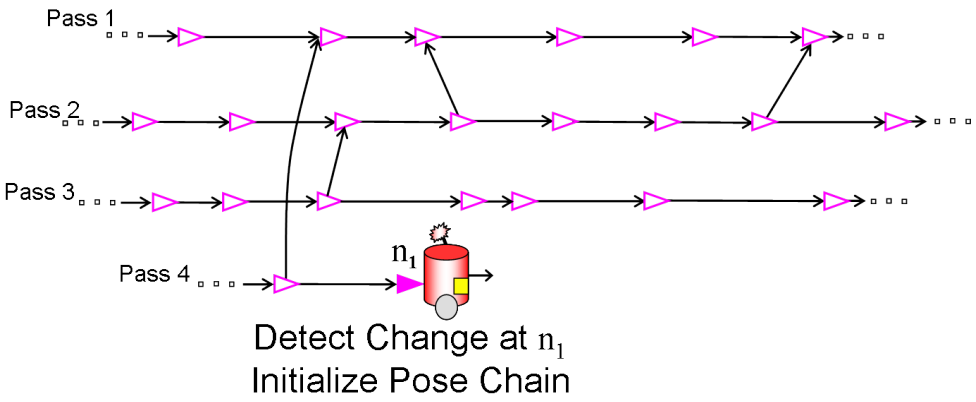
The Dynamic Pose Graph (DPG) allows us to use existing pose-graph methods for trajectory optimization, with our novel algorithms to address changes that occur in the environment. In this work, the robot is continuously updating its trajectory, the DPG, and the active and dynamic maps while the environment changes over time. This continuous process is detailed as a part of the Dynamic Pose Graph SLAM (DPG-SLAM) method. We assume the robot is equipped with a laser range finder and odometry. A summary of the key DPG-SLAM steps is outlined below.

DPG-SLAM Steps:

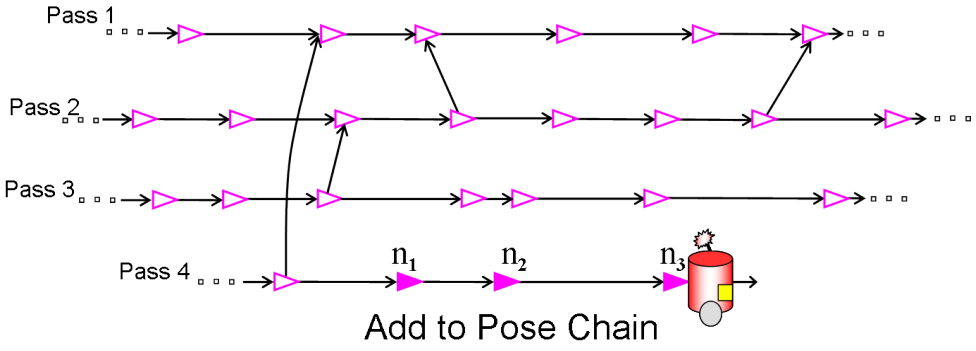
1. Pose Graph SLAM
2. Initialize pose chain
3. Close pose chain
4. Compute active submap
5. Detect and label changes
6. Update active and dynamic maps
7. Remove nodes and constraints
8. Repeat

In general, if there are no changes, the robot executes pose graph SLAM. Once the robot suspects a change, it initializes a *pose chain*. A pose chain is a sequence of poses in the DPG as seen in Figure 5-1. Recall that odometry errors along a sequence of poses grows without bound; only inserting a loop constraint and optimizing the pose graph can significantly reduces the pose error. After the pose chain is initialized, subsequent poses are added to the pose chain until a loop constraint is successfully inserted in to the DPG (see Figure 5-1(b) and Figure 5-1(c)). At this point, pose graph optimization using iSAM is performed, yielding the best pose estimates given the constraints. After optimization the pose chain is ”closed”, and the nodes along the chain are used to update the DPG. Intuitively, the pose chain is closed once the robot revisits a familiar part of the environment.

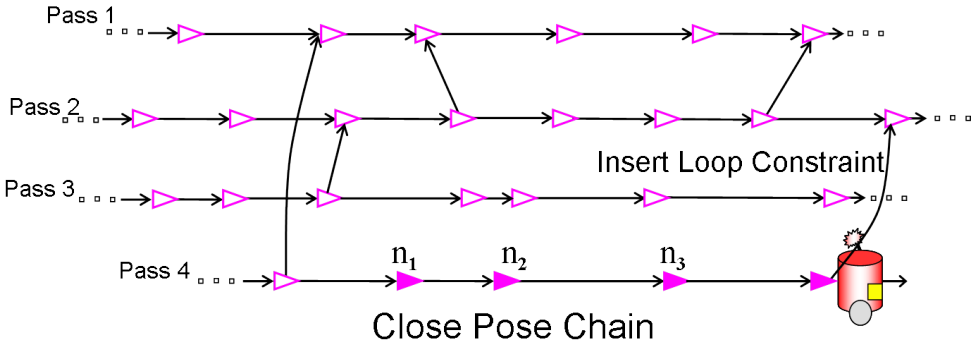
The general DPG-SLAM procedure is given in Algorithm 1. Line 5 refers to executing pose graph SLAM, the first step of DPG-SLAM. Initializing and updating the pose chain, Step 2, is shown on Lines 7 - 13. Steps 3 and 4 are implemented on Lines 17-18. Finally, the process of updating the DPG, Steps 3-6, is given on Lines 19 - 33. The algorithms unique to DPG-SLAM, shown in all caps, are detailed in the following sections.



(a)



(b)



(c)

Figure 5-1: Example of the robot executing the DPG-SLAM steps until the pose chain is closed. (a) shows that the robot suspects a change at n_1. Then nodes are added to the pose chain in (b). Finally, the pose chain, consisting of nodes n_1, n_2, n_3, and n_4, is closed when a loop constraint is inserted from the final node on the pose chain. Then the DPG is optimized to update the trajectory estimates.

Algorithm 1 DPG-SLAM(Graph $graph$, Node n_{start}) *returns* a Dynamic Pose Graph.

```
1: dpg ← initialize_dpg(graph, n_start)
2: while true do
3:    poseChain ← {}
4:    while robot not at home do
5:       pose_graph_slam(dpg)
6:       n_curr ← get_current_pose(dpg)
7:       if is_empty(poseChain) then
8:          ⟨submapNodes, submapPoints⟩ ← COMPUTE-ACTIVE-SUBMAP (dpg, n_curr)
9:          unmatchedPoints ← GET-UNMATCHED-POINTS(submapPoints, n_curr)
10:         changeScore ← COMPUTE-CHANGE-SCORE(unmatchedPoints, n_curr)
11:         if changeScore > changeScoreThreshold then
12:            poseChain ← poseChain ∪ n_curr
13:         end if
14:      else
15:         if NOT(is_empty(poseChain)) then
16:            poseChain ← poseChain ∪ n_curr
17:            if added_loop_constraint(dpg, n_curr) then
18:               optimize_pose_graph(dpg)
19:               changeNodes ← {}
20:               dynamicMapPoints ← {}
21:               for all Node n_i ∈ poseChain do
22:                  ⟨submapNodes, submapPoints⟩ ← COMPUTE-ACTIVE-SUBMAP (dpg,
                     n_i)
23:                  unmatchedPoints ← GET-UNMATCHED-POINTS(submapPoints, n_i)
24:                  changeScore ← COMPUTE-CHANGE-SCORE(unmatchedPoints, n_i)
25:                  if changeScore > changeScoreThreshold then
26:                     node_type(n_i) ← Change Node
27:                     dynamicMapPoints ← LABEL-POINTS(n_i, get_grid(n_i),
                        submapNodes)
28:                  end if
29:               end for
30:               inactiveNodes ← UPDATE-MAPS(dpg, get_removed_points(dynamicMapPoints))
31:               REMOVE-INACTIVE-NODES(inactiveNodes)
32:               poseChain ← {}
33:            end if
34:         end if
35:      end if
36:   end while
37: end while
```

5.2 Compute Active Submap

The set of scans from the pose chain form a submap representing the most recent state of the area of the environment covered by the poses on the pose chain. To update the active and dynamic maps, the pose chain submap points is compared with the *active submap*

from the same part of the environment. An active submap is a set of active points (ranges from "on" sectors) in nodes from previous passes that were added to the DPG before the current pose chain's pass. The previous nodes are referred to as active submap nodes. The fields-of-view (FOVs) of each submap node intersect with the FOV of the pose chain nodes. Figure 5-2 depicts an example of a pose chain and its points, as well as candidate nodes from previous passes that can be combined to create an active submap.

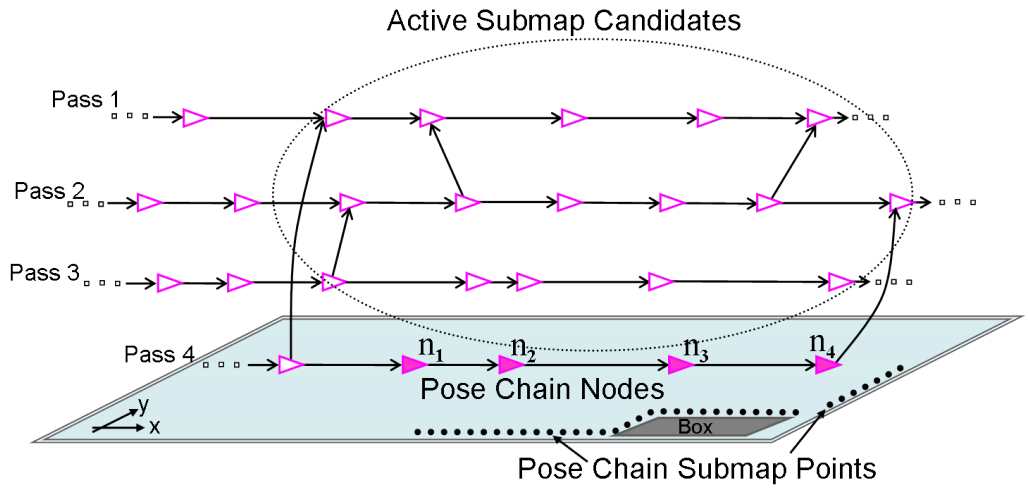


Figure 5-2: Example of a pose chain and a pose chain submap. A large set of potential active submap nodes are shown from previous passes.

An outline of the steps to find the active submap is given below.

Active Submap Steps:

1. Select one node from the pose chain n_{curr}
2. Select a subset of candidate active map nodes that are spatially close to n_{curr}
3. Create n_{curr} occupancy grid with occupied, free, and unknown cells
4. Compute coverage of n_{curr} with active map nodes by marking occupancy grid cells of n_{curr} according to the rules in Table 5.1.

The robot operates for long periods repeatedly revisiting areas and updating its map. When the robot re-visits part of the environment there may be numerous DPG nodes from several previous passes that represent the area. As a consequence, most of the nodes will contain redundant active map points. For example if there are 100 nodes from previous passes with FOVs that intersect the FOV of the current node, then it is likely that it is not necessary to use all 100 nodes to create an active submap. In the best case, our method would select the minimal set of nodes with FOVs that cover the FOV of the current node. However identifying this perfect subset can be computationally expensive. In our approach we find the first subset of nodes that intersect all or a large part (e.g. a coverage threshold of 90%) of the FOV of the current node.

Table 5.1: Rules for marking grid cells covered. If the cells were unknown, or are currently unknown, then we do not have enough information to state if the cell is covered, thus we mark it N/A.

active submap noden_i	current node n_{curr}	Label
free	free	mark as covered
occupied	free	mark as covered
unknown	free	N/A
free	occupied	mark as covered
occupied	occupied	mark as covered
unknown	occupied	N/A
free	unknown	N/A
occupied	unknown	N/A
unknown	unknown	N/A

Our approach to finding the active submap is to find the active submap nodes for one pose-chain node, n_{curr}, at a time. Determining the set of active submap nodes is similar to the problem of finding loop closing candidates in pose graph SLAM [12, 30, 35, 57]. As stated we are interested in finding a subset of nodes in the DPG where the union of the FOVs of the nodes covers completely or partially n_{curr}'s FOV. In contrast, complete FOV coverage is not necessarily a requirement for loop closing candidates. The set of scans from the submap nodes are used to determine the active submap points which, in turn, are used to detect changes. More specifically, if the set of scans from each submap node is { z_1, z_2, ..., z_n }, then $submapPoints$ = { r_i | $r_i \in$ { z_1, z_2, ..., z_n } AND r_i intersects the FOV(n) }, then the active submap points of the points in the submap nodes scans.

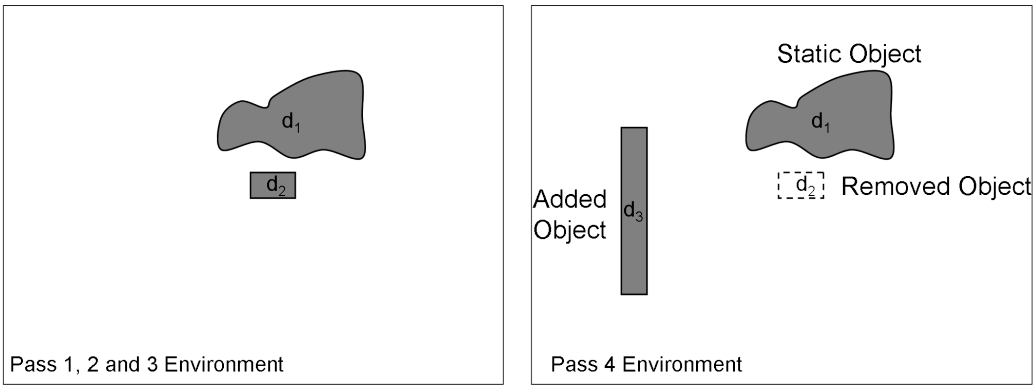
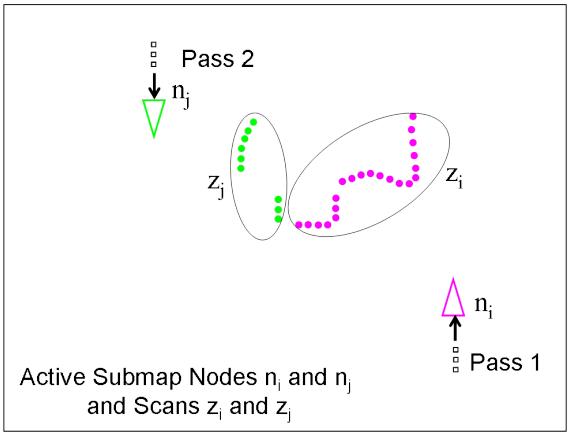
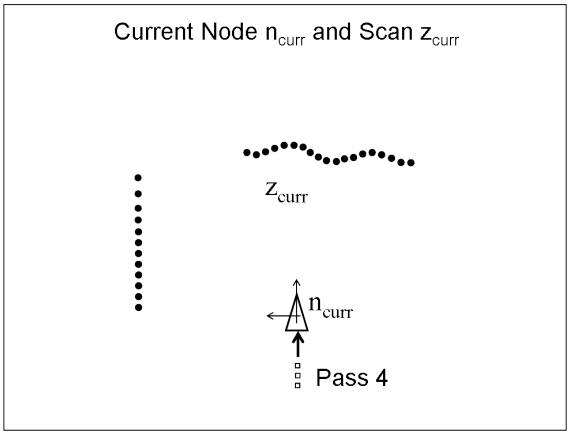


Figure 5-3: Example changes in low-dynamic environment during passes 1-3 and pass 4. There is no change during passes 1-3, as shown in Figure (a). After pass 3, a change occurs and the low-dynamic object d_2 is removed and d_3 is added as shown in Figure (b).

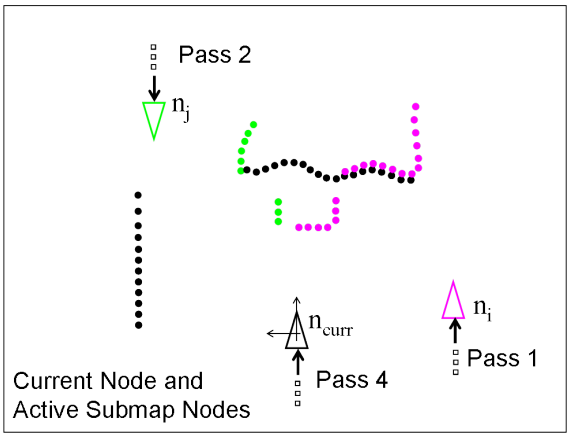
To determine the overlap or intersection of the FOV of node n_{curr} an occupancy grid for n_{curr} is constructed with cells that are defined as free, occupied, and unknown (Line 2). The unknown cells are a result of an obstruction in the robot's view. The grid is constructed temporarily for this step of the DPG-SLAM process. The scans of the submap nodes are used to approximate the amount of coverage by counting the cells marked as "covered".



(a)



(b)



(c)

Figure 5-4: Nodes and their laser range scans from the passes through the environment in Figure 5-3.

More specifically the amount of coverage of node n's FOV is determined by first creating an occupancy grid for n_{curr}. Then each of the submap nodes are projected into the reference frame of n_{curr} (Line 8). Next the cells of the occupancy grid are marked covered if and only if the points of the submap nodes cover the cells according to Table 5.1. Figure 5-3 illustrates an example environment during pass 1 and 2 (Figure 5-3(a)), and during pass 3 (Figure 5-3(b)). Scans of two candidate submap nodes, n_i and n_j from pass 1 and pass 2 respectively, are shown in Figure 5-4. In addition, the scan for the current node at pass 3 is shown in Figure 5-4(a). An example of the grid for n_{curr} is given in Figure 5-5(a). Then the active points from candidates nodes n_i and n_j are projected onto the grid of n_{curr} in Figure 5-5(b), depicting an occupancy grid based on n_i and n_j. The covered free and occupied cells of n_{curr} are marked with X's in Figure 5-5(c). Finally, the example depicts the active submap nodes and the active submap points of the current node in Figure 5-5(d).

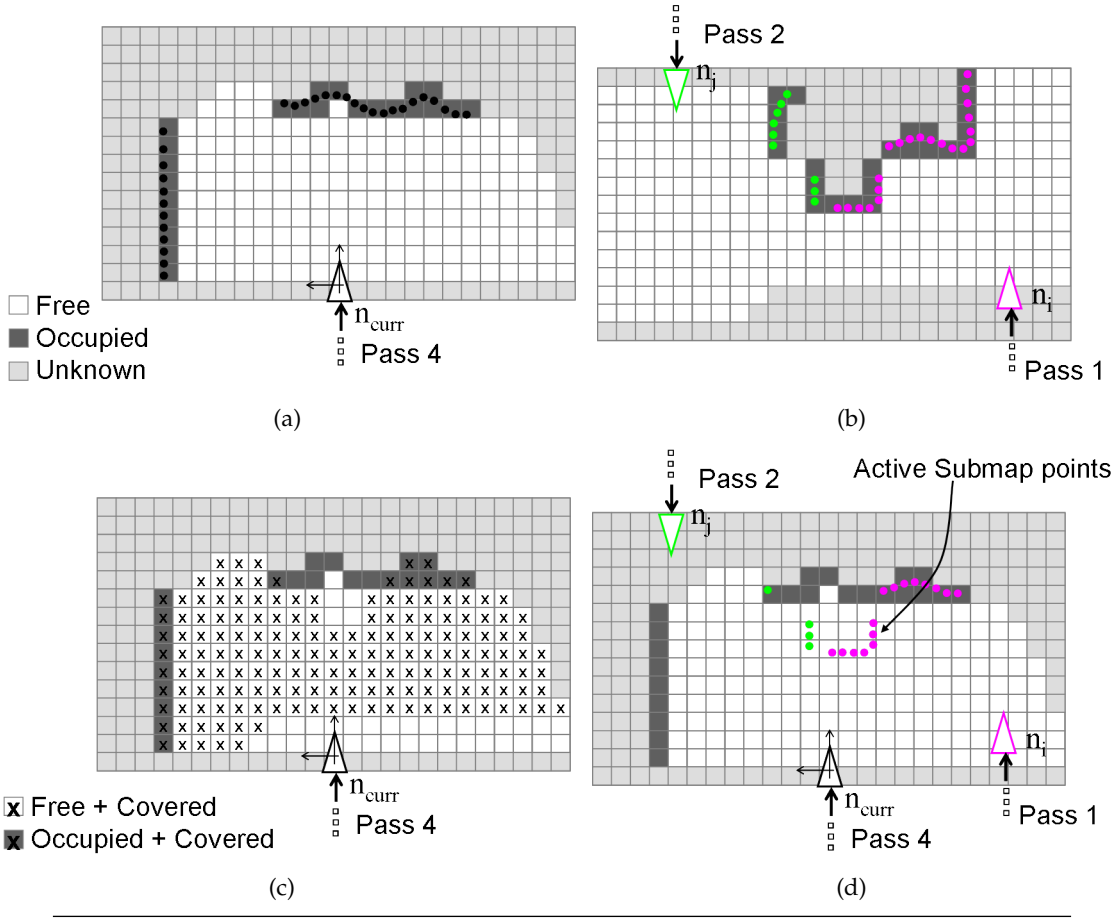


Figure 5-5: Illustration of the procedure to determine the coverage of n_{curr}'s FOV with candidate submap nodes n_i and n_j. The environment and the range scans of each node are shown in Figure 5-3 and Figure 5-4.

The COMPUTE-ACTIVE-SUBMAP algorithm given in Algorithm 2 takes as input a node n_{curr} from the current pose chain and the Dynamic Pose Graph dpg and returns the active submap nodes and active submap points. A pre-specified percentage of the current node's FOV (defined by the $grid$) must be covered in order to have sufficient points for change detection. The pre-specified amount is defined by MAX_COVERAGE. In addition,

candidate submap nodes are initially selected based on proximity to n_{curr}, but must also pass an uncertainty test. This test is performed as a part of the get_overlapping_nodes() function on Line 1 and is described below.

Algorithm 2 COMPUTE-ACTIVE-SUBMAP(Dynamic Pose Graph dpg, Node n_{curr}) *returns* a set of nodes with laser ranges that overlap the current node, and returns the submap which is the set of points that over lap the current node.

1: *candidateNodes* ← get_overlapping_nodes(dpg)
2: $grid$ ← create_grid(n_{curr})
3: *coveredCells* ← initialize_boolean_grid()
4: *totalCoveredCells* ← get_covered_cells(*grid*)
5: *submapNodes* ← { }
6: *candidateCoveredCellsCounter* ← 0
7: **for all** Node $n_i \in$ *candidateNodes* **do**
8: $set\_pose(n_i^{'}) \leftarrow \ominus x_{curr} \oplus x_i$
9: $set\_ranges(n_i^{'}) \leftarrow \ominus x_{curr} \oplus z_i$
10: **for all** ranges $r_j \in z_i'$ **do**
11: **if** candidateCoveredCellsCounter/totalCoveredCells ¿ $MAX\_COVERAGE$ **then**
12: break
13: **else**
14: *rayTraceCells* ← ray_trace($n_i^{'}$, r_j, *grid*)
15: **for all** Cell $c_k \in rayTraceCells$ **do**
16: $row \leftarrow$ get_row(c_k)
17: $col \leftarrow$ get_col(c_k)
18: **if** ($coveredCells$[row][col] == false) AND (*grid*[row][col] == free OR occupied) **then**
19: *submapNodes* ← *submapNodes* $\cup$ n_i
20: *candidateCoveredCellsCounter* ← *candidateCoveredCellsCounter* + 1
21: *submapPoints* ← *submapPoints* $\cup$ r_j
22: $coveredCells$[row][col] ← true
23: **end if**
24: **end for**
25: **end if**
26: **end for**
27: **end for**
28: return *submapNodes*, *submapPoints*

5.2.1 Node Uncertainty Test

As the robot navigates it re-visits areas and updates its map. Each time the robot re-visits a specific area changes may have occurred and the robot needs to updates its active and dynamic maps. However, the uncertainty in the robot poses (current pose and poses from submap nodes) at that location affect its ability to determine if a change occurred. In order to accurately detect and identify changes the robot must be able to answer the question "Am I lost, or has the world changed?" The first part of the questions refers to the relative covariance between the current node and nodes in the submap. That is the robot may be

very uncertain of its position relative to the submap. As a result it is lost and therefore cannot accurately discern whether or not a change have occurred. The second part of the question implies that the robot is fairly certain about pose estimate and the estimate of the poses in the submap. However, small or dramatic changes have occurred since the last time the robot visited the area, and thus, the result of the scan matcher would be unreliable.

For example, in Figure 5-6 the low-dynamic object d_1 is removed after pass 1. At pass 2, node n_i sufficiently overlaps n_j however there is large relative uncertainty between the two poses, as a result n_i is not a reliable submap node for change detection. Figure 5-6(b) depicts a much smaller area of uncertainty between the two nodes. In general, if the uncertainty in the predicted transformation between two nodes is above a certain threshold then the change detection algorithm will not be able to differentiate between change or a misalignment (from matching scans) caused by inaccurate prediction.

This section describes the uncertainty test that candidate active submap nodes must pass as a part of the get_overlapping_nodes() function in Algorithm 2 Line 1.

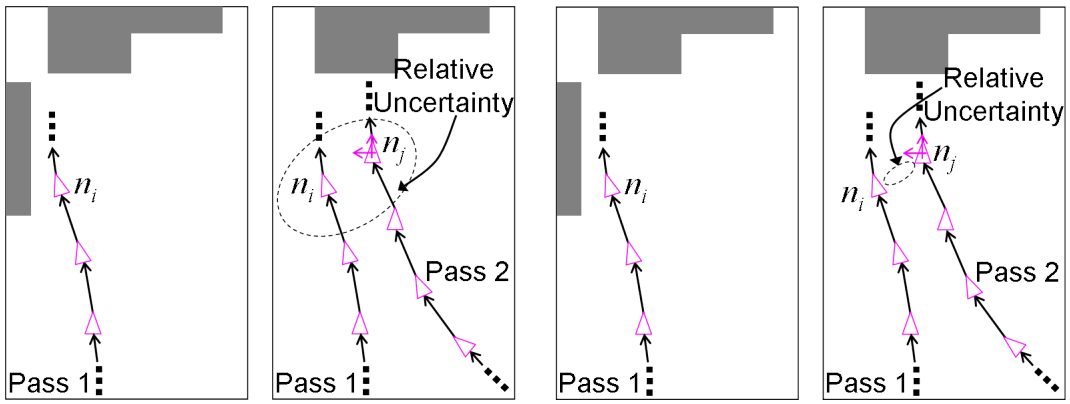


Figure 5-6: Illustration of relative pose uncertainty between a node and a candidate submap node. A low-dynamic object is removed after pass 1. The ellipsoids represent the area in which the two nodes may reside. The larger ellipse in Figure (a) depicts high uncertainty between the two nodes n_i and n_j. The relative uncertainty between the two nodes depicted in Figure (a) is much smaller and n_i can be included in the local submap.

Given two nodes n_i and n_j with means $\mathbf{x_i}$ and $\mathbf{x_j}$ with estimated poses $\hat{\mathbf{x}}_\mathbf{i}$ and $\hat{\mathbf{x}}_\mathbf{j}$ then we compute the error ϵ as follows: $\hat{\mathbf{x}}_\mathbf{i} = \mathbf{x_i} + \omega_i$, $\hat{\mathbf{x}}_\mathbf{j} = \mathbf{x_j} + \omega_j$, where ω_i and ω_j are noise. Let,

$\Delta = \mathbf{x_j} - \mathbf{x_i}$, and $\hat{\Delta} = \hat{\mathbf{x}}_\mathbf{j} - \hat{\mathbf{x}}_\mathbf{i}$, where $\hat{\Delta}$ is the prediction of the relative transform between the two nodes. Then the error ϵ is defined as follows:

$\epsilon = \hat{\Delta} - \Delta = w_j - w_i = w_{ji}$, where w_{ji} is a random variable derived from a Normal distribution, $Normal(0, C)$.

The prediction of the relative transform (or constraint), $T'_{i,j}$, between any two nodes n_i and n_j, is computed from the pose estimates currently in the DPG. The aim of the uncertainty test is to determine the likelihood that the error of the prediction, denoted ϵ is bounded by a given threshold m. We write $\mathrm{P}(|\epsilon| > m) = \mathrm{P}(\epsilon^2 > m^2) = \mathrm{P}(\epsilon^T C^{-1} \epsilon > m^T C^{-1} m)$ and apply a χ^2 test, where

$\epsilon^T C^{-1} \epsilon$ follows a χ^2 distribution. To compute the probability the prediction covariance, C, is required. We compute a conservative estimate of the prediction covariance, C, using

the Dijkstra Projection described in [12]. Then our test is if $m^T C^{-1} m > b$ then reject the node, otherwise accept the node.

The goal of the node uncertainty test is to compute the likelihood that the prediction of the relative transform between a candidate active submap node and the current node, n_{curr}, is greater than a given distance from the mean. To achieve this, we compute the distance to the mean of the prediction of the relative transform from a set of threshold vectors. Six error pre-defined threshold vectors, denoted $m_1, ..., m_6$, are given to represent the maximum tolerable error from the mean along each axis,

$$m_1 = \begin{bmatrix} -\sigma_x \\ 0 \\ 0 \end{bmatrix}, m_2 = \begin{bmatrix} +\sigma_x \\ 0 \\ 0 \end{bmatrix}, m_3 = \begin{bmatrix} 0 \\ -\sigma_y \\ 0 \end{bmatrix}, m_4 = \begin{bmatrix} 0 \\ +\sigma_y \\ 0 \end{bmatrix}, m_5 = \begin{bmatrix} 0 \\ 0 \\ -\sigma_\theta \end{bmatrix}, m_6 = \begin{bmatrix} 0 \\ 0 \\ +\sigma_\theta \end{bmatrix},$$

where σ is a given distance for the threshold.

Finally if the node n_i passes the uncertainty test, then the candidate node n_i is included in the set of active submap nodes.

The COMPUTE-ACTIVE-SUBMAP algorithm finds a set of "nearby" nodes and points from previous passes that are later used to detect changes. The algorithm has some limitation that result in old outdated ranges being include in the active map. The first limitation is that there may be a few range points of the current node that are not covered by the active submap and thus not used in change detection. A second limitation is that the active map nodes may only intersect a small portion of the FOV of the current node. As a result, the coverage threshold is not exceeded and change detection is not done and the submap of the current area is not updated. However, as the robot continues to navigate and cover the environment change detection will likely be performed and the submap of this particular area of the environment will be updated.

5.3 Detect and Label Changes

Our goal is to provide the robot with the capability to determine if a change occurred between the current pass through the environment and previous passes. A robot detects a change when it revisits a location from a previous pass and compares its current scan with the active submap. There are three main steps to detect and label change. These steps are shown below and refer to Lines 23 - 27 in Algorithm 1.

Detect and Label Changes Steps:

1. Get unmatched points
2. Detect changes
3. If changes detected then label points

The goal is to compute the probability that there is change given the current scan and active map points, $P(c_{curr}|z_{curr}, \eta)$, where c_{curr} is the event that the current scan represents a change(s) from the previous submap, and η is the set of active map points that do not match (i.e., approximately coincide with) any points on the current scan z_{curr}. To determine if there is a change we approximate the probability by a binary function indicating if there are changes in the environment or not (see Equation 5.1). This function determines

if the amount of change is greater than a given threshold δ, and is related to the general problem formulation P(C —X, Z) given in 4.2.

$$P(c_{curr}|z_{curr}, \eta) = \begin{cases} 1, & \textit{if score}(z_{curr}, \eta) > \delta \\ 0, & \textit{otherwise} \end{cases} \tag{5.1}$$

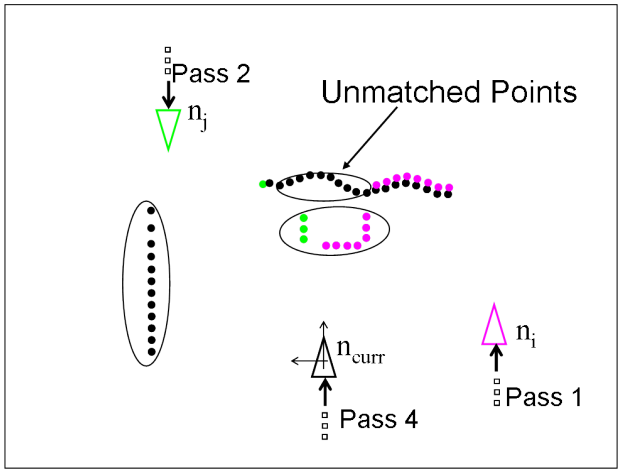
To look for changes we need the set of unmatched points from the active submap, denoted η, and the points in the current node's scan z_{curr}. The first step is to collect the set of unmatched points that are within the FOV . The set of unmatched points refers to the points in η and the points in z_{curr} that do not match any active map points. An example of the set of unmatched points from Figure 5-4 is shown in Figure 5-7. To determine the unmatched points, we propose two possible methods. The first method is to use scan matching to register the z_{curr} and the active submap points. Points not used to register the two scans are unmatched points. The second method is to use n_{curr}'s grid created in COMPUTE-ACTIVE-SUBMAP. We keep the points from n_{curr} and the points from the active map nodes in their corresponding grid cells. Then the cells that have points from only the active map nodes or only the current node are considered unmatched points (see Figure 5-7(b)). This process is similar to the occupancy grid map technique given in [11] to differentiate between static and dynamic.

The function that computes the $score$ to detect change is called COMPUTE-CHANGE-SCORE and is given in Algorithm 3. The function computes the amount of change between the current scan and the active submap. The current node n_{curr} and the unmatched points are inputs to the function. We then divide the node's angular sensor range into equal sized segments. These are temporary and are not the same as a sectors defined in Chapter 4. For example, for a 180^o sensor range it might be divided up into 2^o segments. Then each of the unmatched points are assigned to a segment based on their angle relative to the current node's pose, x_i (see Figure 5-8). Finally, the number of segments covered by the unmatched points represents an approximation on the percentage of change.

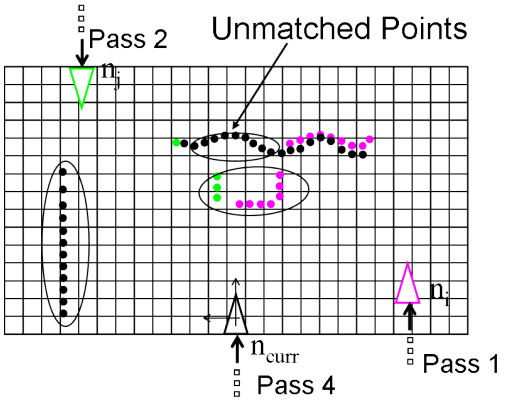
Algorithm 3 COMPUTE-CHANGE-SCORE(Node n_{curr}, Points $unmatchedPoints$) *returns* a score for percent of change.

```
changeCounter ← 0
coveredSegments ← segment_sensor_range(angleResolution)
for all p_i ∈ unmatchedPonts do
    segment ← get_segment(coveredSegments, get_angle(p_i))
    if segment == false then
        coveredSegments(segment) ← true
        changeCounter ← changeCounter + 1
    end if
end for
percentChange ← changeCounter/ get_total_segments(coveredSegments)
return percentChange
```

If the percentage of segments that include unmatched points is greater than δ from Equation 5.1, then the points are labeled *static*, *removed*, or *added*, ie. $l_k \in \{static, removed, added\}$. Static points are the set of points that are derived from parts of the environment where the robot has previously visited, but its view was obstructed. Thus, the are was unknown and is currently known. Added points are new points from parts of the environ-



(a)



(b)

Figure 5-7: Example of the unmatched points (points that are circled in the images). (a) can be achieved by registering the two set of points with scan matching and (b) illustrates the grid approach.

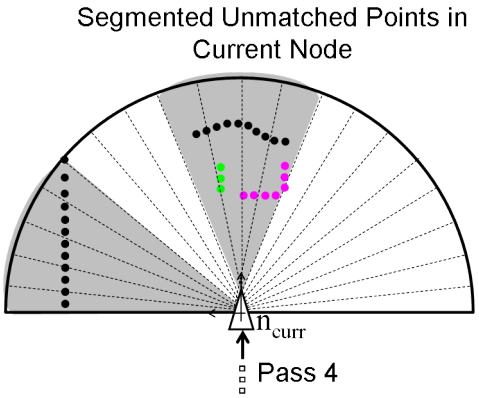


Figure 5-8: Example of the unmatched points in their respective segments for change detection. There are 20 segments and unmatched points occupy 9 of them. Thus, the percentage of change (the change score) is approximately 45%, and if the percentage of change is greater than δ then n_{curr} is a change node and the unmatched points are then labeled added or removed.

ment where the robot has not visited or parts of the environment that were previously free space and are now occupied. Lastly, the removed points are points that were previously in the map and are no longer in the map.

The labeling of each point affect the active map and the dynamic map. Recall that the active map is the set of all static and added points, and the dynamic map is the set of all added and removed points. The labels are a reflection of the types of change that occur in the environment. That is, they represent change from low-dynamic objects that are added, moved, or removed, noting that a moved object is defined by the combination of added and removed.

Algorithm 4 LABEL-POINTS(Occupancy Grid $currentGrid$, Occupancy Grid $submapGrid$) *returns* a set of points that are labeled added or removed.

```
1: dynamicMapPoints ← {}
2: for row = 1 to num_rows(grid) do
3:   for col = 1 to num_cols(grid) do
4:     if (currentGrid[row][col] == occupied) AND (submapGrid[row][col] == free)
       then
5:       currentCellPoints ← get_points(currentGrid[row][col])
6:       for all p_i ∈ currentCellPoints do
7:         label(p_i) ← added
8:         dynamicMapPoints ← dynamicMapPoints ∪ p_i
9:       end for
10:    else if (currentGrid[row][col] == free) AND (submapGrid[row][col] == occupied)
       then
11:      submapCellPoints ← get_points(submapGrid[row][col])
12:      for all p_j ∈ submapCellPoints do
13:        label(p_j) ← removed point
14:        dynamicMapPoints ← dynamicMapPoints ∪ p_j
15:      end for
16:    end if
17:  end for
18: end for
19: return dynamicMapPoints
```

The procedure for labeling points is described in Algorithm 19, LABEL-POINTS. The occupancy grid for the current node, $currentGrid$ (Figure 5-5(a)), and the occupancy grid for the active submap nodes $submapGrid$ (Figure 5-5(b)) are the inputs to update the dynamic map, ($dynamicMapPoints$). These two grids are overlaid and the state of the cells is compared from before to after, where the $submapGrid$ is the state before and the $currentGrid$ is the state after. The rules are shown in Table 5.2. In this method it is only necessary to traverse the grid cells from each grid that refer to the unmatched points, although the LABEL-POINTS algorithm traverses over all the grid cells. Finally, the procedure returns the set of dynamic map points, where the added points are updated in the active map and the added and removed points are used to update the dynamic map. Figure 5-9 the added and removed points from the example in the previous section. Note that the two occupancy grids are small local grids relative to the current node, and are used temporarily for this procedure.

Table 5.2: Rules for labeling points.

submapGrid	*currentGrid*	Label
free	occupied	label the current points *added*
occupied	free	label the submap points *removed*
unknown	occupied	label the current points *static* (by default)

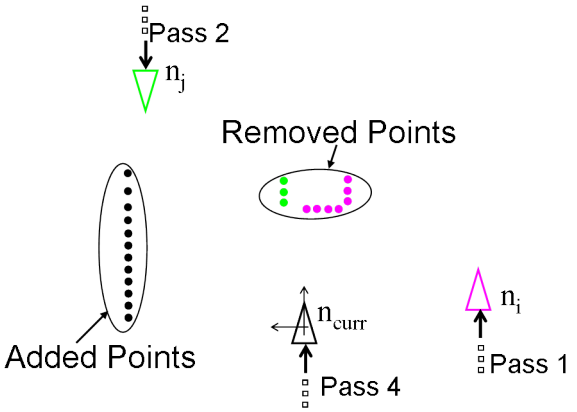


Figure 5-9: The added and removed points from the current node's scan z_{curr} and the active submap points. The labels for the points were determined by traversing the current grid in Figure 5-5(a) and the submap grid in Figure 5-5(d) and applying the rules in Table 5.2.

To maintain accurate and up-to-date active and dynamic maps, we need to detect changes and identify ranges the do not accurately reflect the current state of the environment. There are a few limitations of the DETECT-CHANGE and the LABEL-POINTS procedures. At times, there will be active submap points that have changed but are outside the FOV of the current node, and within the current node's sensor range. These points will not be detected, and thus, will remain in the active map. In addition, due to sensor noise and error in the pose estimates, there will be some points that are not included in the set of unmatched points. As a result, these points will not be labeled and will remain in the active map. In practice, these limitations are shown to minimally impact the accuracy of the active and dynamic maps.

5.4 Update Active and Dynamic Maps

To update the active and dynamic maps the labeled points from change detection and labeling are used. The dynamic map points are updated to include the added and removed points. The active map is updated by first including the added and static points from the current pose chain node n_{curr}. The removed points, assumed to be derived from a low-dynamic object that has been moved or removed, are used to update the active map by turning certain sectors of nodes off.

Recall that the active submap nodes are a subset of the nodes in the area that the robot is revisiting (see Figure 5-2). The collection of scans from submap nodes are a representative submap of this area. To update the active map all the ranges referring to the removed object should be discarded from the active map. More specifically, the removed points, and all

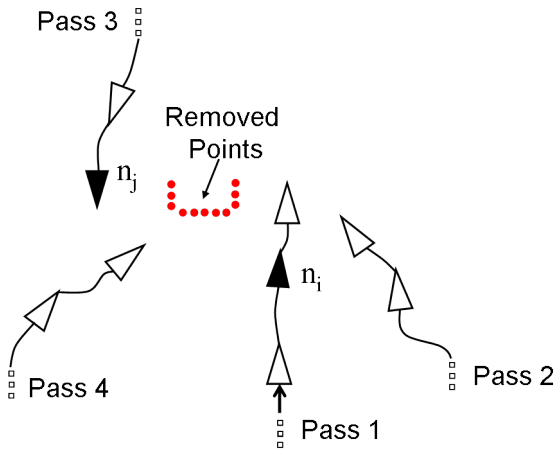


Figure 5-10: Example of several nodes in the same area, where all of the nodes have one or more sectors that intersect with the removed points. Two of the nodes n_i and n_j are the active submap nodes.

the points from the nodes in the area that are coincident with the removed points should be discarded from the active map. For example, in Figure 5-10 there are two active submap nodes, n_i and n_j, that contain ranges from an object that have been removed. There are also nodes that have ranges that coincide with the removed points, which should also be discarded.

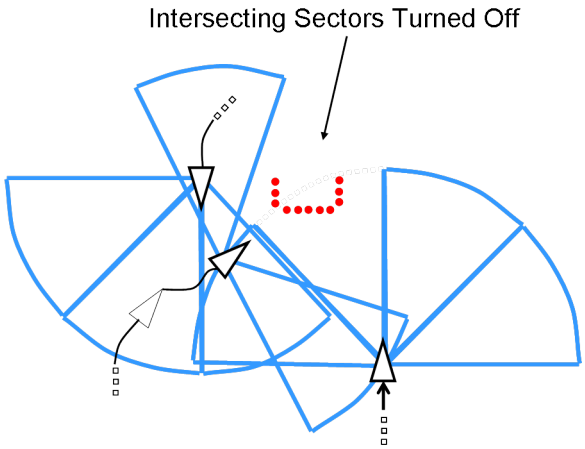


Figure 5-11: Illustration of sectors intersecting with the removed points. The sectors shown in bold lines are *on* and the remaining sectors are turned *off*.

The process of updating the active map given a set of range points is given in Algorithm 5. The algorithm iterates over each removed point and turns *off* the sectors of nodes that intersect with each removed point (see Figure5-11). These points in those sectors do not represent the most recent state of the area that the robot has revisited. If all the sectors of a node (or some given percentage of the number of sectors) are turned *off*, then the node is labeled *inactive* and its laser scan is completely discarded from the active map. In addition, recall that active points are the set of points from a node's "on" sectors. Similarly, inactive

points are points from the sectors of a node that are turned "off". Therefore the $activemap = activemap \bigcup staticpoints - inactivepoints$.

Algorithm 5 UPDATE-MAPS(DynamicPoseGraph dpg, Points $removedPoints$) *returns* a set of inactive nodes.

```
1: inactiveNodes ← {}
2: for all p_i ∈ removedPoints do
3:    nearbyNodes ← get_potential_overlapping_nodes(p_i, dpg)
4:    for all n_j ∈ nearbyNodes do
5:       if active(n_j) == true then
6:          if intersects(p_i, n_j) then
7:             sector ← get_sector(n_j, p_i)
8:             set_state(sector) ← off
9:             if percentage of on-sectors < percentOnThreshold then
10:               set_state(n_j) ← inactive
11:               inactiveNodes ← inactiveNodes ∪ n_j
12:            end if
13:         end if
14:      end if
15:   end for
16: end for
17: return inactiveNodes
```

The process of discovering added and removed points is subject to noise in the sensor measurements as well as noise in the pose estimates. Nodes with sectors that border the removed points, but do not overlap might not be turned off. These nodes may have points that should be removed. As a result, there may be some erroneous points included in the active map. These points contribute to an effect we call *ghosting* in the active map. In practice, however, we found that the amount of these erroneous points is significantly small compared to the number of correct active map points. In addition, points that are inserted into the dynamic map are inserted based on their location in the global reference frame at the time that the points are identified. Therefore, if the pose estimates of the nodes from which the points were collected ever change, the location of the points will not move.

5.5 Removing DPG Nodes and Constraints

One of the main goals of this thesis, outlined in Chapter 1, is to address the problem of computational tractability. The tractability problem arises because the pose graph grows as the robot continuously navigates through and re-visits areas of the environment. Continuously adding constraints to the DPG increases the computation time to compute the robot's trajectory, X, by pose graph optimization. We reduce the size of the DPG by removing inactive nodes from the graph. Also, we assume that the environment changes, otherwise nodes and constraints would not be removed from the DPG (in this case pose graph SLAM for static environments should be applied).

During the process of updating active and dynamic maps nodes with all sectors turned *off* become inactive. Inactive nodes do not contribute to the active or dynamic maps. They serve as place holders to keep the incoming and out going constraints of the inactive node a

part of the DPG. Our goal is to remove the inactive nodes and their incoming and outgoing constraints from the DPG. To achieve this there are two details to consider:

1. *The DPG must remain connected.* Nodes that represent an *articulation point* [34] must not be removed from the DPG. An articulation point in a graph is a node whose removal would cause a connected graph to be disconnected. In DPG-SLAM the constraints between the nodes in the graph are used as input to pose graph optimization. In pose graph optimization the DPG constraints and nodes represent a matrix, and thus, the problem becomes singular, i.e. under-constrained [28, 36].

2. *Restrict the length of the chain of nodes to be removed.* During the process of removing an inactive node, it is likely that additional nodes from the same pass as the inactive nodes, and that are sequentially connected to the inactive node are removed (see Figure5-12(c)). Removing nodes from the DPG affects the active map's coverage of the environment. It also affects the density of points that refer to static objects. Thus, we limit the number of nodes that can be removed at once.

3. *Remove singleton nodes.* While removing inactive nodes, it is possible for a node in the pose graph to be disconnected because all of its incoming and outgoing edges have been removed. At this point, the singleton node should also be removed from the DPG in order to ensure the graph remains connected.

Removing nodes modifies X because each node refers to a pose on the trajectory. Thus, a subset of nodes in X are maintained, $X^* \subseteq X$. Additionally, when nodes are removed their corresponding laser scans are removed too. The active and dynamic maps are created based on the nodes in the dynamic pose graph. Consequently, applying DPG-SLAM with and without node removal on the same set of data, yields different active and dynamic maps (see example in 5.6).

Our aim is to reduce the number of constraints and thus address the problem of tractability of pose graph optimization on a DPG that grows indefinitely. An outline of the steps to remove an inactive node is given below.

Remove Inactive Node Steps:

1. Traverse back each predecessor node and insert start constraint
2. Traverse forward to each succeeding node and insert end constraint
3. Create removal chain
4. Remove nodes and constraints along removal chain

Our method to remove nodes from the DPG differs from marginalizing out nodes in a pose graph [38, 42, 69]. Marginalization produces additional constraints in the pose graph based on the dependencies (incoming and outgoing constraints) of the node being removed. However, in our approach we limit the number of constraints to ensure our DPG is connected by inserting at most two constraints. The first constraint added (or used if one already exists) is from predecessor nodes along the same sequence as the inactive node. The second constraint added (or used if one already exists) is from a successor node

sequentially connected to the inactive node. The addition of constraints is shown in Figure 5-12(d) where a start constraint and an end constraint are inserted into the DPG. A start constraint represents a familiar part of the environment. That is, there is sufficient overlap between the two scans at the two nodes that the start constraint it connects. The same is true for an end constraint. Removing constraints may temporarily reduce the accuracy of the pose estimates, but as the robot continues to navigate and add new constraint the pose estimates will improve.

Figure 5-12 illustrates the steps of creating a removal chain and removing the nodes in the chain. In Figure 5-12(a) the node to be removed is n_i and filled in gray. The first step to remove an inactive node, shown in Figure 5-12(b), is to backtrack from n_i to its sequential predecessors that are in the same pass as n_i and are not connected to n_i via a loop constraint. While traversing back in the graph each sequential node is added to the set of *removal nodes*. To ensure the graph remains connected, an attempt to insert a loop constraint, called a *start constraint*, is done at each sequential predecessor node of n_i. The same process is done for the successor nodes until a loop constraint called an *end constraint* is inserted (see Figure 5-12(c)). Finally, if the length of the removal chain does not exceed the pre-specified maximum removal chain length, then the nodes and constraints along the removal chain are removed from the DPG, as shown in Figure 5-12(d).

The REMOVE-INACTIVE-NODES procedure is given in Algorithm 6. The main purpose is to reduce the size of DPG at regular intervals. Often times older inaccurate ranges are removed, and redundant ranges are removed. The input to REMOVE-INACTIVE-NODES is a set of inactive nodes generated from updating the active map and the DPG. To remove and inactive node, n_i a *removal chain* is constructed on Lines 2-4. A removal chain is a sequence of nodes from the same pass that are connected to n_i and are candidates for removal. Algorithm 7 details the process of constructing a removal chain. There are many different options to inserting the start and end constraints for a removal chain. In this work, the start constraint and the end constraint are inserted into maps that are more recent than the inactive node. That is, the start and end constraints connect with nodes in passes that come after the pass of the inactive node. Older passes will eventually be removed as the environment changes over time. REMOVE-INACTIVE-NODES returns the updated DPG with 0 or more nodes and edges removed.

In general the REMOVE-INACTIVE-NODES procedure performs as desired to address the computational tractability problem for large DPGs. However, the procedure does have some limitations. First, to remove an inactive node the REMOVE-INACTIVE-NODES procedure might require backtracking to the start of the pass of the inactive node (or similarly traversing forward to the end of the pass). If the removal chain length is not exceeded, then the nodes on the chain are removed and their information (ie. laser scans, etc) is lost. In our experiments we rarely encountered such scenarios where an entire pass is removed from the DPG. A second limitation is that not all inactive nodes will be removed if the criteria are not all met. Thus, the DPG will likely continue to include some inactive nodes. Another limitation is that nodes in the removal chain with several incoming and outgoing constraints might be removed. Removing these constraints affects the quality of the pose estimates. We argue that in the long-term mapping problem the pose estimates will improve as the robot continues to navigate and revisit various parts of the environment.

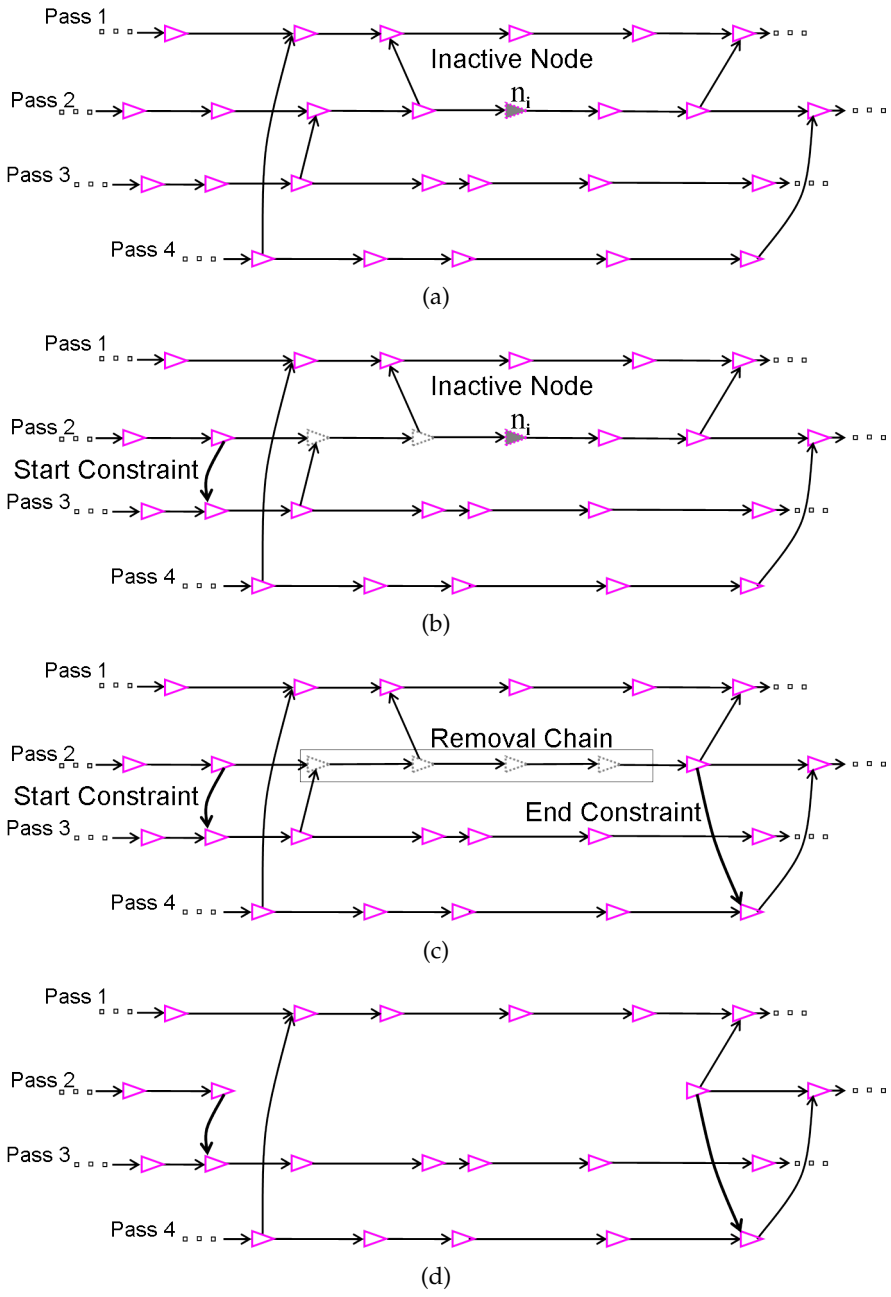


Figure 5-12: Example of removing an inactive node, n_i, from the DPG. A start constraint represents a familiar part of the environment. That is, there was sufficient overlap between the two scans at the two nodes that the start constraint it connects. The same is true for an end constraint.

Algorithm 6 REMOVE-INACTIVE-NODES(DynamicPoseGraph dpg, Nodes $inactiveNodes$) *returns* the dynamic pose graph with zero or more nodes and edges removed.

for all $n_i \in inactiveNodes$ **do**
  $\langle startSeqNode, startChainEdge \rangle \leftarrow$ GET-REMOVAL-CHAIN($usePredecessor$, n_i, dpg)
  if $startChainEdge \neq null$ **then**
    $\langle endSeqNode, endChainEdge \rangle \leftarrow$ GET-REMOVAL-CHAIN($useSuccessor$, n_i, dpg)
    if $startChainEdge \neq null$ AND $endChainEdge \neq null$ **then**
      insert_edge(dpg, $startChainEdge$)
      insert_edge(dpg, $endChainEdge$)
      if no articulated points on the removal chain **then**
        remove_node_sequence(dpg, $startSeqNode$, $endSeqNode$)
      end if
    end if
  end if
end for
return dpg

Algorithm 7 GET-REMOVAL-CHAIN(SearchDirection $searchDir$, Node n, DPG dpg) *returns* the sequential predecessor or successor node of node n (based on the search direction) and the added loop constraint edge.

$chainEdge \leftarrow null$
$chainCounter \leftarrow 0$
$seqNode \leftarrow$ get_sequential_node($searchDir$, n)
while $seqNode \neq null$ AND $chainCounter \leq maxRemovalLength$ **do**
  Edge $edge \leftarrow$ insert_constraint($seqNode$, dpg)
  if $edge \neq null$ **then**
    $remNode \leftarrow$ get_source($edge$)
    $chainEdge \leftarrow edge$
    break
  end if
  $chainCounter \leftarrow chainCounter + 1$
  $seqNode \leftarrow$ get_sequential_node($searchDir$, $seqNode$)
end while
return $seqNode$, $chainEdge$

5.6 DPG-SLAM Example

In this section we walk through an example of DPG-SLAM and a variant DPG-SLAM-NR, where no nodes or constraints are removed. The example is from 4 passes through the CSAIL Reading Room as described in Chapter 4. Nodes contain 5 sectors and change is detected when 20% or more of the environment has changed. The two algorithms illustrate the tradeoff between not removing nodes or constraints and removing nodes and constraints (DPG-SLAM). This section details the evolution of each map per pass and the dynamic maps as well, while Chapter 6summarizes the results applied to a number of different scenarios. To begin, Figure 5-13 presents a summary of images of the active map, DPG, and dynamic map generated by both of the algorithms.

5.6.1 DPG-SLAM-NR

The DPG-SLAM-NR active maps have a few ghosting points from older passes that are not in the DPG-SLAM active maps. It also shows that parts of the static areas (walls) are removed, due to sectors that have been turned off. These areas are seen in the upper left corner of Figure 5-13(d) of DPG-SLAM, where there are much less points than in the upper left corner of Figure 5-13(c) DPG-SLAM-NR. A summary of the total nodes and constraints after each pass is shown for DPG-SLAM-NR applied to the 4 CSAIL Reading Room data sets. The Table shows how much the number of change and inactive nodes increase after each pass.

Table 5.3: Totals of the node types and constraints in the DPG after each pass. The data is from 4 passes through the CSAIL Reading Room and DPG-SLAM-NR is applied.

#Passes	Nodes	Constraints	Change Nodes	Inactive Nodes
1	119	129	—	—
2	252	263	63	3
3	373	452	143	30
4	495	612	202	80

Figure 5-14 shows the number of removed and added points that are inserted into the dynamic map after each pass. The Figure clearly shows that there are some points in the dynamic map that derive from static parts of the Reading Room. This is an artifact of the DPG-SLAM method where the constant updating of the pose estimates can cause the map to shift slightly. As a result, part of a wall during one pass might be slightly moved in a later pass.

Figure 5-15 depicts images of the evolution of the map from pass 1. As changes are detected in later passes, some nodes from pass 1 become inactive and other nodes have some of their sectors turned off. As a result, it appears that the pass 1 map starts to fade with the increasing number of passes.We see a similar phenomenon with the maps from the other passes as well (see Figure 5-16 and Figure 5-17).

The images in Figure 5-16 and Figure 5-17 show the evolution of the maps referring to the second and third pass maps of the CSAIL Reading Room. It is clear that ranges from the dynamic object are removed over time. In addition, the number of inactive nodes and change nodes increase. For example in Figure 5-16(b) the regular nodes circled in the right corner become change nodes after pass 4 as shown in Figure 5-16(c). The fourth map

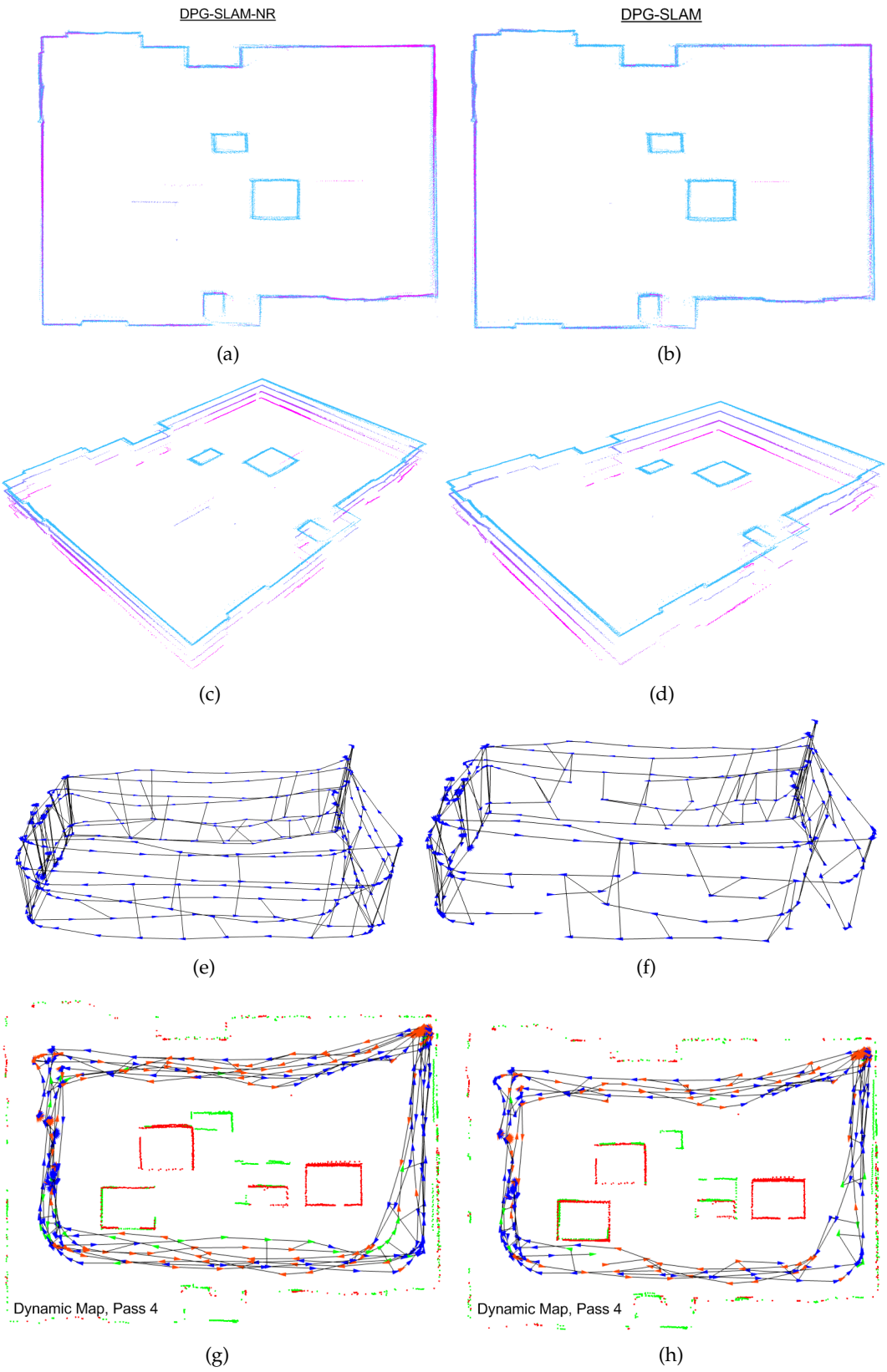
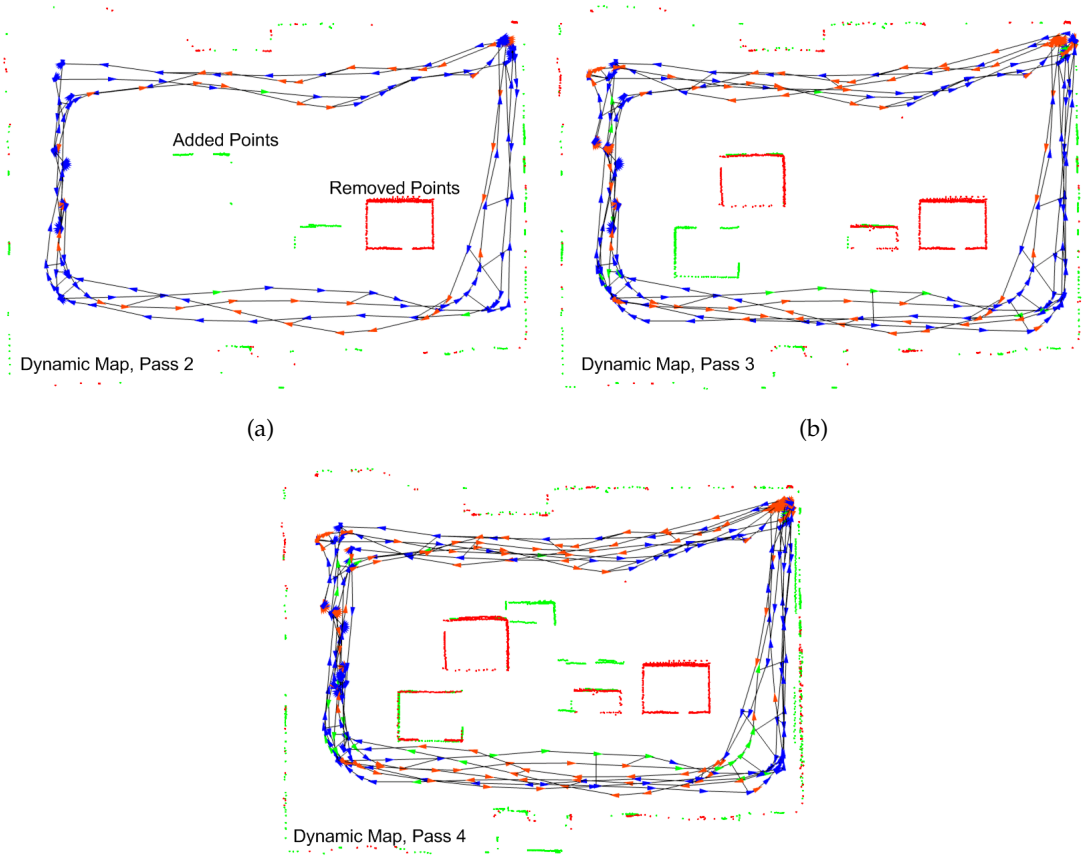


Figure 5-13: On the left are the active, DPG, and dynamic maps for the DPG-SLAM-NR algorithm and on the right are the same images for the DPG-SLAM algorithm. The points are color-coded from each pass, where magenta is the oldest pass and cyan is the most recent pass. In Figure (g) and Figure (h), the change nodes are colored in orange, the inactive nodes are colored in green, the added points are colored in red, and the removed points are colored in red.



(a) (b)

(c)

Figure 5-14: Images of the dynamic map after the second third and fourth pass through the Reading Room, where DPG-SLAM-NR was applied.

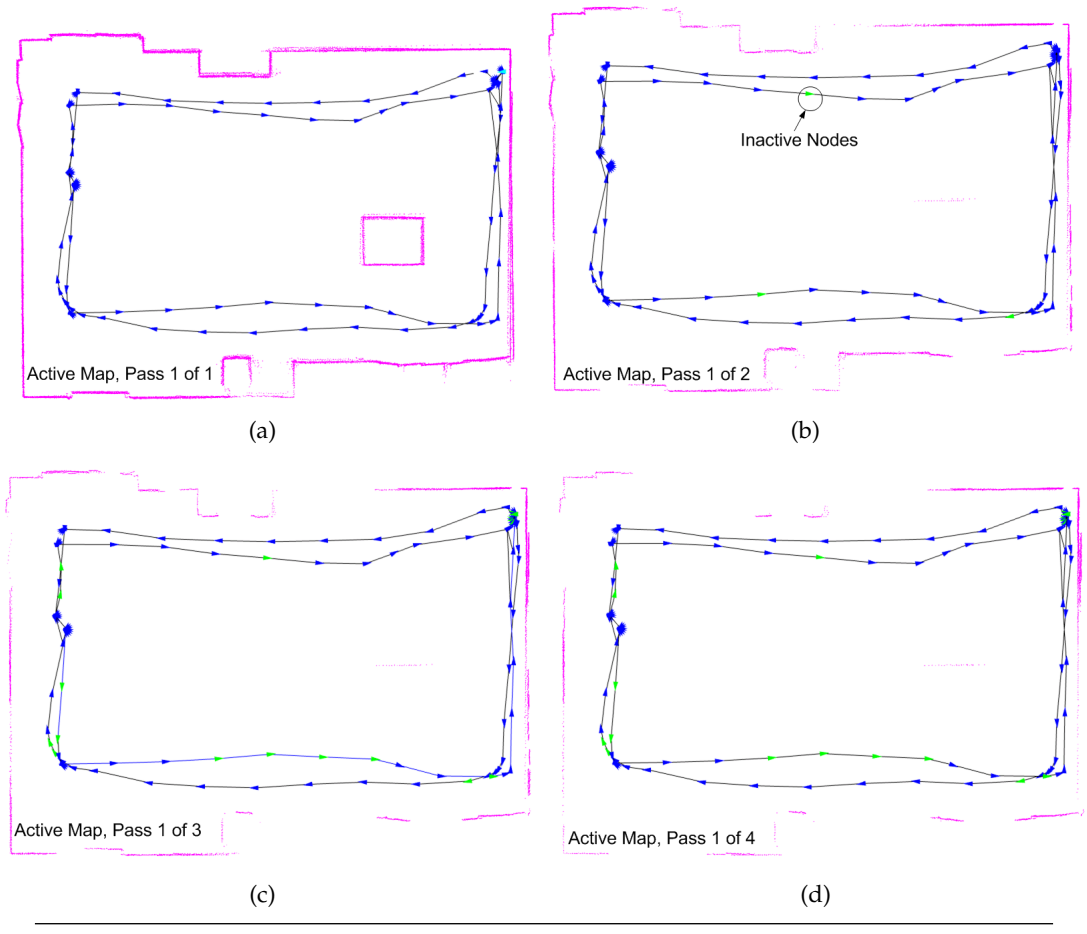


Figure 5-15: Evolution of the submap from pass 1 as the robot makes up to 4 passes through the Reading Room, where no nodes/constraints are removed.

from the final pass is given in Figure 5-18 and the regular nodes and the change nodes are shown.

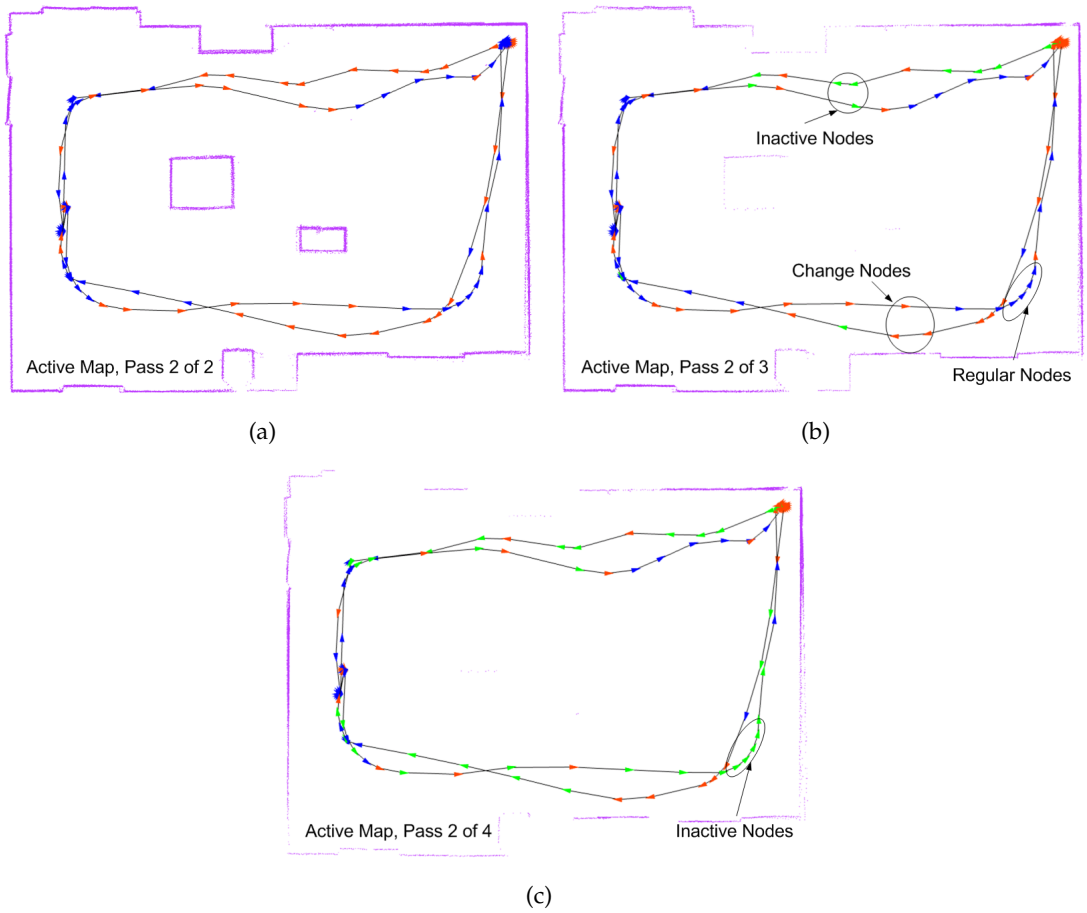


Figure 5-16: Evolution of the submap from pass 2 from the Reading Room, where DPG-SLAM-NR was applied.

To summarize the results of the DPG-SLAM-NR algorithm, the graph in Figure 5-19 shows the rate of growth over each iteration for the number of nodes, constraints, inactive nodes and change nodes.

5.6.2 DPG-SLAM

This section highlights the results of the node and constraint removal, DPG-SLAM, on the 4 passes through the CSAIL Reading Room data set. The active map, DPG, and the dynamic map are given in Figure 5-13. The amount of ghosting points in the active map is much less than in the DPG-SLAM-NR active map. This is due to the complete removal of nodes, from the removal chain, where not all the sectors are turned off. In addition, Table 5.4 summarizes the total number of nodes, constraints, change nodes, inactive nodes, removed nodes, and removed constraints, after each pass.

The dynamic map from the DPG-SLAM algorithm is shown for each pass in Figure 5-20. Also, the evolution of the submaps from each pass are Figure 5-21, Figure 5-22, and Figure 5-23. These images show the reduction in the number of nodes after each pass, as

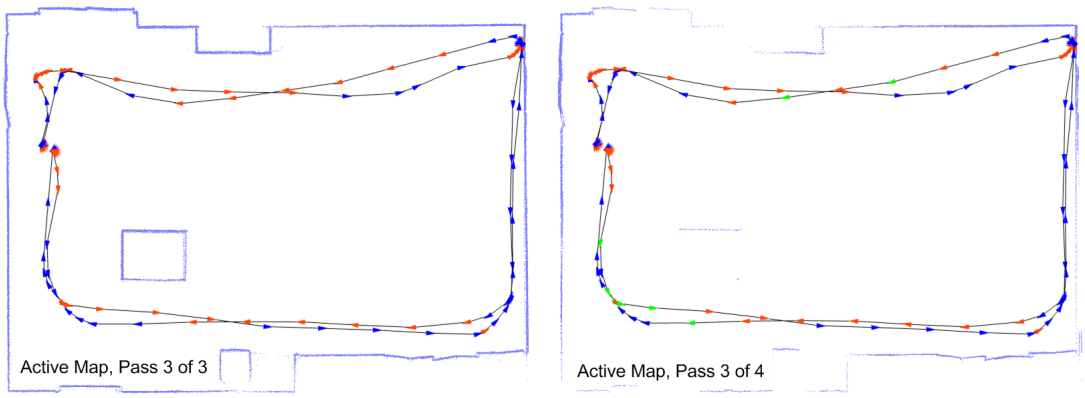


Figure 5-17: Evolution of the submap from pass 3 from the Reading Room, where DPG-SLAM-NR was applied.

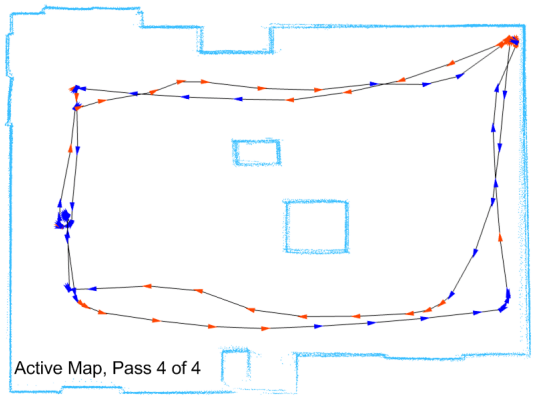


Figure 5-18: The DPG-SLAM-NR submap from the final pass through the Reading Room, pass 4, showing the change nodes.

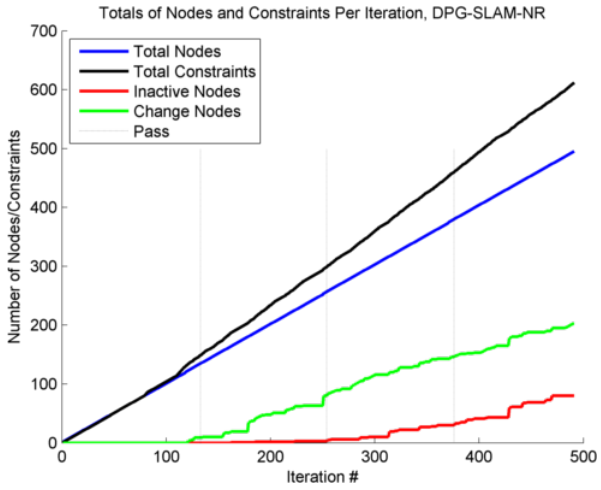


Figure 5-19: Summary of Total Nodes/Constraints DPG-SLAM, CSAIL Reading Room.

Table 5.4: Totals of the node types and constraints in the DPG after each pass. The data is from 4 passes through the CSAIL Reading Room and DPG-SLAM is applied.

#Passes	Nodes	Constraints	Change	Inactive	Rem Nodes	Rem Constraints
1	119	129	—-	—-	—-	—-
2	238	277	63	3	14	17
3	318	381	135	30	55	87
4	395	472	192	69	100	171

the active maps fade. Some nodes appear as singleton, but they are connected to another node from a different pass, via a loop constraint.

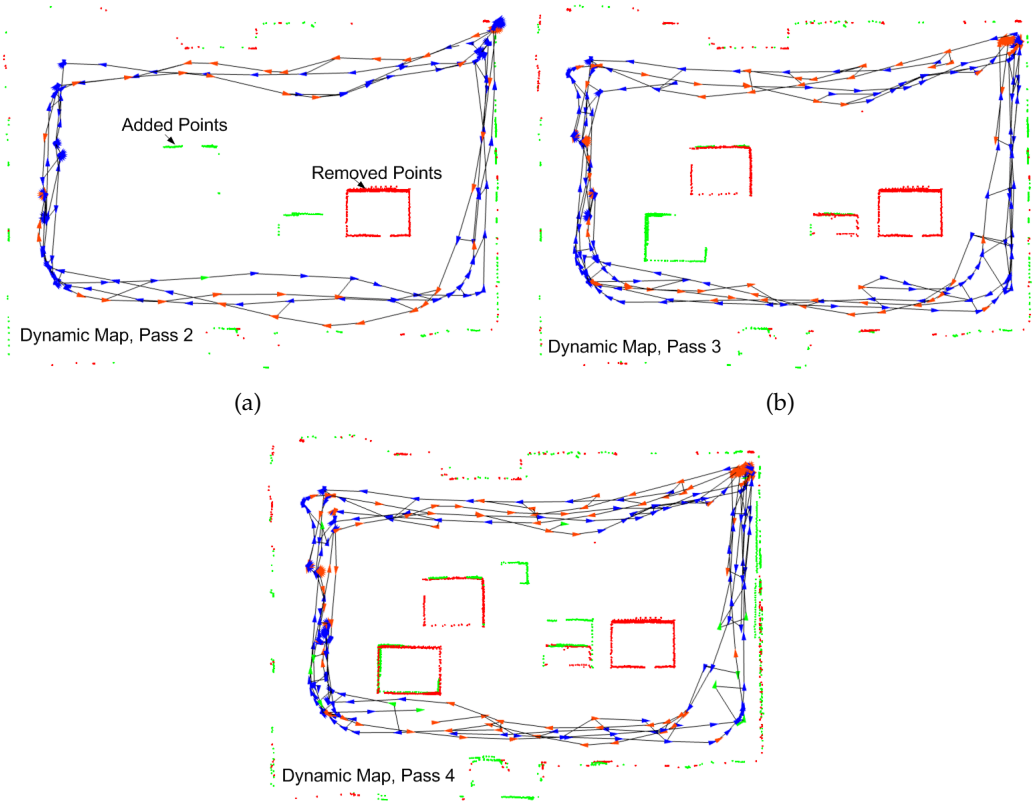


Figure 5-20: Images of the dynamic map after each pass.

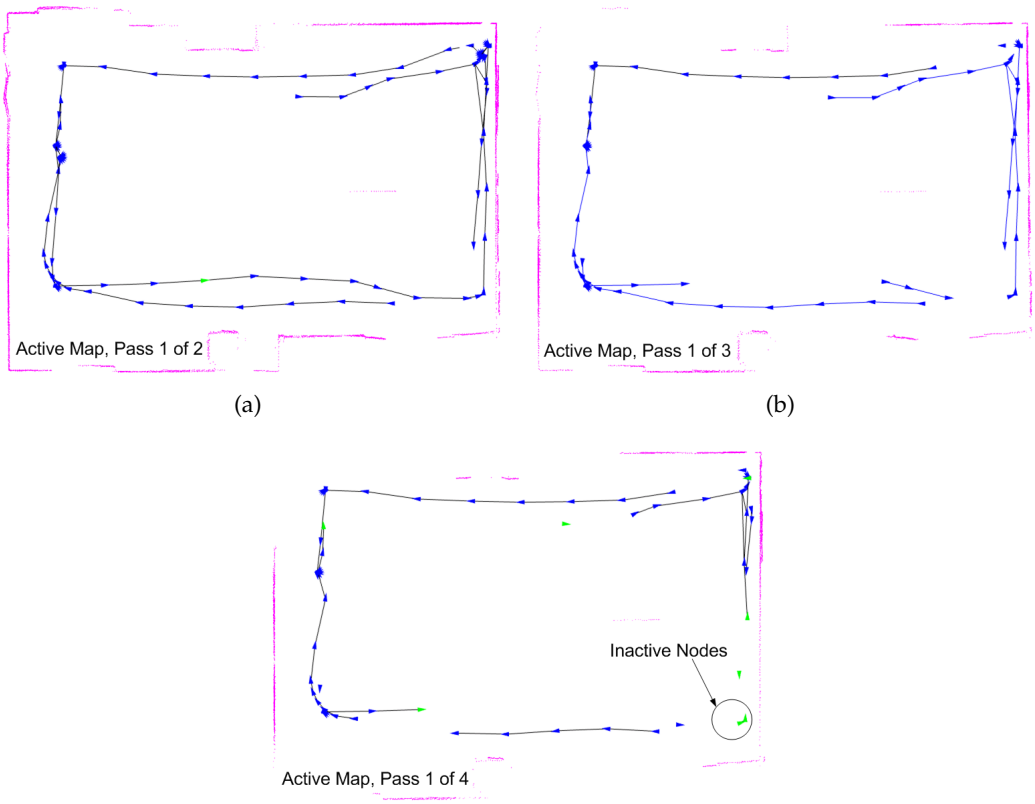
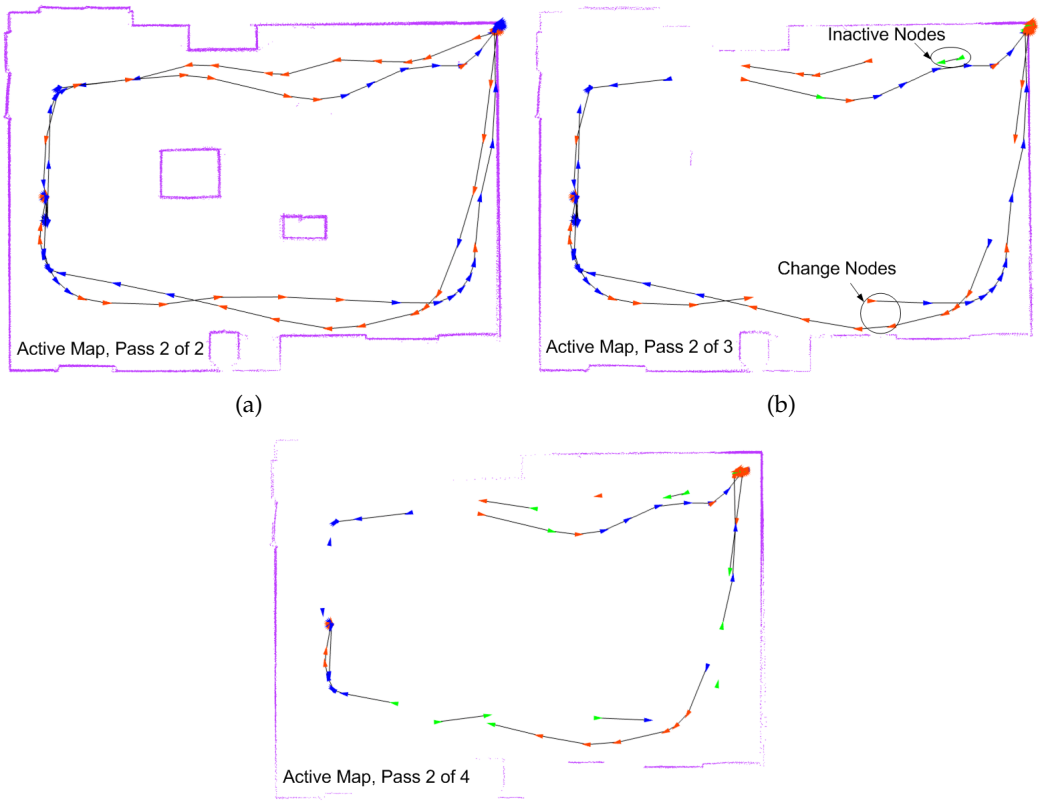


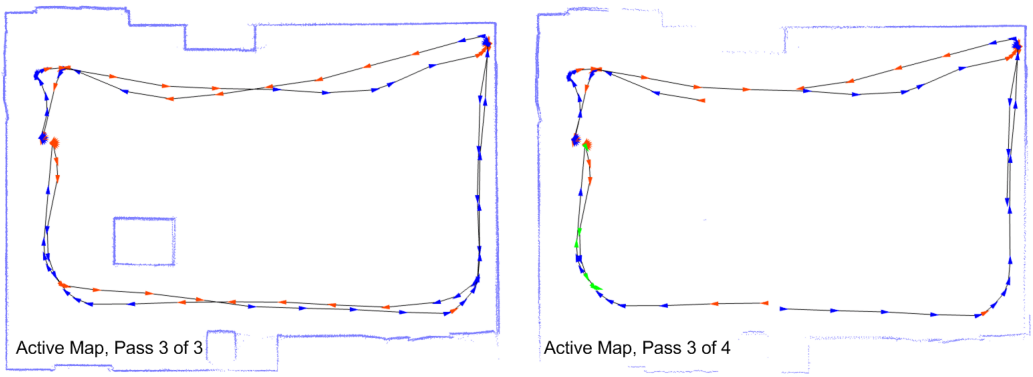
Figure 5-21: Evolution of the submap from pass 1 as the robot makes up to 4 passes through the Reading Room. The pass 1 submap is not shown because it is the same as the pass 1 submap in DPG-SLAM-NR.



(a) (b)

(c)

Figure 5-22: Evolution of the submap from pass 2 as the robot makes 4 passes through the Reading Room.



(a) (b)

Figure 5-23: Evolution of the submap from pass 3 as the robot makes 4 passes through the Reading Room.

The map of the fourth pass is shown Figure 5-24, where the regular nodes and the change nodes are highlighted. Lastly, the graph depicting the growth of each node type and constraint over all iterations is given in Figure 5-25.

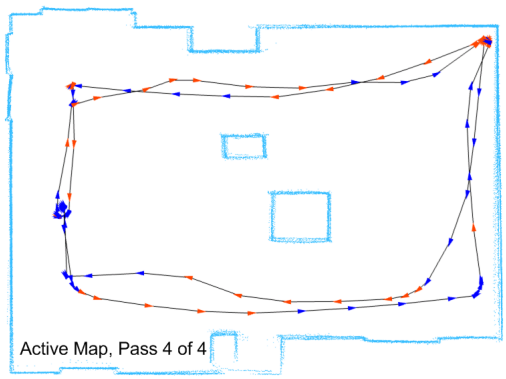


Figure 5-24: Image of pass 4 at the end of 4 passes through the Reading Room.

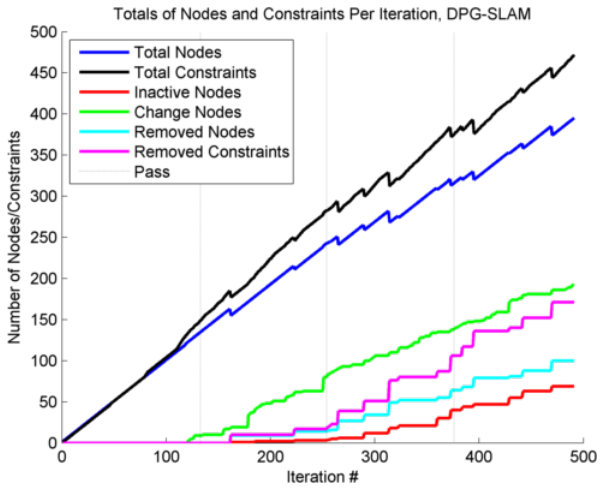


Figure 5-25: Summary of Total Nodes/Constraints DPG-SLAM, CSAIL Reading Room.

5.7 Summary

This chapter presented the DPG-SLAM method, a novel procedure for long-term persistent mapping in dynamic environments. DPG-SLAM operates on a Dynamic Pose Graph model and maintains accurate and up-to-date active and dynamic maps. A detailed example of DPG-SLAM and a variant, DPG-SLAM-NR where no nodes are removed, was provided. The example illustrated the affects of removing and not removing nodes and constraints from a DPG. In the following Chapter, a wide range of experiments that demonstrate the efficacy of DPG-SLAM and DPG-SLAM-NR is presented.

Chapter 6

Analysis and Results

In this thesis we develop a novel approach to solving the long-term mobile robot mapping problem, called Dynamic Pose Graph SLAM (DPG-SLAM). We focus on the critical challenge of maintaining a map of a dynamic environment focusing on changes that occur slowly with the passage of time. To analyze and empirically validate the DPG-SLAM method, results on data collected from several passes through two real-world indoor data sets, the CSAIL Reading Room and the University of Tubingen Robot Lab, are presented. Specifically, results are presented from experiments where the two algorithms, DPG-SLAM-NR (no nodes or constraints removed) and DPG-SLAM, are applied to short-term and long-term operation in each environment. We are specifically interested in determining the accuracy and the efficiency of our DPG-SLAM method. Our analysis illustrates the trade-off between maintaining an up-to-date map while keeping the DPG nodes and constraints, and maintaining an accurate and up-to-date map while removing nodes and constraints.

6.1 Experimental Data

Laser range data from the CSAIL Reading Room and University of Tubingen Robot Lab were generated from a B21 mobile robot equipped with a forward-facing laser range finder that was driven around each space for numerous passes.

CSAIL Reading Room. In this data set the robot makes a total of 20 passes throughout a room that is approximately $10.5m$ x $7.5m$. The environment consists of two low-dynamic objects which are boxes labeled d_1 and d_2 (see Figure 6-1). To simulate change, one or both of the boxes are moved, added, or removed after each pass as seen in Figure 6-1.

University of Tubingen Robot Lab. This data set was provided by Peter Biber [9]. To demonstrate a robot operating for a long period in a dynamic environment, the robot was hand-driven around an indoor space over a period of 5 weeks. In general, data was collected in the morning, after lunch, and in the evening creating a data set of over 70 passes. The changes occurred naturally as a part of the environment. The environment is approximately $50m$ x $40m$ and is "L-shaped" with a left corridor, a right corridor, and a bottom corridor. Figure 6-2 depicts four example maps of the Robot Lab generated by pose-graph SLAM, where a few changes between each map are identified.

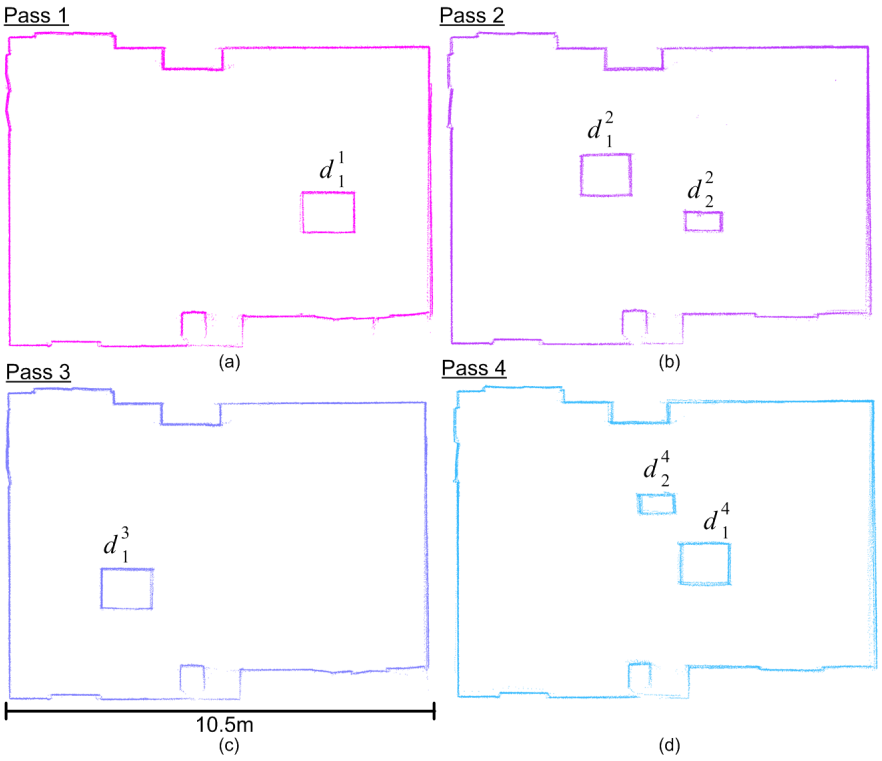


Figure 6-1: Example maps of four individual CSAIL Reading Room maps after four passes. Pose graph SLAM is to generate each map. Subscripts of each box (low-dynamic object) are used for identification and superscripts are used to denote the pass number.

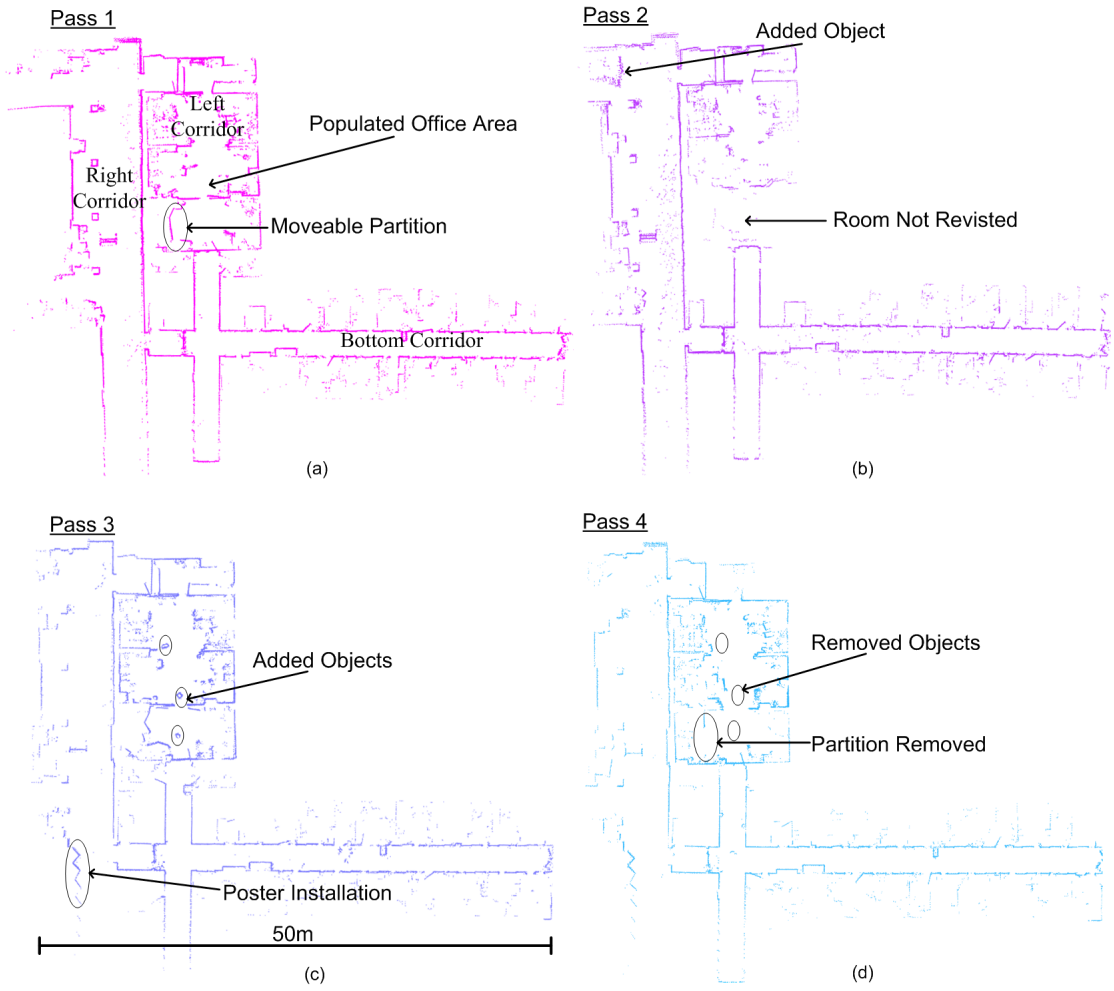


Figure 6-2: Example maps generated by pose graph SLAM from the University of Tubingen data set. A few of the changes resulting from added, moved or removed low-dynamic objects are highlighted. For example, the right-side corridor contains student offices and is labeled as a populated office area.

6.2 Overview

DPG-SLAM provides a mobile robot with the ability to maintain an accurate and up-to-date map of the environment. Older changes (from stale data) are eventually removed from the active map and added to the dynamic map. At the same time, static parts of the environment that have been revisited often become more prominent in the active map. To analyze the accuracy and the efficiency of DPG-SLAM, a set of experiments to qualitatively and quantitatively interpret the efficacy of our method are conducted. In the experiments, results are presented from DPG-SLAM-NR, where no nodes or constraints are removed, and DPG-SLAM, where nodes and constraints are removed. Results from short-term operation (a few passes) and long-term operation (several passes) in the CSAIL Reading Room and the University of Tubingen Robot Lab are presented.

In each experiment, we varied the number of sectors (partitions of a laser range scan) of a node. The number of sectors affects the overall accuracy of that active map in three main ways. map quality in three main ways. First, the sectors of a node that are "on" contain the points that make up the active map. Second, change detection is performed specifically on the points in sectors that are "on." The points in these sectors affect the results of change detection and updating the active and dynamic maps. Second, a node with fewer sectors has the potential to become inactive sooner than a node with several sectors. Recall, an inactive node is one that has all of its sectors turned "off." These nodes are also candidates for removal, and if removed, affect the pose estimates and the active map.

Below is a summary of the graphs and performance measures used in this chapter to qualitatively and quantitatively demonstrate the efficacy of DPG-SLAM.

Active Map Images. The orthogonal projection of the active maps from each pass on top of each other, and a three-dimensional side-view of the active map. The active map is illustrated by layers of maps generated from each pass. These layers are shown along the vertical (z-axis) in the images. In the images each map is color-coded according to a color-scale from magenta to cyan. The map from the oldest pass is magenta and the map from the most recent pass is cyan.

Total Nodes. The total number of nodes in the DPG after each iteration and each pass.

Total Constraints. The total number of DPG constraints after each iteration, where numerous iterations make-up a pass.

Total Change Nodes. The total number of change nodes per iteration.

Total Inactive Nodes. The total number of inactive nodes after each iteration.

Total Removed Nodes. Total nodes removed over each iteration, and thus each pass. Illustrates how sensitive node removal is as a function of the environment changes. Note, node removal also results in the removal of one or more constraints.

Total Removed Constraints. The number of constraints removed as a result of applying DPG-SLAM with node and constraint removal.

Static Histogram. This is a two-dimensional histogram that represents the density of the static parts of the environment for each pass. It is represented as a gray-scale grid (or bitmap) of that static parts of the environment created from the active map. The aim of this measure is to show how well the static information is preserved in the

active map. For example, if the robot makes four passes through an environment and measures the same wall each time, then that wall will be dark in the static histogram. However, if the robot only measures the wall two out of the four times through the environment the wall will be shown as gray in the static histogram. This measure highlights static and added objects based on the number of time the robot re-visits an are and takes multiple measurements of the static object. The focus is on the static objects, such as the walls.

Mean Point Distance. Computes the mean of the distances for each range point in the active map to ground truth. It is applied to the CSAIL Reading Room data set where part of the ground truth, represented by line segments, is known. An example of the method to compute the mean distance is given Figure 6-3. The active map points in Figure 6-3(a) are selected to estimate the transform from the active map frame to the ground truth frame by randomly selecting a subset of active map points (shown in black in Figure 6-3(b)). Then then all the active map points are assigned to the closest line segment and their distance to the line segment is computed (see Figure 6-3(c)).

Run Time Ratio. Computes the ratio of the runtime of the DPG-SLAM algorithm to the DPG-SLAM-NR algorithm (i.e. DPG-SLAM/DPG-SLAM-NR).

Experimental results are shown on the CSAIL Reading Room data set followed by the University of Tubingen Robot Lab data set. In the beginning of each section we present a summary of results with a comparison of DPG-SLAM and DPG-SLAM-NR. Then results are given for both short-term and long-term operation, where the number of sectors is varied. The Reading Room short-term data is collected from four passes and the long-term data is collected from the robot making twenty passes. In the Robot Lab data set the short-term data is collected data from four passes and the long-term data was collected from sixty passes. In addition, in the Robot Lab data set ground truth is not known and the low-dynamic objects are also not known. Images of the active maps, general graphs of the totals of each node and constraint, and performance measures are provided to illustrate the performance of DPG-SLAM on these two slowly changing dynamic environments.

6.3 CSAIL Reading Room Experimental Analysis and Results

We tested noes with 1, 3, ..., 12 sectors for the CSAIL Reading Room experiments. The change threshold was 20%, that is a change was detected if 20% of the current scan is different than previous passes in the same area. The two low-dynamic objects, d_1 and d_2, are added, removed, or moved within the interior of the room after each pass. Each pass consisted of the robot navigating through the room such that it is able to collect measurements of the entire space by the end of its pass. As a result, the best-case active map should contain ranges from the walls from every pass, as well as ranges from d_1 and d_2 measured during the most recent pass (see Chapter 5 for details).

6.3.1 Summary Comparison of DPG-SLAM and DPG-SLAM-NR, 20 Passes

Figure 6-4 shows the images resulting from applying DPG-SLAM (on the right) and DPG-SLAM-NR (on the right) to 20 passes through the CSAIL reading room. In this experiment each node had 5 sectors. The robot traveled a total distance of 1.0km. The images show

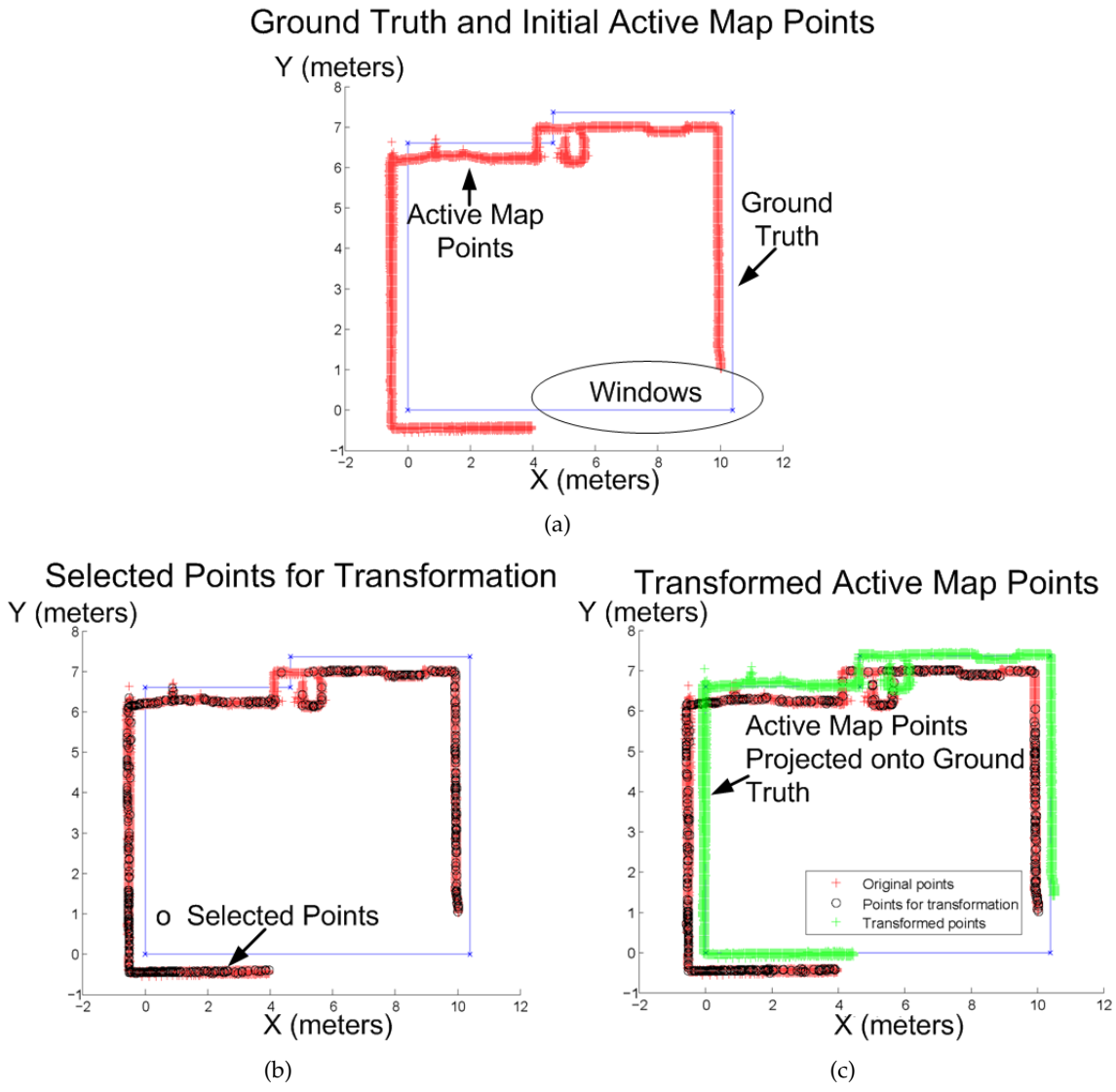


Figure 6-3: Example of the steps to compute the mean distance of the active map points. The areas where the windows are shown are not used in the computation. Note, the process is similar to applying ICP [7, 48]to match laser range scans.

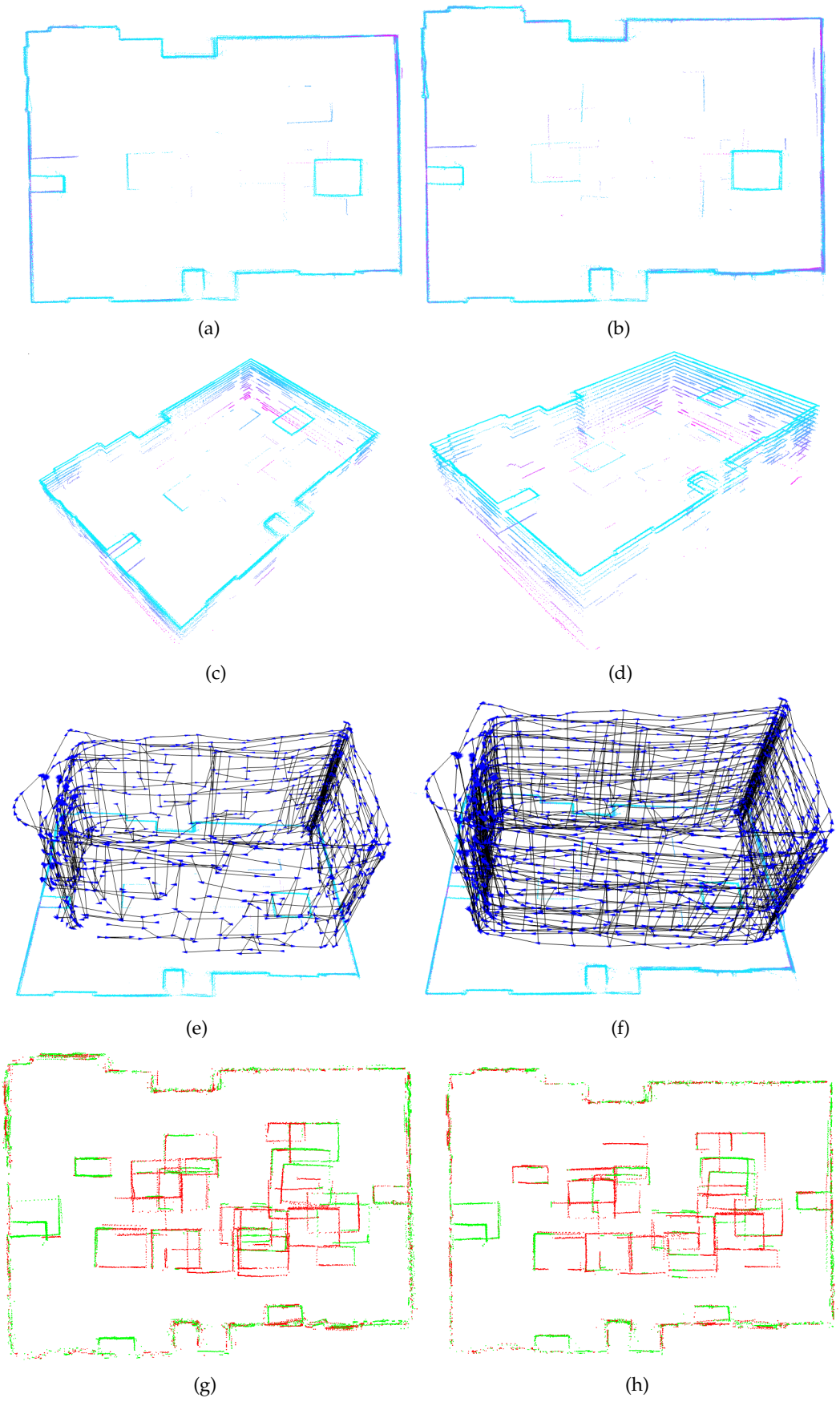


Figure 6-4: On the left are the active, DPG, and dynamic maps for the DPG-SLAM algorithm and on the right are the same images for the DPG-SLAM-NR algorithm. The points are color-coded from each pass, where magenta is the oldest pass and cyan is the most recent pass. In this experiment each node had 5 sectors.

that the density of static points (from the walls) is greater for the DPG-SLAM-NR than the DPG-SLAM. On the other hand, the size of the DPG shown in Figure 6-4(e) is significantly reduced. There are a total of 1,345 nodes and 1,647 constraints resulting from DPG-SLAM, and 2,468 nodes and 3,158 constraints from the DPG-SLAM-NR algorithm.

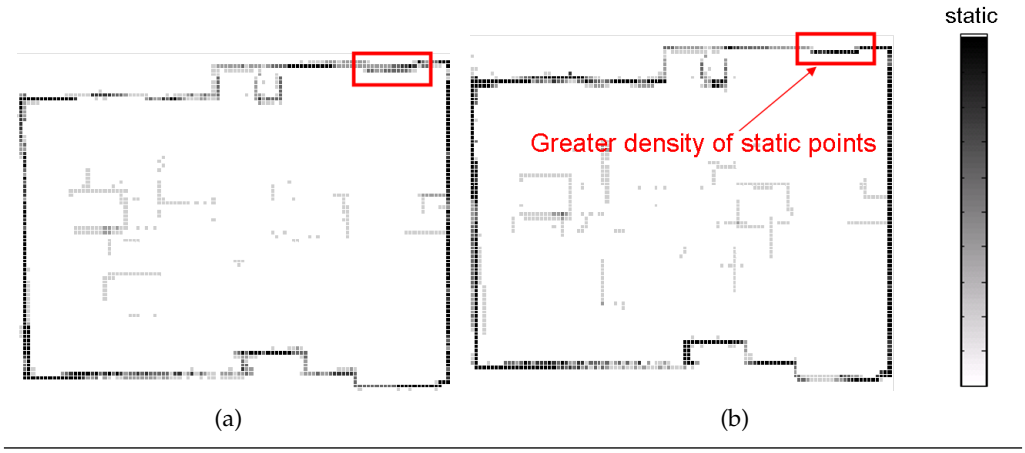


Figure 6-5: Static histograms of DPG-SLAM (on the left) and DPG-SLAM-NR (on the right) active maps. The density of static map points is shown as being darker in the images.

To illustrate the accuracy of our method we show the static histograms in Figure 6-5, and the ground truth accuracy in Figure 6-6. The static histograms show clearly darker area for the DPG-SLAM-NR method. This is due to nodes being removed during DPG-SLAM that are not inactive, and thus, their valid points are removed from the active map. Lastly, to demonstrate the efficiency of removing nodes and constraints from the DPG, we present the ratio of the runtime of DPG-SLAM to DPG-SLAM-NR in Figure 6-7. During the early iterations (and passes) the computational cost of DPG-SLAM is slightly greater than not removing nodes and constraints. However, in later iterations it is clear that reducing the size of the DPG yields a great savings in computation.

6.3.2 Short-term Operation, Four Passes

Below is an image of Pose Graph SLAM using iSAM applied to the short-term CSAIL Reading Room data (see Figure 6-8). The robot traveled 205 meters and a total of 495 nodes and 612 constraints (edges) were added to the pose graph.

DPG-SLAM-NR Results

Figure 6-9 shows the active maps generated for varying number of sectors where no node or edge is removed. In this set of experiments, the resulting number of nodes and edges remain the same, ie. 495 nodes and 612 constraints. This is because as mentioned in Chapter 5, potential loop constraints can be generated from ranges in active and inactive nodes.

The active map resulting from 1 sector show no ghosting, whereas the active map from twelve sectors, though minimal, shows the most amount of ghosting. It is also evident that the quality and density, measured by the static histogram, of the active map is sensitive to the number sectors. The amount of ranges that remain in the active map after each pass is an indicator of the density of points. For example, with 1 sector as shown in Figure 6-27,

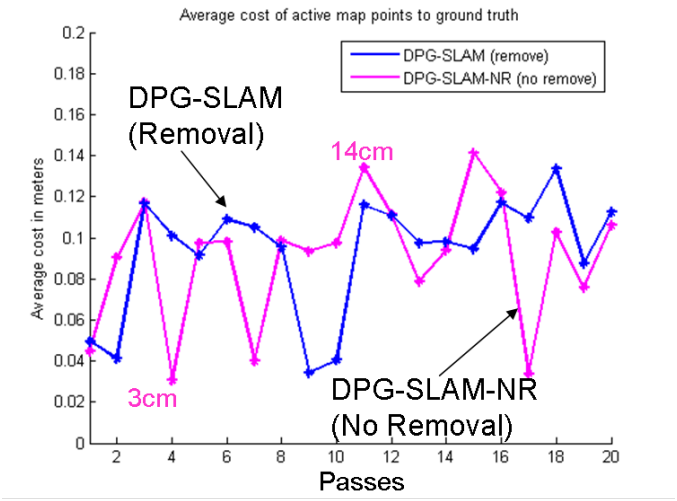


Figure 6-6: Ground truth of 20 passes through the Reading Room, with an active map error: 3cm-13cm.

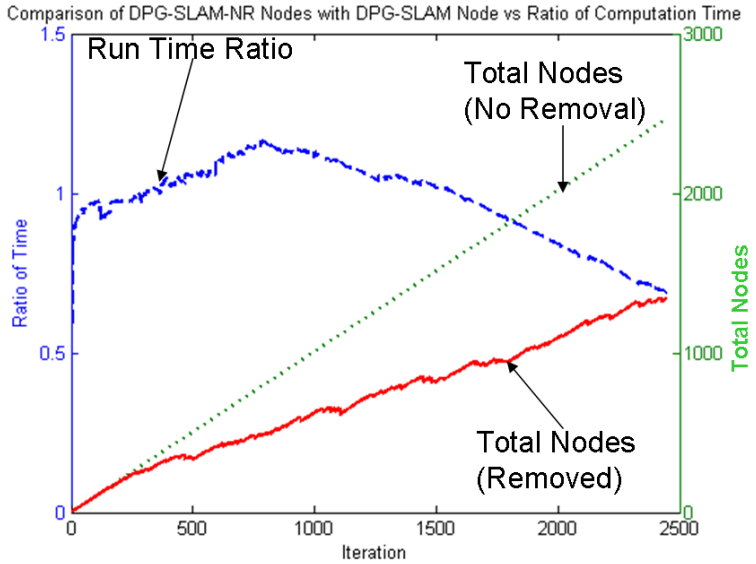


Figure 6-7: Run time ratio of 20 passes through the Reading Room.

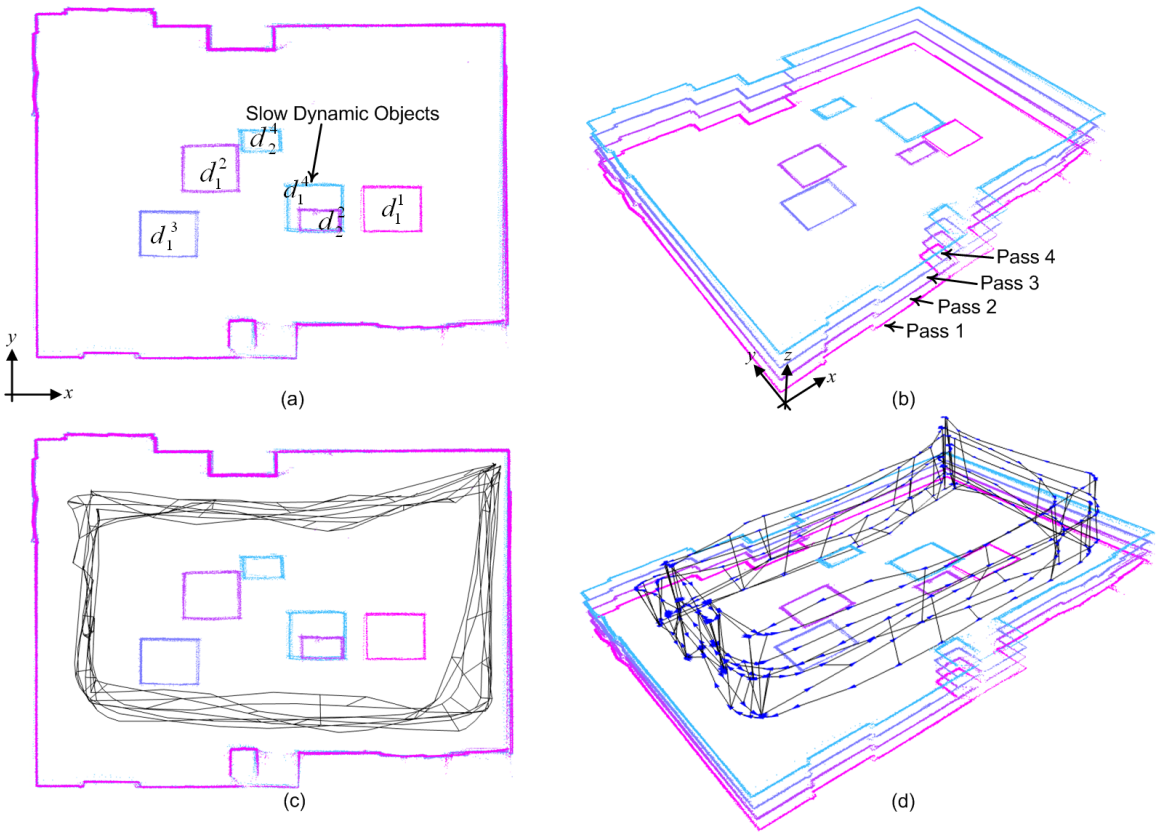


Figure 6-8: The map generated by applying pose graph slam to the CSAIL Reading Room data set with four passes.

only ranges from pass 3 and pass 4 remain in the active map; whereas in the experiment with twelve sectors the active map contains ranges from all four passes.

The total number of change nodes and inactive nodes for each experiment after the robot makes four passes through the CSAIL Reading Room is shown in Table 6.1. The totals each node type as the robot maintains and updates the DPG is also shown in Figure 6-10. The total number of change nodes is approximately the same for the experiments with three or more sectors. In general, the algorithm should detect a change at approximately the same poses, regardless of the number of sectors. However, the number of change nodes is much greater for 1 sector. In the CSAIL Reading Room experiments, the robot navigates along the boundary of the environment and the changes occur in the interior. Thus, part of each scan taken at each node contains ranges from the walls, which do not change. In the 1 sector case, a node's entire scan is turned off when a change is detected, and the node is inactive. Then the data at the node is assumed to be stale, and the node is not used again for change detection. In addition, the room changes significantly enough from pass to pass such that when a when attempting to detect change the only available candidates are mostly from the immediate pass. All other nearby nodes from earlier pases were previously set to inactive.

Table 6.1: Total number of change nodes and inactive nodes for each experiment after the robot makes four passes through the CSAIL Reading Room. There are a total of 495 nodes and 612 constraints in each experiment.

Sectors	Change	Inactive
1	349	369
3	222	250
6	211	54
9	212	15
12	214	3

Another observation is that the number of inactive nodes decreases greatly as the number of sectors increases. Recall that a node becomes inactive when all of its sectors are turned off (note, a change node can be also be an inactive note). Therefore, there has to be a change or a set of changes that effects all of the sectors of a node in order for the node to be inactive. Thus, it is expected the number of inactive nodes would decrease at the number of sectors increases. That is, as seen in Table 6.1 and Figure 6-10c, nodes with more sectors are less sensitive to becoming inactive.

The static histograms for the four passes through the CSAIL Reading Room DPG-SLAM-NR experiments are shown in Figure 6-11. The darker areas in each sub-figure denote the parts of the environment that have little or no change. That is, the parts of the environment with static objects. In the ideal case, the walls would be the darkest part of the environment, and ideally shown in black, as they remain static while the robot makes several passes. In Figure 6-11 the walls are shown as well as some of the low-dynamic objects that were not removed from the active map. For example, the 1 sector experiment (Figure 6-10(a)) depicts the walls with approximately the same intensity as the low-dynamic objects from the most recent pass (see Figure 6-9a). In this experiment, the inclusion of only the low-dynamic objects from pass 4 is accurate. However, the resulting active map from the 1 sector experiment only contains ranges from pass 3 and pass 4 (see Figure 6-9b). On the other hand, the static histogram for the 12 sectors experiment in Figure 6-11e shows

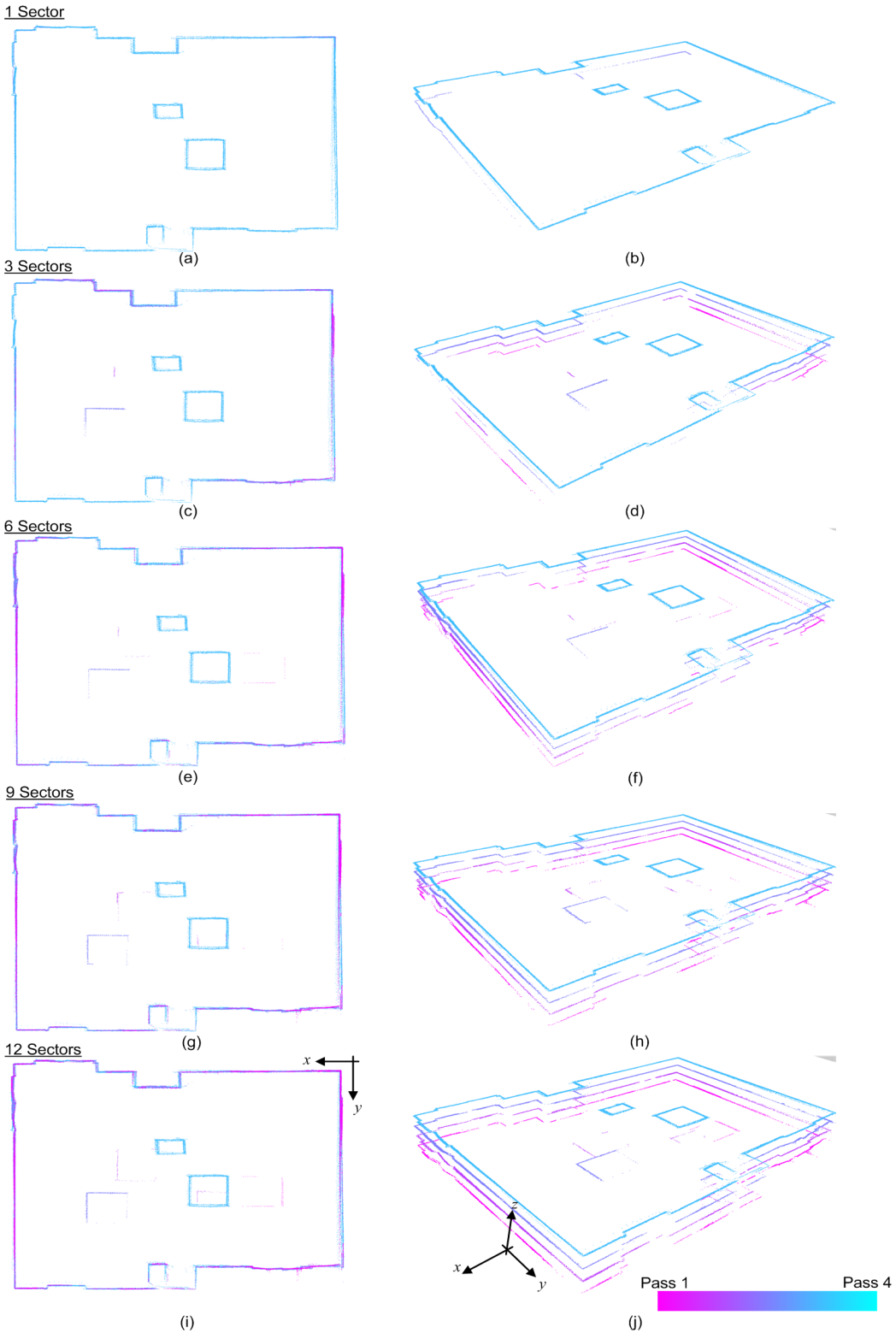


Figure 6-9: The active map generated from DPG-SLAM-NR with no node/edge removed. Each experiment contains a different number of sectors. The images show that as the number of sectors increase the amount of points in the resulting active map increases.

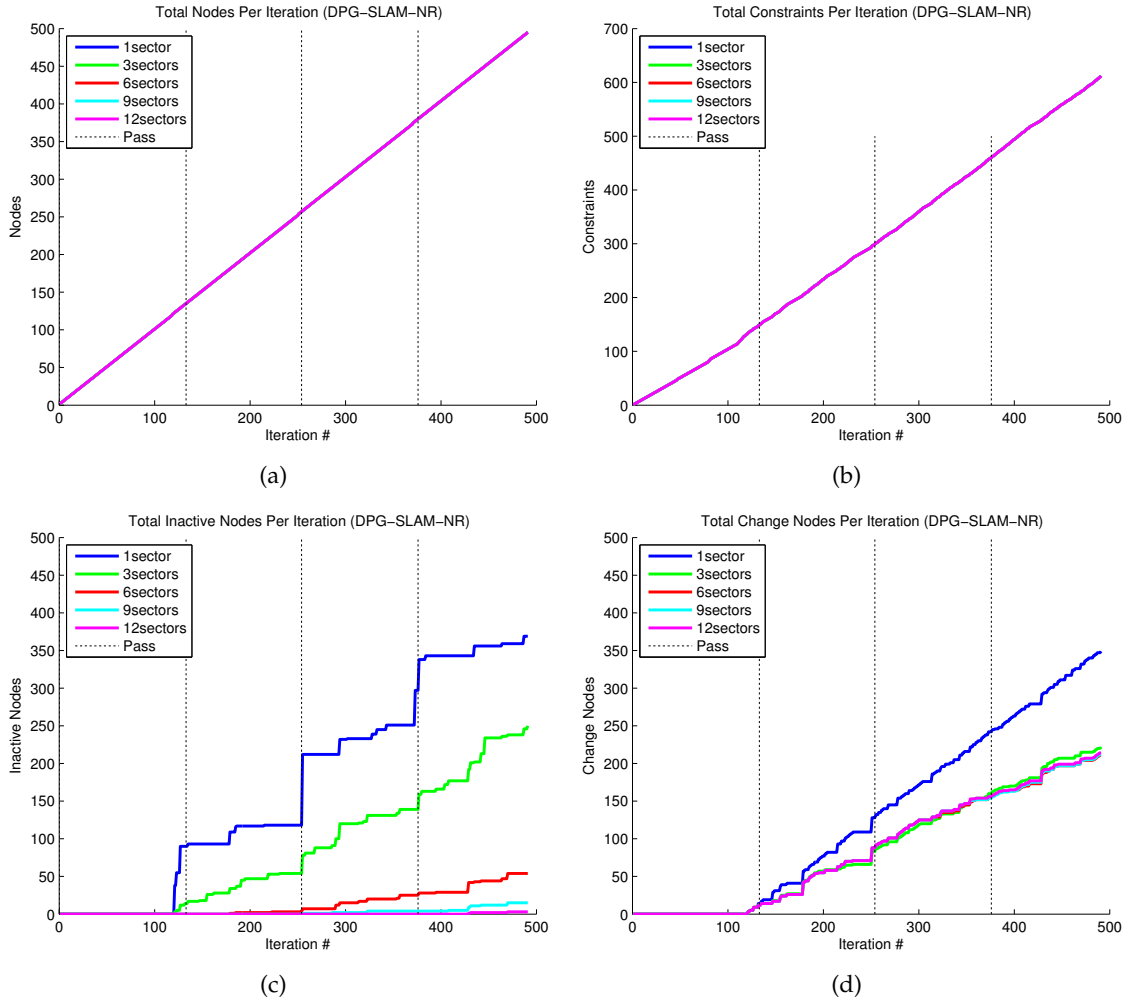


Figure 6-10: Totals of DPG nodes and constraints from the DPG-SLAM-NR algorithm as the robot makes four passes through the CSAIL Reading Room. In Figures (a) and (b) the total nodes and constraints increase at the same rate for each experiment, and all the lines are plotted on top of each other. This is a result of keeping the nodes and constraints in the DPG-SLAM-NR. Figure (c) illustrates how the number of inactive nodes increases with each pass. Finally, Figure (d) shows that, in general, the number of change nodes grow at about the same rate in the experiments with more than 1 sector.

the walls with dark outlines, as desired. But it also contains the ghosting from part of the low-dynamic objects from passes 1-3. This is an example of one of the algorithmic trade-offs between reacting to change above a certain threshold, where smaller changes may be a part of the resulting map.

The results of apply the mean-point-distance measure are shown in Table 6.2. The performance measure provides a measure of accuracy of the known ground truth. The table shows that in general, the results of DPG-SLAM-NR yield accurate active maps with an average distance of each point to ground truth between 3cm and 11cm for each experiment. The averages are also computed for each number of passes and number of sectors shown in the Mean-Pass row and the Mean-Sectors column in Table 6.2. Note that ground truth was measured by and is subject to human error.

Table 6.2: Mean-point-distance for each of the experiments. The mean and standard deviation for the number of passes and number of sectors are provided.

Sectors	P 1	P 1,2	P 1,2,3	P 1,2,3,4	Mean-Sector	Std-Sector
1	0.045m	0.106m	0.106m	0.107m	0.091m	0.031m
3	0.048m	0.038m	0.098m	0.038m	0.056m	0.029m
6	0.053m	0.100m	0.113m	0.033m	0.075m	0.038m
9	0.054m	0.044m	0.103m	0.090m	0.073m	0.028m
12	0.055m	0.045m	0.110m	0.032m	0.061m	0.034m
Mean-Pass	0.051m	0.067m	0.106m	0.060m	——	——
Std-Pass	0.004m	0.033m	0.006m	0.036m	——	——

DPG-SLAM Results

The active maps from DPG-SLAM on the CSAIL Reading Room data set with four passes are given in Figure 6-12. In the experiments the poses can shift as nodes and edges are remove from the dynamic pose graph and pose graph optimization (iSAM) is used to recalculate the pose estimates. As a result, there are differences in the active maps of DPG-SLAM-NR and DPG-SLAM. For example, the nine sectors active map has much less ghosting caused by old and stale laser scans (Figure 6-12g and h), as compared to the same nine sectors experiment where DPG-SLAM-NR is applied (Figure 6-9g and h).

Table 6.3 shows the totals resulting from removing nodes and constraints (edges), and Figure 6-13 shows how they grow with each pass. The results indicate that the threes sectors experiment is particularly sensitive and yielded the smallest DPG (the fewest nodes and edges). The number of change nodes is similar for both DPG-SLAM-NR and DPG-SLAM. This reinforces the conjecture that the nodes in the different experiments that are approximately in the same location should be identified as the same type either change node or not. In addition, as the number of sectors increases beyond three sectors, the number of removed nodes and edges decreases. As mentioned earlier, it is expected that the number of nodes and edges decreases because nodes with more sectors general take longer to become inactive.

The static histograms depicted in Figure 6-14 illustrate how well the walls were preserved in the active maps, event hough nodes, which store the laser scans, were removed. In contrast to the 1 sector experiment (see Figure 6-14a), the three sectors (see Figure 6-14b) has significantly more nodes and edges removed. When comparing the static histograms

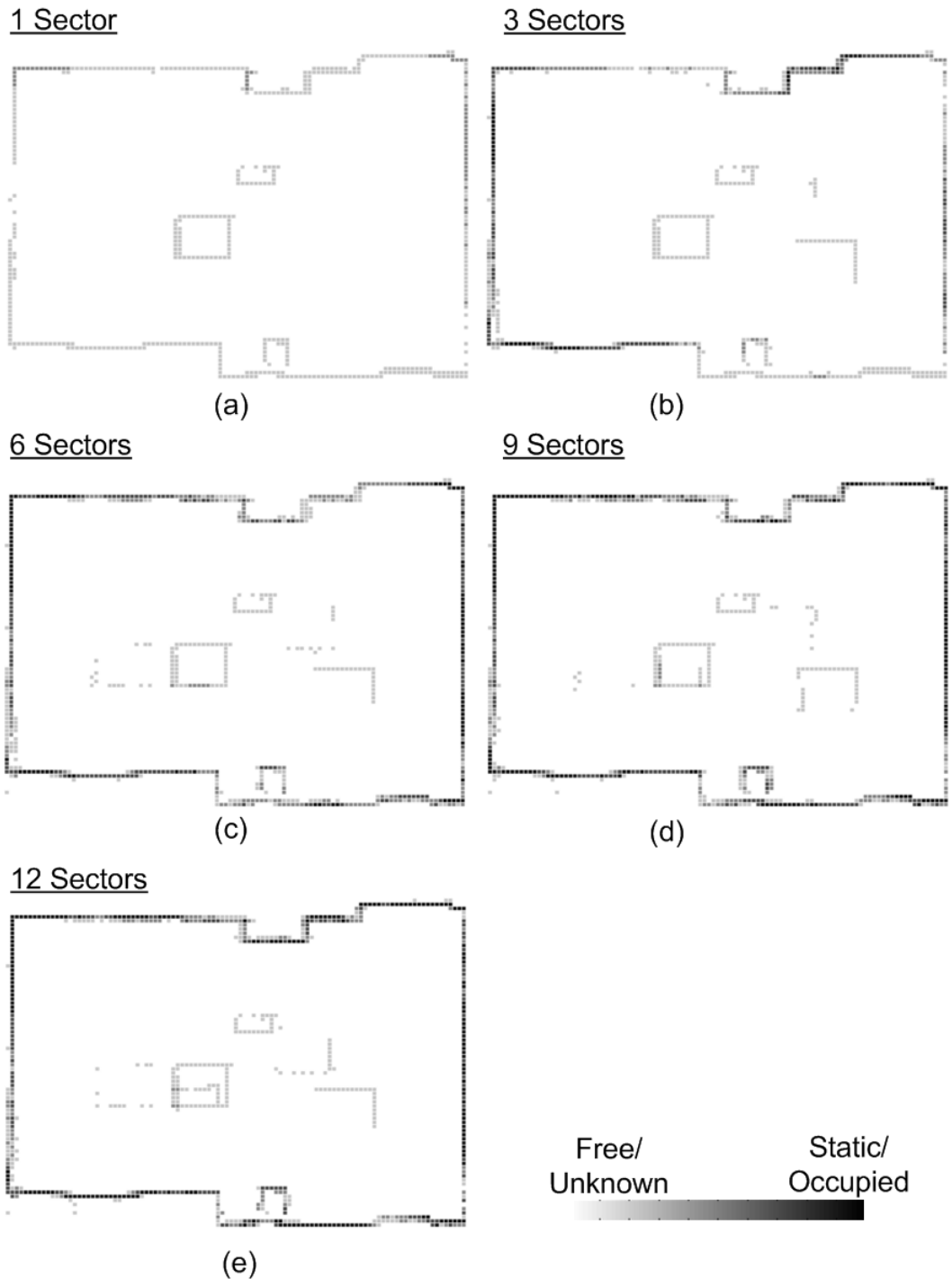


Figure 6-11: The static histogram generated from each active map for the different number of sectors. The intensity (darkness) of the lines represents the level of certainty the robot has about that part of the environment being static.

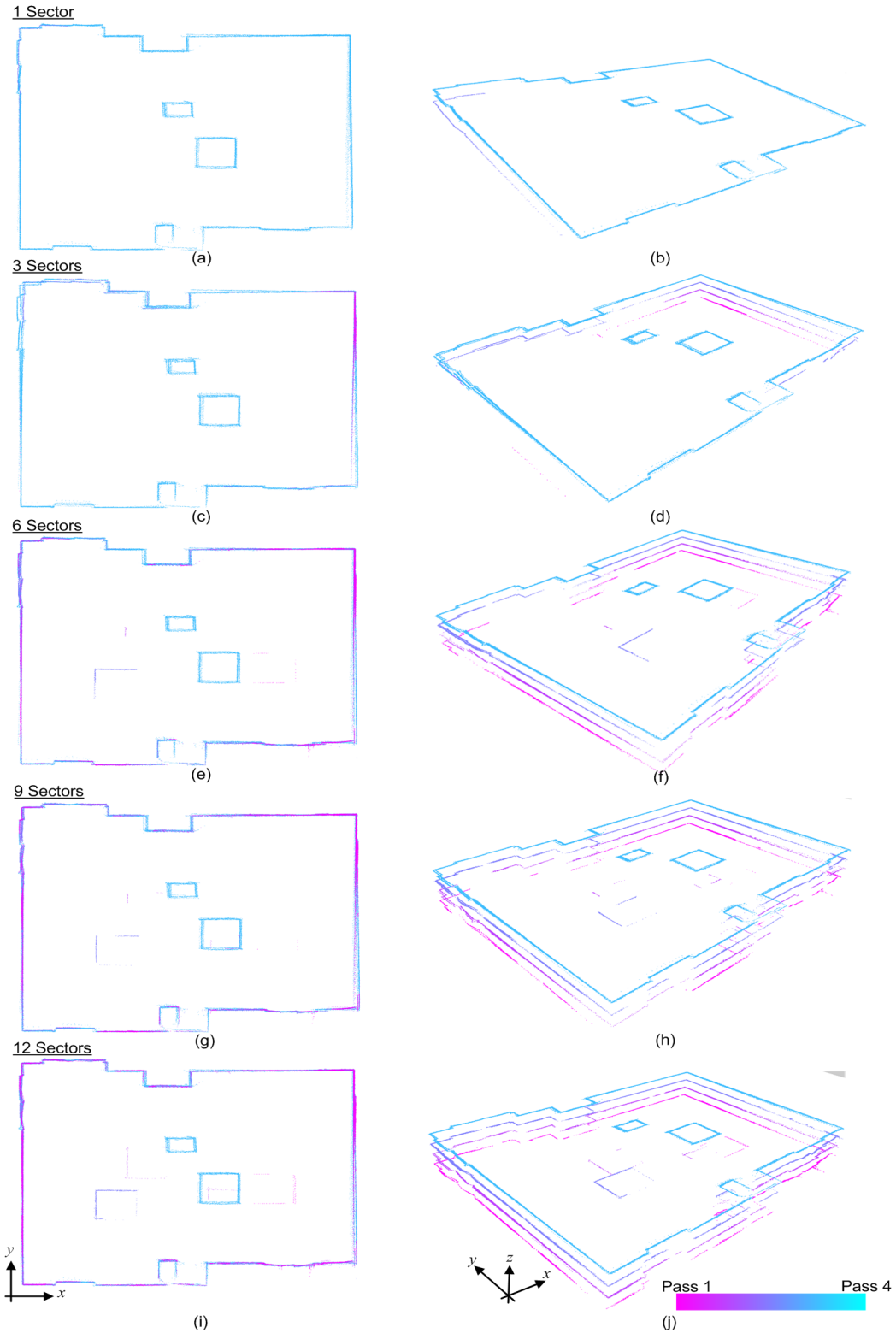


Figure 6-12: The active map generated from DPG-SLAM where nodes and edges are inserted and can be removed. The images show that in the CSAIL Reading Room data set, as the number of sectors increase the amount of range points in each active map increases.

Table 6.3: Totals for each type of node after the robot makes four passes through the CSAIL Reading Room and DPG-SLAM was applied.

Sectors	Nodes	Constraints	Change	Inactive	Rem Nodes	Rem Constraints
1	450	547	357	365	45	78
3	307	382	261	227	188	284
6	448	549	208	36	47	85
9	465	573	207	15	30	52
12	492	605	214	3	3	9

of the one and three sector experiment, the three sectors static histogram yields better results to the 1 sector. The three sectors static histogram contains darker walls than the 1 sector; specifically along the left wall where ranges from all four passes are included in the resulting active map. However, the DPG of the three sectors is significantly smaller, reducing the overall pose graph optimization computation time.

Finally, Table 6.4 shows the accuracy computed by the mean-point-distance of the DPG-SLAM active maps after four passes through the CSAIL Reading Room. Like the DPG-SLAM-NR results, the measure shows accurate maps where the average distance of a point is at most 11cm.

Table 6.4: Average sum of distance in meters for selected active map points projected onto the ground truth. The data is shown for the active map after increasing number of passes where DPG-SLAM was applied.

Sectors	P 1	P 1,2	P 1,2,3	P 1,2,3,4	Mean-Sector	Std-Sector
1	0.047m	0.103m	0.109m	0.102m	0.090m	0.029m
3	0.046m	0.034m	0.113m	0.092m	0.071m	0.037m
6	0.049m	0.040m	0.040m	0.031m	0.040m	0.007m
9	0.047m	0.041m	0.102m	0.105m	0.074m	0.035m
12	0.049m	0.102m	0.110m	0.093m	0.089m	0.030m
Mean-Pass	0.048m	0.064m	0.095m	0.085m	——	——
Std-Pass	0.001m	0.035m	0.031m	0.031m	——	——

6.3.3 Long-term Operation, Twenty Passes

To simulate long-term operation in the CSAIL Reading Room, a mobile robot was driven around the room for twenty passes. After each of the passes, at least one of the two low-dynamic objects was moved, removed or added (if not previously in the space). Results for 1, 3, 6, 9, and 12 sector experiments are provided where both DPG-SLAM-NR and DPG-SLAM are applied. In these experiments the robot traveled 1.0km with a total of 2,468 nodes and 3,158 constraints.

Images od Pose Graph SLAM applied to the twenty passes are given in Figure 6-15. The robot traveled 1,014 meters and the pose graph contains 2,468 nodes and 3,158 constraints. Without continuous map maintenance and change detection, the map of the environment appears cluttered with numerous objects. The ideal active map would contain ranges from all the walls collected over all twenty passes, as well as the two low-dynamic objects as they are positioned during the final pass.

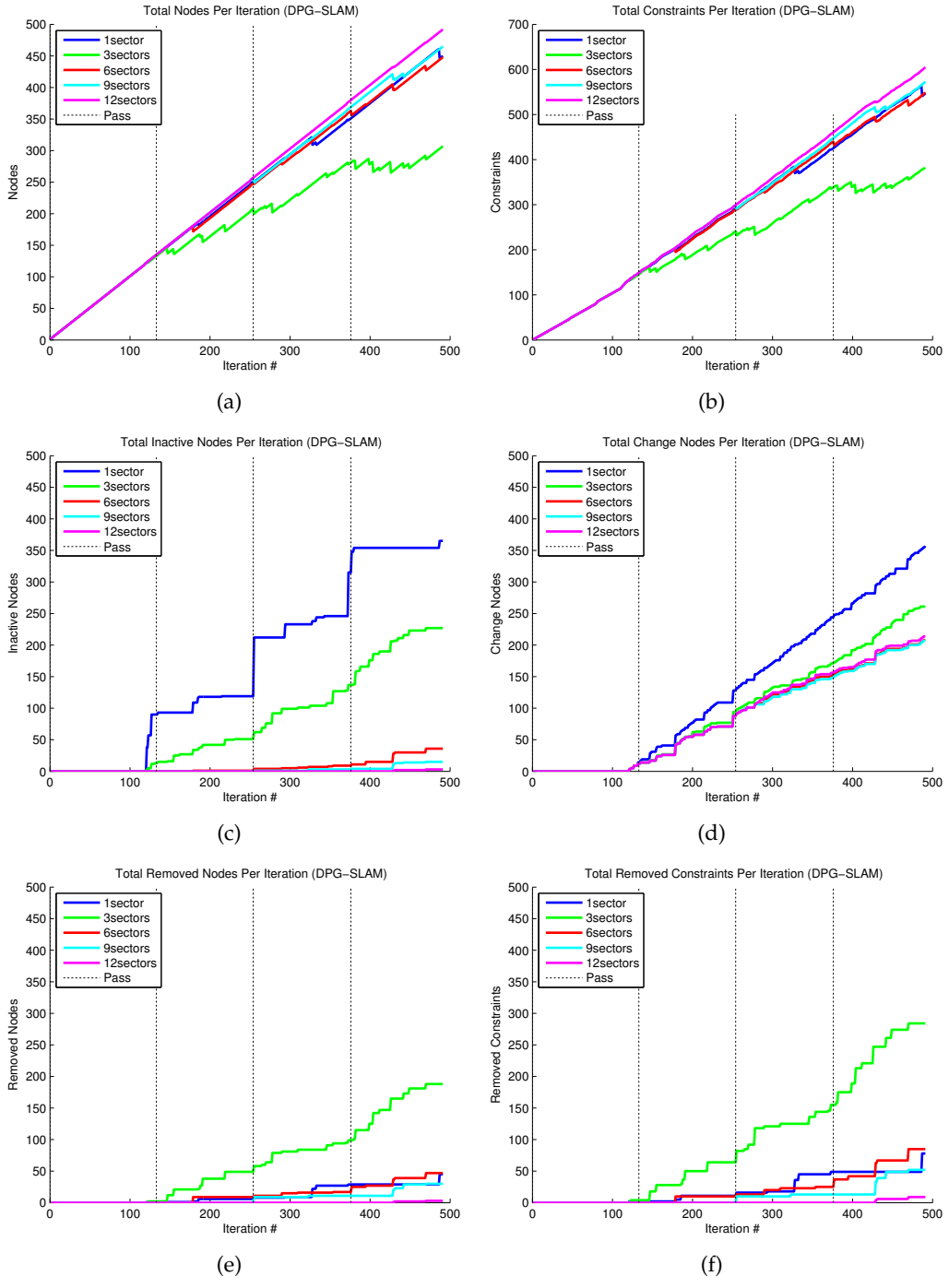


Figure 6-13: The graphs show how the growth of the number of constraints and each node type over all four passes where DPG-SLAM is applied.

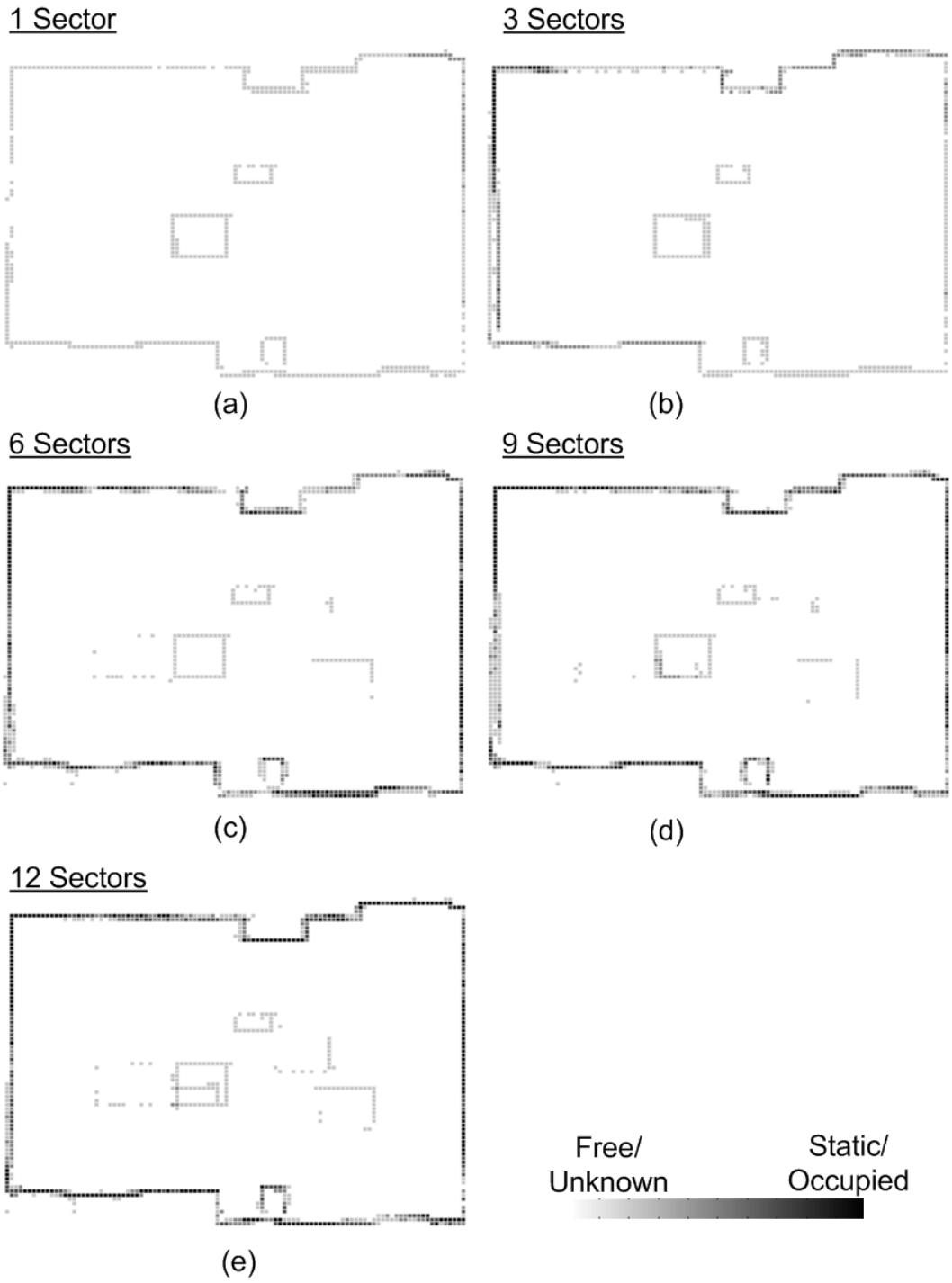


Figure 6-14: The static histograms generated from four passes through the CSAIL reading room and by applying DPG-SLAM to compute the active map.

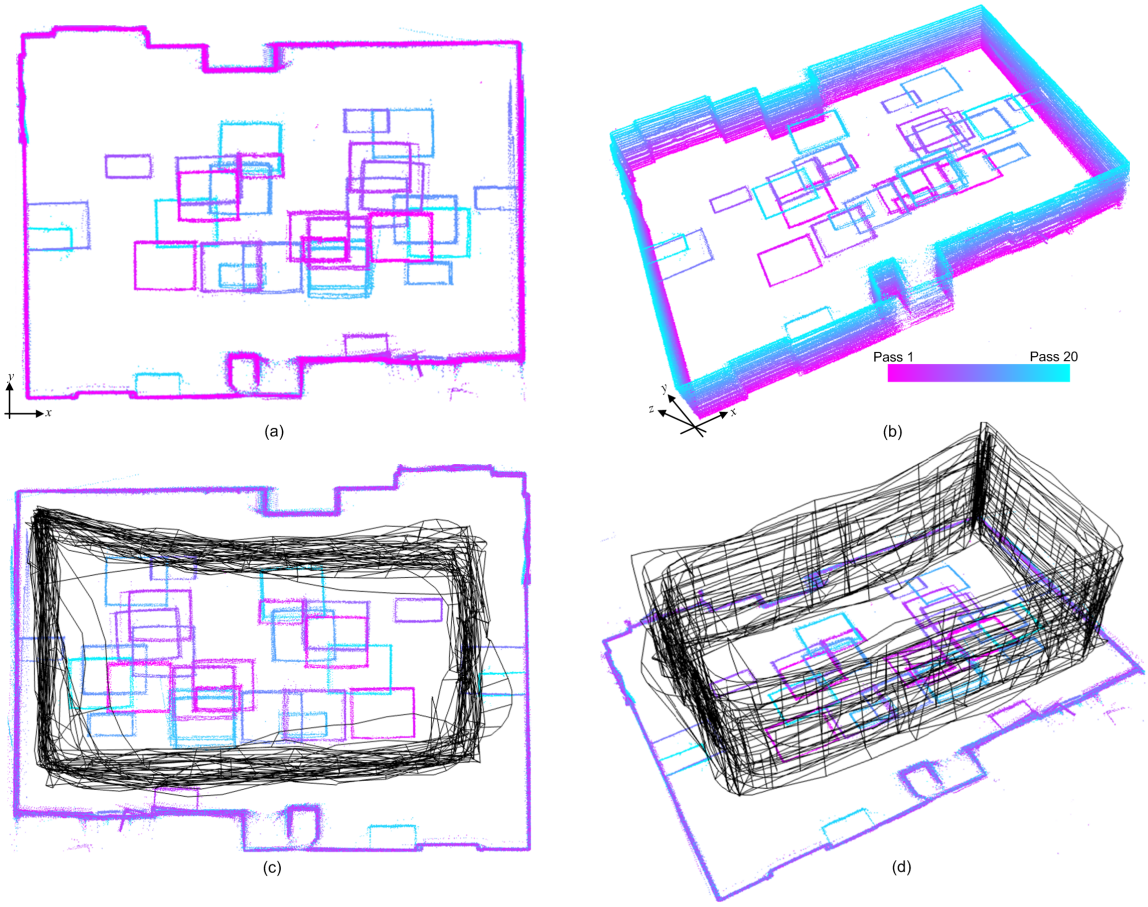


Figure 6-15: Images of the map generated from Pose Graph SLAM applied to the twenty passes through the CSAIL Reading Room. The sub-figures (a) and (c) illustrate an orthogonal projection of all the maps from each pass. Sub-figure (b) illustrate a 3-D side view of all twenty maps, and (d) depicts the edges from the resulting pose graph.

DPG-SLAM-NR Results

The active maps for 1, 3, 6, 9, and 12 sectors, where DPG-SLAM-NR is applied, are shown in Figure 6-16. In each of the maps there is a little ghosting from older ranges from low-dynamic objects that have moved. The 1 sector experiment has the least amount of ghosting and the amount of ghosting increases with the remaining number of sectors. Moreover, the amount of range points from static objects from several from several of the passes, increases with the addition of more sectors.

Table 6.5 summarizes the total number of change and inactive nodes. Figure 6-17 show the graphs of each node type and the constraints over the twenty passes. The number of change nodes are comparable for three or more sectors. The number of change nodes in 1 sector is much greater. This is due to the large number of inactive nodes, which leaves fewer active nodes that can be used for change detection. Also, the number of inactive nodes reduces while the number of sectors increases. It takes longer for a node with several sectors to become inactive because all of its sector have to be affected by changes.

Table 6.5: Total number of change nodes and inactive nodes for each experiment after the robot makes twenty passes through the CSAIL Reading Room, where DPG-SLAM-NR is applied. A total of 2,468 nodes and 3,158 constraints were added in each experiment.

Sectors	Change	Inactive
1	2,306	2,211
3	1,693	1,941
6	1,418	1,137
9	1,373	460
12	1,356	209

The static histograms generated from the active maps are shown in Figure 6-17. The 1 sector static histogram has a very faint outline of the walls. Whereas, nine sectors and twelve sectors have a clear outline of the walls. There are also faint traces form the ghosting in each of the static histograms.

DPG-SLAM Results

Figure 6-18 shows the images of the active maps generated by applying DPG-SLAM. Even though nodes and constraints were removed, the maps are visibly similar the DPG-SLAM-NR active maps. A summary of the totals of each type of node and total constraints is given in Table 6.6. In addition, three and six sectors contain the largest number of removed constraints. The six sectors experiment resulted in the smallest DPG, with 1,422 nodes and 1,639 constraints. Also the number of change nodes are similar, except that 1 sector contains much more change nodes. Again, the number of inactive nodes decreases with the increase in the number of sectors.

Figure 6-20 shows the totals as they increase with each iteration. The number of constraints for one and twelve sectors are shown to grow similarly (see Figure 6-20b). Although 1 sector experiments contains the greatest number of total inactive nodes, it has the fewest number of removed constraints. This is a result of when node removal is performed during DPG-SLAM. Specifically with 1 sector scenarios, the DPG nodes are highly sensitive to changes and become inactive quickly. As a result, the pose chains for removing

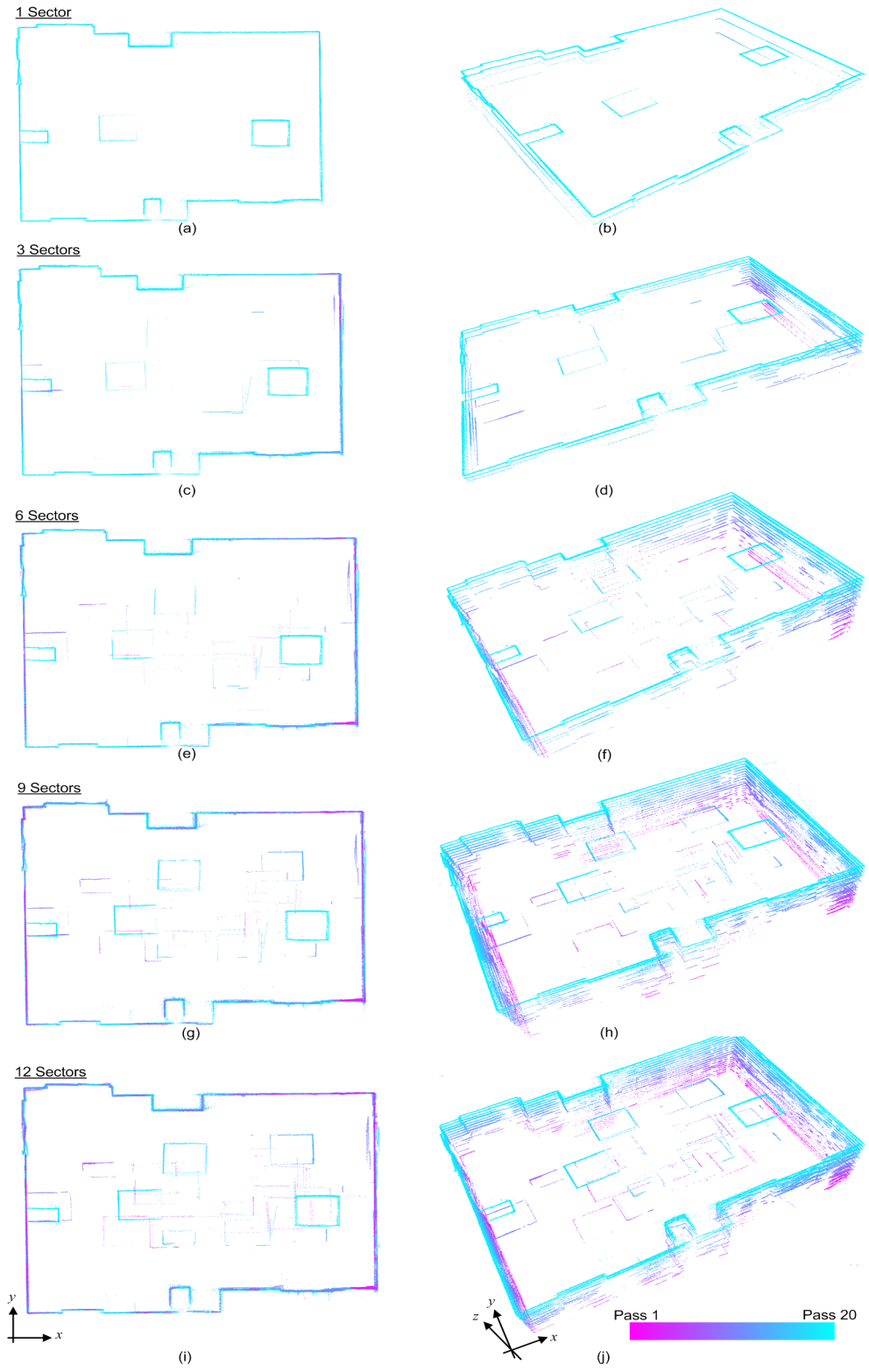


Figure 6-16: The active map generated from DPG-SLAM with no node/edge removed.

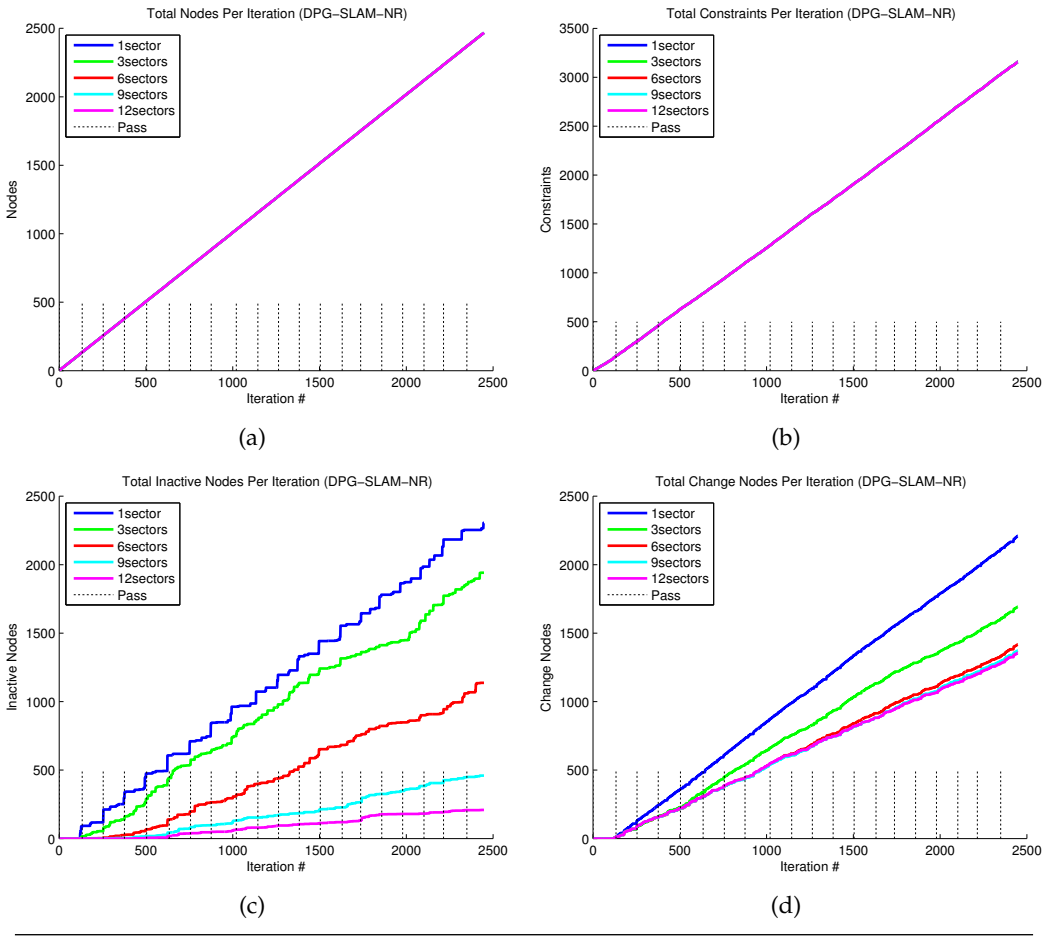


Figure 6-17: DPG-SLAM-NR graph depicting the rate of growth of the number of constraints and nodes as the number of passes increases.

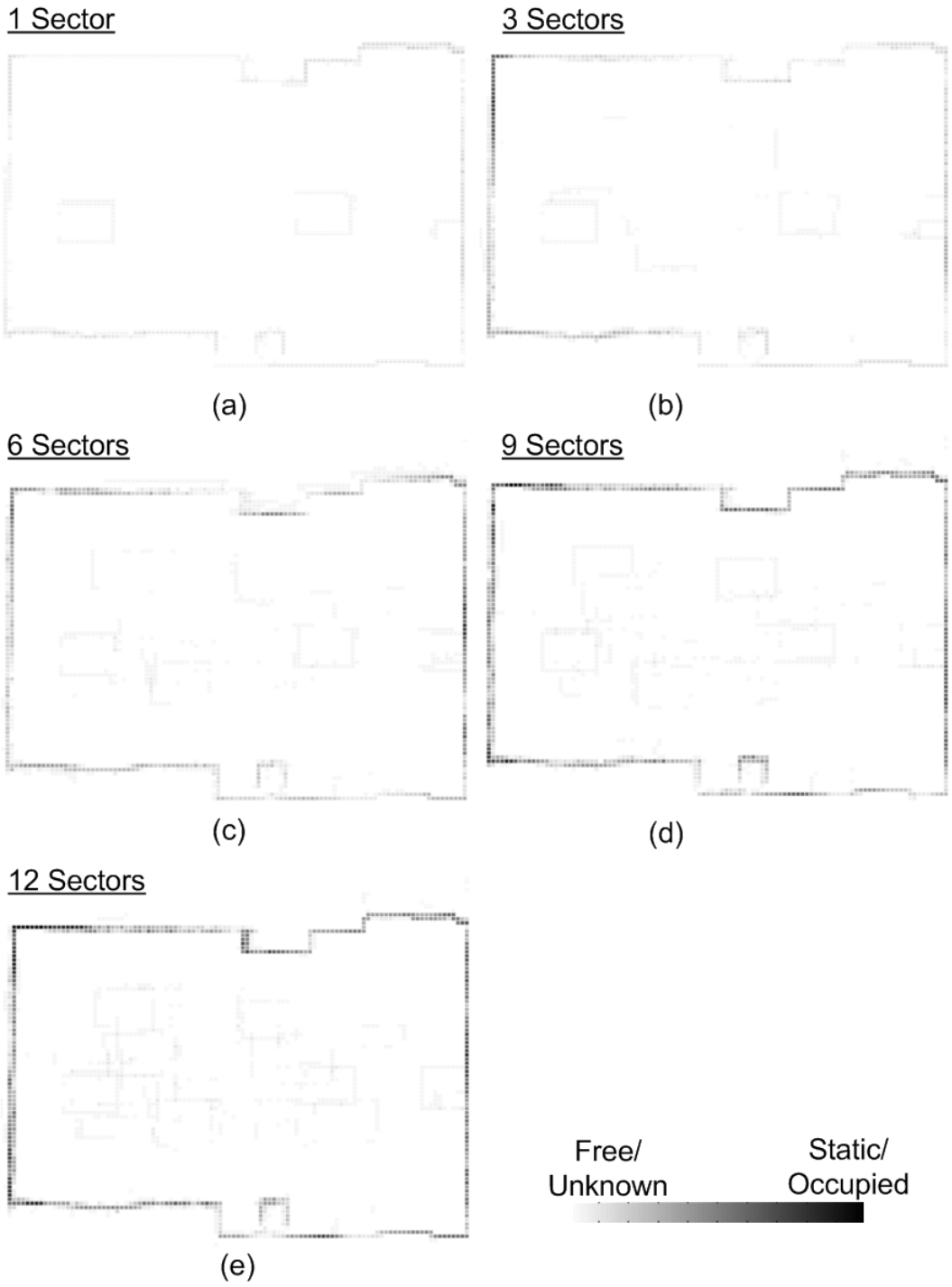


Figure 6-18: The static histograms generated from twenty passes through the CSAIL Reading Room. The images are generated from the active maps where DPG-SLAM-NR was applied.

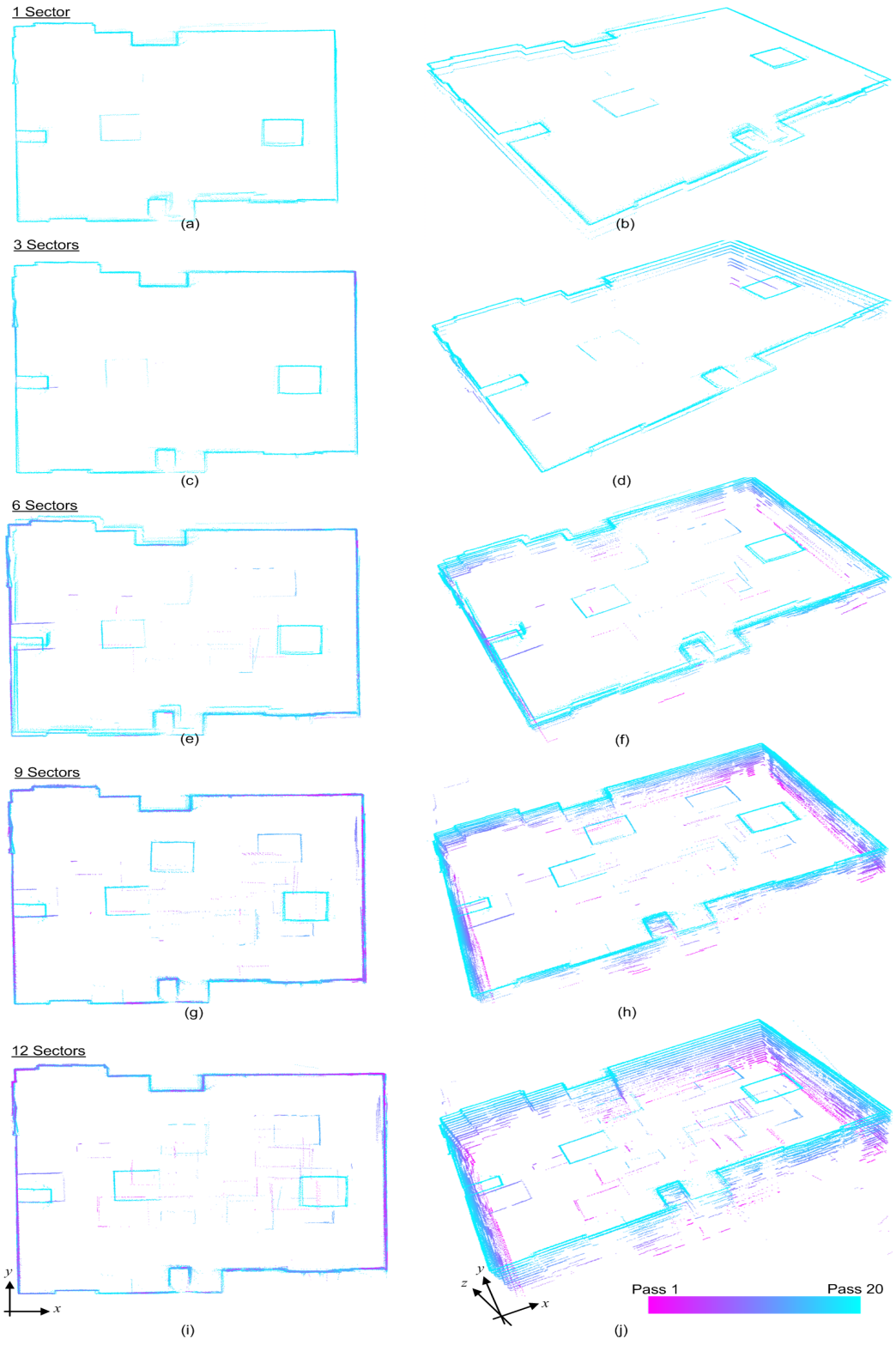


Figure 6-19: The active map generated from DPG-SLAM with no node/edge removed.

inactive nodes are often as long as all the nodes in a pass. One of our criteria for removing nodes is based on the length of the pose chain so that we can maintain a certain level of coverage as well as constraints. In the case of 1 sector, where most or all nodes in a sequence referring to a specific pass are inactive, these nodes are not removed.

Table 6.6: Total for each type of node and constraint after the robot makes 20 passes through the CSAIL Reading Room and DPG-SLAM was applied.

Sectors	Nodes	Constraints	Change	Inactive	Rem Nodes	Rem Constraints
1	2,086	2,681	2,259	2,287	382	574
3	1,452	2,239	1,736	1,772	1,016	1,016
6	1,422	1,639	1,378	814	1,046	1,833
9	1,920	2,304	1,374	325	548	951
12	2,087	2,599	1,359	169	381	656

Finally, Figure 6-21 shows the static histograms for each DPG-SLAM experiment. Like the DPG-SLAM-NR static histograms, the nine and twelve sectors yield dark outlines of the walls. This is a desired result because unlike DPG-SLAM-NR, the size of the DPG has been reduced. Consequently, improving the overall computational cost of pose graph optimization applied to this DPG.

6.4 University of Tubingen Experimental Analysis and Results

In this section we present short-term and long-term results from the Univ. of Tubingen Robot Lab data set. The short-term operation experiments consists of data from four days, where the robot made four passes through the robot lab. The long-term operation experiments covered approximately five weeks, where the experiments included data from sixty passes over the five week time period. At the completion of the four passes the robot traveled 620 meters, and upon completion of the sixty passes the robot traveled a total of 7.4km.

To begin, we show a summary of the accuracy and efficiency of DPG-SLAM and DPG-SLAM-NR after 60 passes. Then we present the results from short-term and long-term operation. The short-term experiments include nodes with 1,4,8, and 12 sectors. The long-term experiments include nodes with 4, 8, 12, and 16 sectors. From our experimental observations, the 1 sector experiments were very sensitive to change and yielded active maps with minimal density (ie. included ranges primarily from one or very few passes). In addition, the experiments depict active maps that are updated when 30% change is detected.

6.4.1 Summary Comparison of DPG-SLAM and DPG-SLAM-NR, 60 Passes

Figure 6-22 shows the images resulting from applying DPG-SLAM (on the right) and DPG-SLAM-NR (on the right) to 60 passes through the Univ. of Tubingen Robot Lab, with 8 sectors per node. The parts of the images that are shown in magenta imply that data from earlier passes remains in the active map. This is shown along the "T" intersection in the middle. In addition, magenta points can be seen along the top right corridor where posters were installed (highlighted in Figure 6-2c). These points remain in the map because the installation is not removed and thus, the points beyond the installation are *unknown*, i.e.

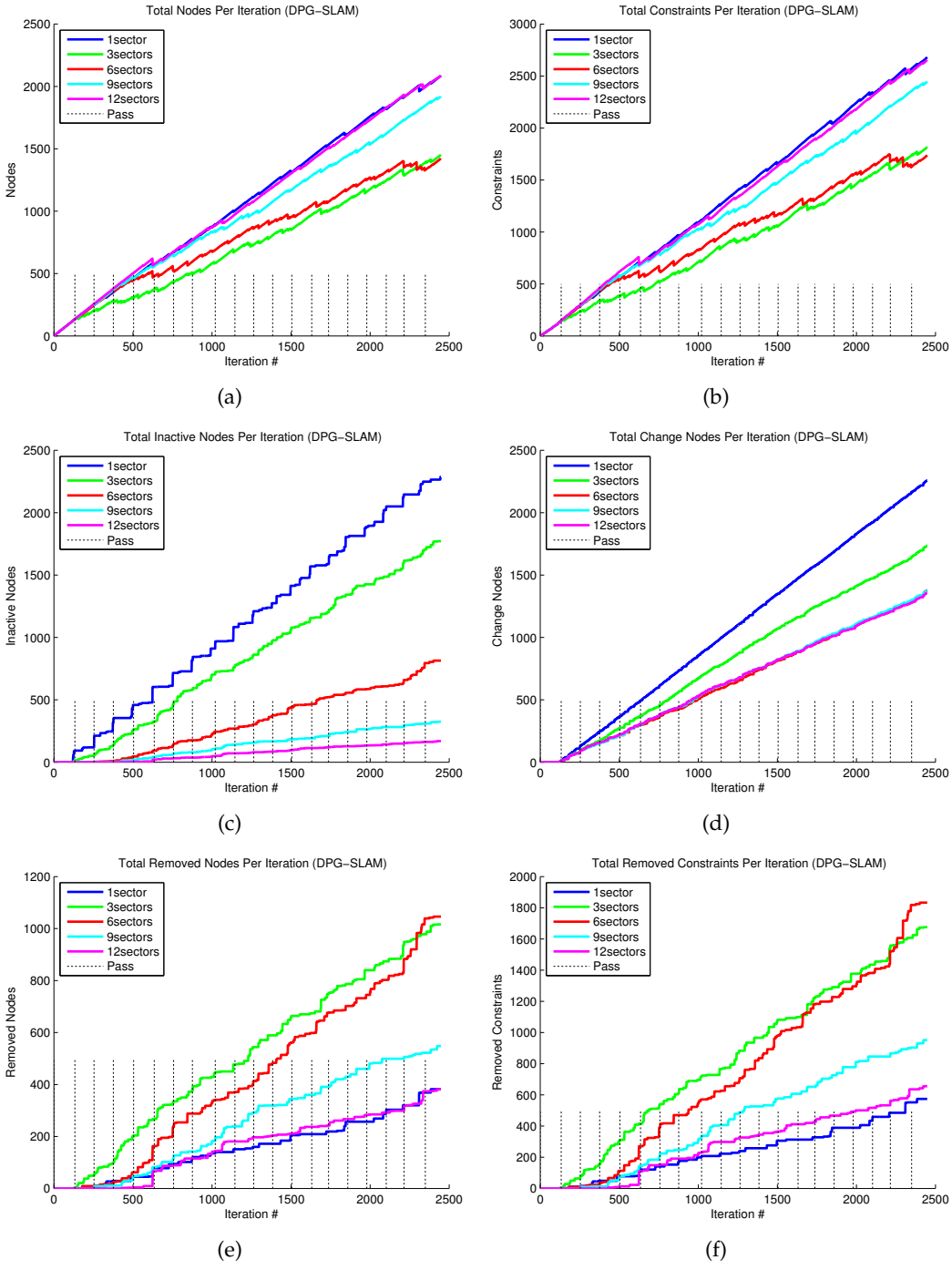


Figure 6-20: The graphs illustrate the rate of growth of the number of constraints and nodes as the number of passes increases (DPG-SLAM was applied).

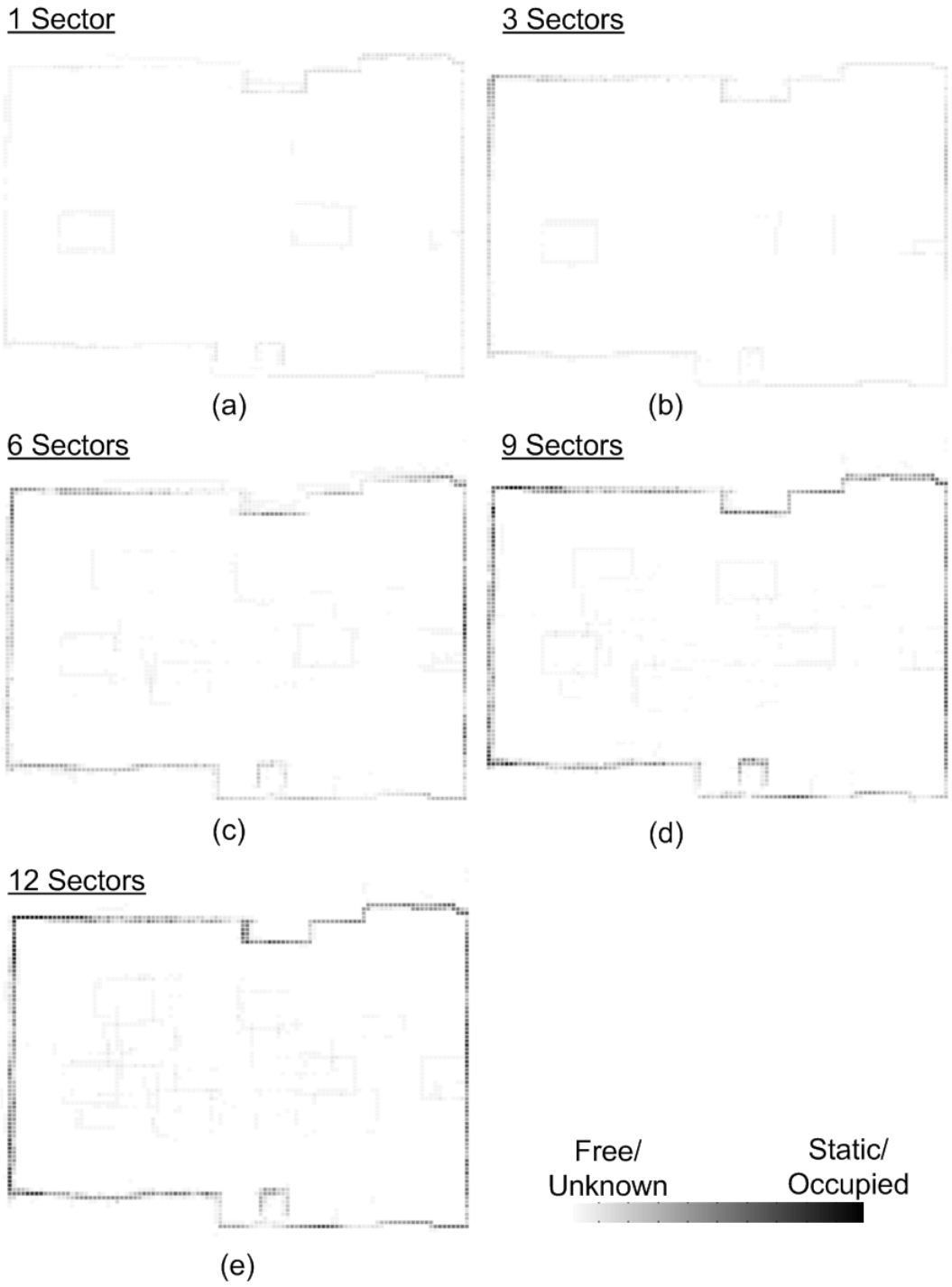


Figure 6-21: The static histograms generated after the robot makes 20 passes through the CSAIL Reading Room and applying DPG-SLAM to maintain its active map.

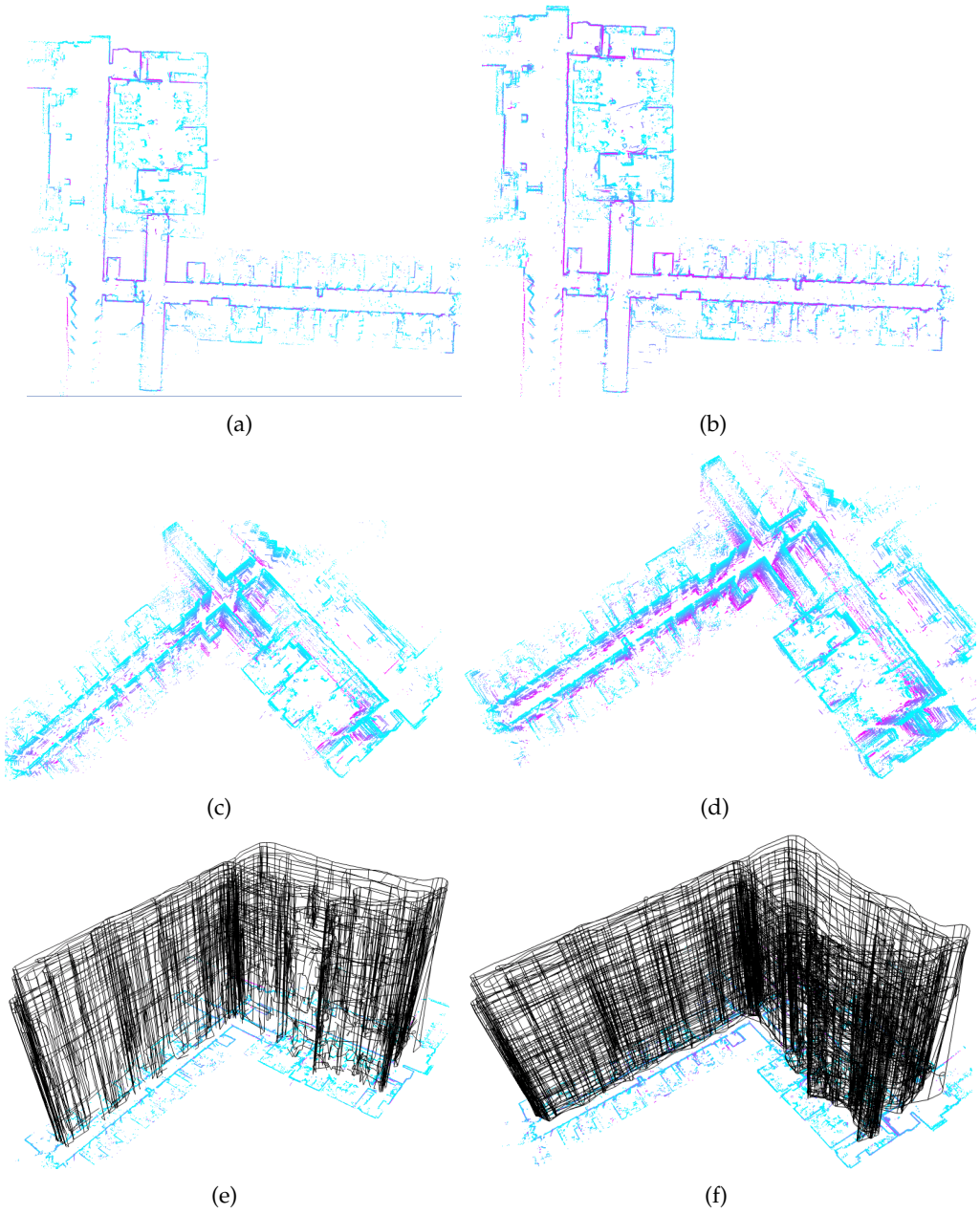


Figure 6-22: On the left are the active, DPG, and dynamic maps for the DPG-SLAM algorithm and on the right are the same images for the DPG-SLAM-NR algorithm. In this experiment each node had 8 sectors.

outside the robot's FOV. Again, the density of static points from the walls is greater for the DPG-SLAM-NR than the DPG-SLAM.The size of the DPG is significantly reduced for the DPG-SLAM algorithm. There are a total of 4,511 nodes and 5,809 constraints for the DPG-SLAM algorithm, and a total of 8,392 nodes and 11,350 constraints resulting from the DPG-SLAM-NR algorithm. In the images, the active maps from both algorithms are very similar even though the size of the DPG for the DPG-SLAM algorithm is nearly half the size of the DPG from the DPG-SLAM-NR algorithm

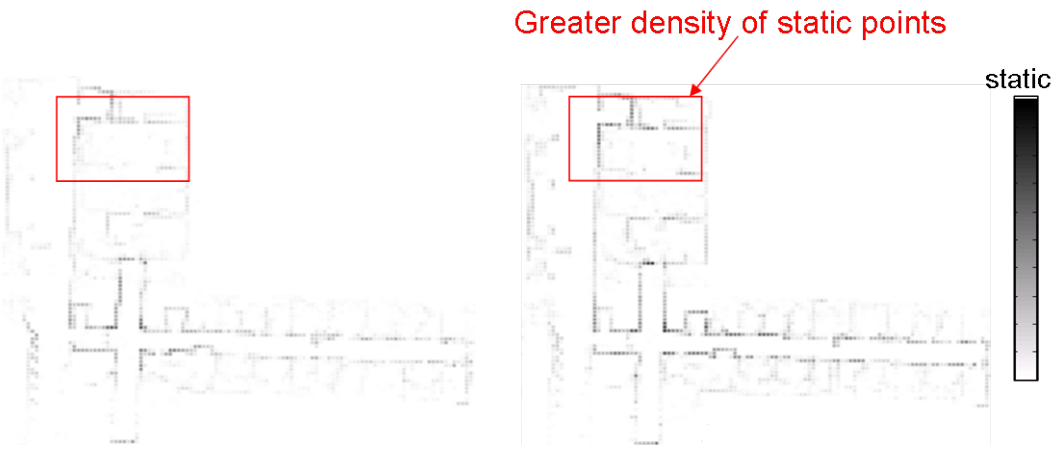


Figure 6-23: Static histograms of DPG-SLAM (on the left) and DPG-SLAM-NR (on the right) active maps from 60 passes through the University of Tubingen Robot Lab. The density of static map points is shown as being darker in the images.

The static histograms of both algorithms are shown in Figure 6-23, and the run time ratio is shown in Figure 6-24. Clearly the active map from the DPG-SLAM algorithm was affected by removing nodes and constraints. However, the run time increased significantly with each iteration.

6.4.2 Short-term Operation, Four Passes

Figure 6-26(a) and Figure 6-26 depict the map after applying Pose Graph SLAM to the Univ. of Tubingen four passes data set. The resulting pose graph contains 833 nodes and 1,010 constraints.

DPG-SLAM-NR Results

The individual passes used in the short-term experiments are shown in Figure 6-2. The active map for each of the experiments is given in Figure 6-27 and we describe a few key observations with the results. The added objects labeled in Figure 6-2c are correctly not included in the active maps. Remnants of the moveable partition (Figure 6-2d) are evident mainly in the twelve sectors active map. Lastly, the poster installation which was added after pass two remained in the active maps. This is because the posters blocked the view of the robot and thus there was no way to determine if there was a change that occurred behind the posters.

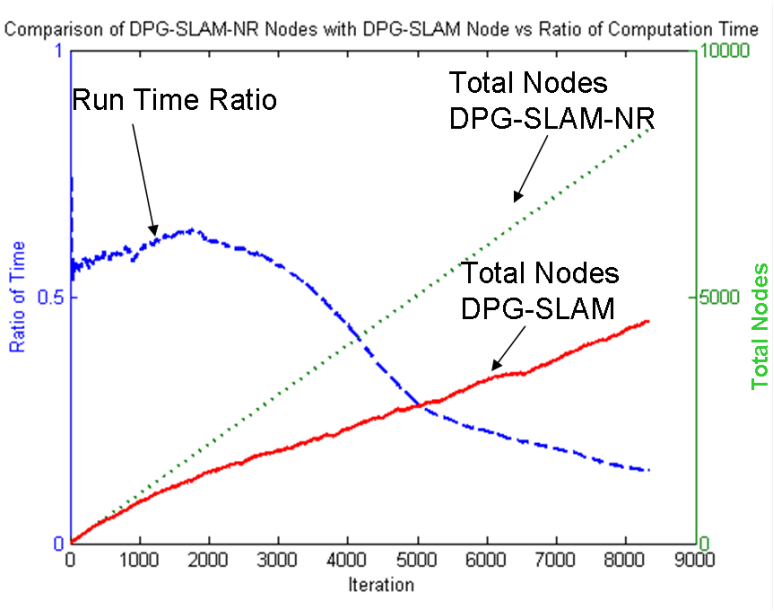


Figure 6-24: Run time ratio of 60 passes through the Univ. of Tubingen.

Table 6.7: Total number of change nodes and inactive nodes after the robot makes four passes through the Univ. of Tubingen Robot Lab. A total of 833 nodes and 1,010 constraints were added to each DPG and DPG-SLAM-NR was applied.

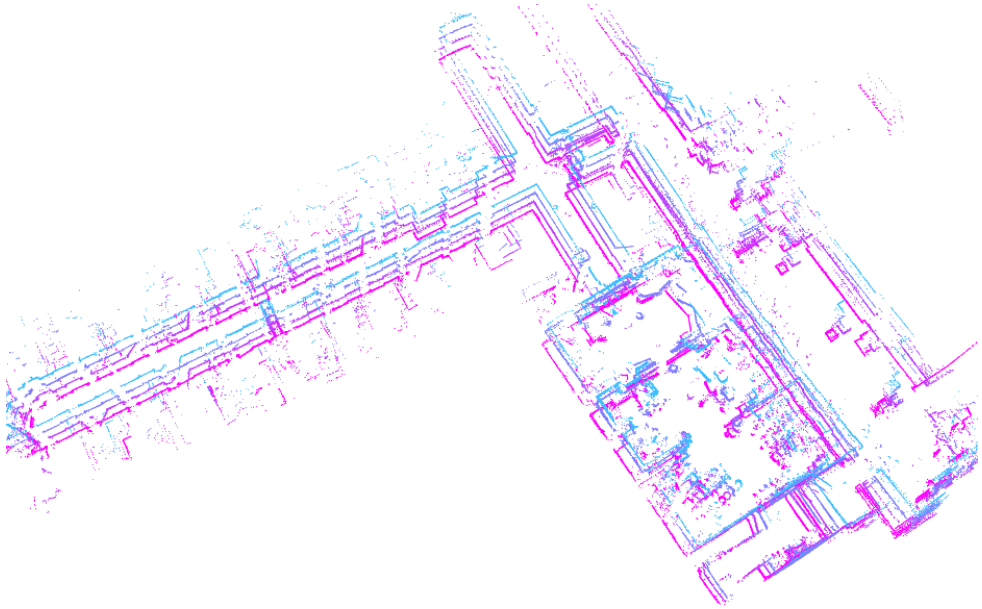
Sectors	Change	Inactive
1	494	698
4	378	387
8	363	134
12	357	54

Table 6.7 depicts the totals for change and inactive nodes according to the number of sectors. Figure 6-28 shows how the number of each type of node and number of constraints increase with each pass. The number of inactive nodes decreases as the number of sectors increases. This is expected as nodes with more sectors require changes to affect all of the sectors in order to be labeled inactive.

Moreover, the total number of change nodes resulting from the experiments with four or more sectors are approximately the same. Similar results, in terms of the number of change nodes, were also seen in the CSAIL Reading Room experiments. Also, the location of change nodes does not vary much as each DPG is created, and nodes are in similar positions in each of the DPGs. Thus, nodes from each experiment that are in areas where changes are detected are labeled change nodes.

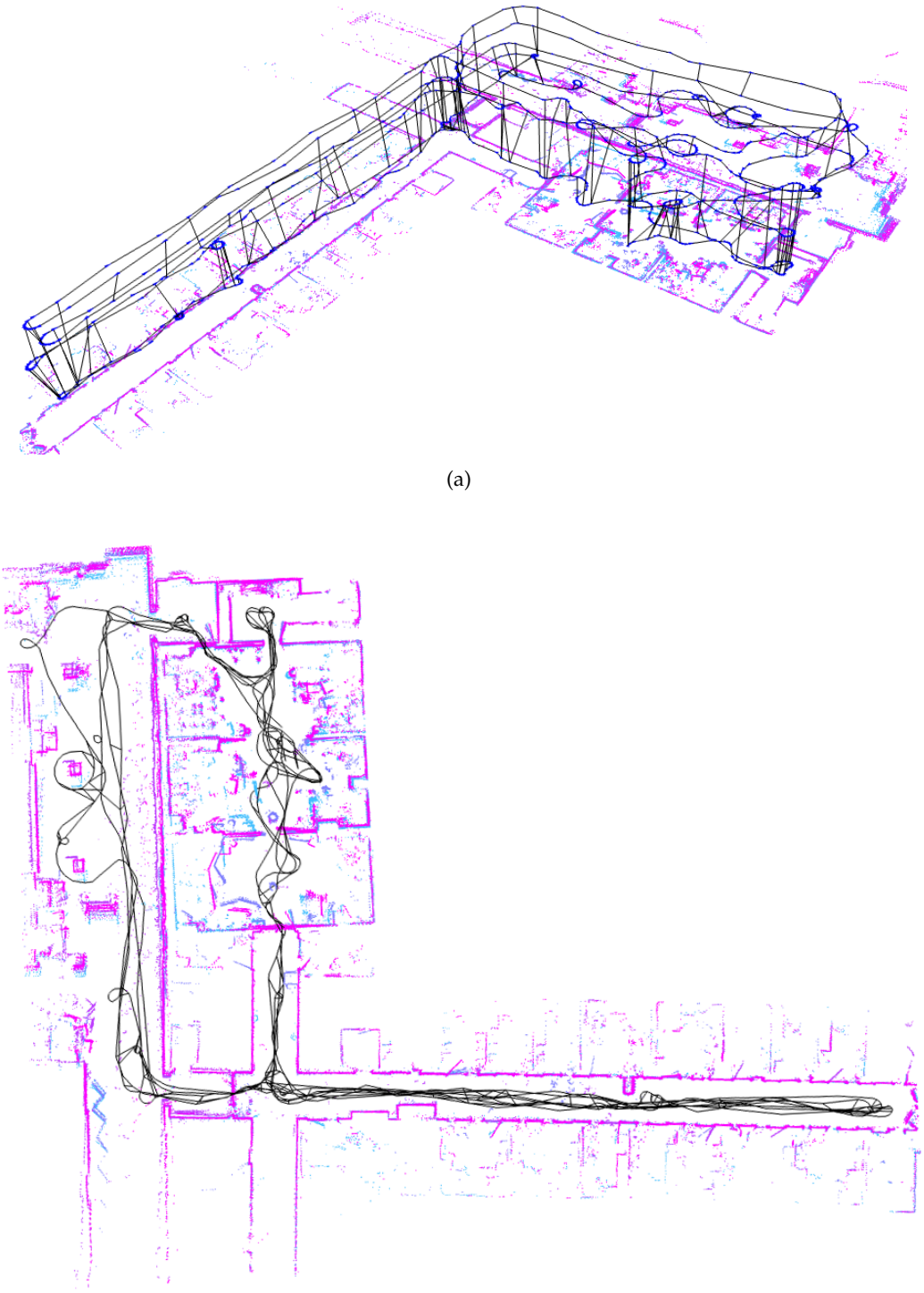
Finally, the static histograms for four passes through the Robot Lab are shown in Figure 6-29. By inspection both the eight sectors and twelve sectors yield sufficient and accurate results in terms of denoting the static areas. Beginning with the four sectors static histogram there are dark areas that increase in intensity and static areas with the eight and twelve passes.

(a)



(b)

Figure 6-25: Examples of the orthogonal projection, (a), and three-dimensional side-view, (b), of the pose graph SLAM map generated from the four passes in the short-term experiments. Note, some of the changes from this data set are hand-labeled in Figure 6-2.



(a)

(b)

Figure 6-26: In Figure 6-26(a) the pose graph is shown and the height of each edge is depicted as a function of time. Figure 6-26(b) is an image of the orthogonal projection with the pose graph edges shown.

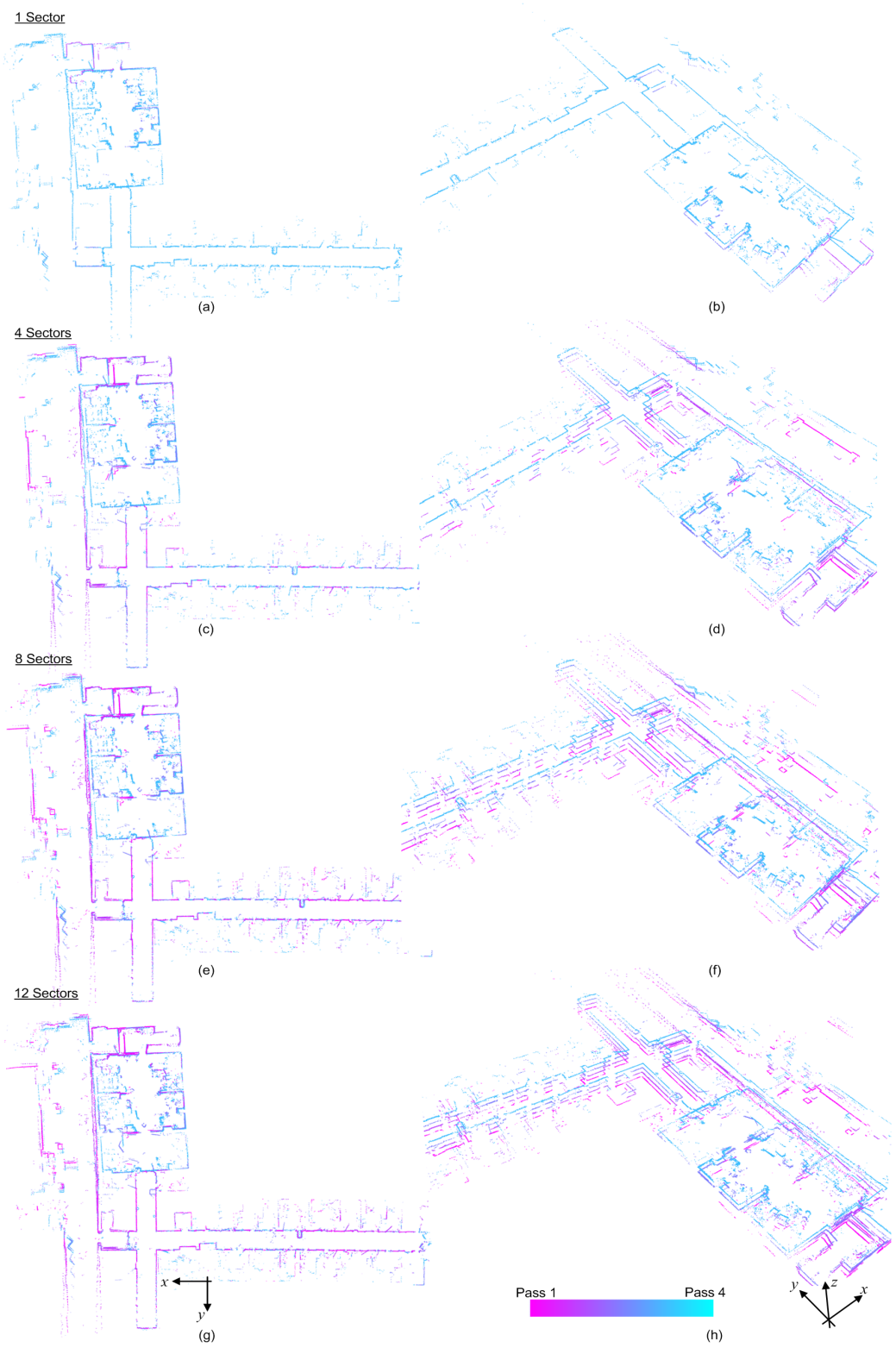


Figure 6-27: Images of the active maps where DPG-SLAM-NR was applied to four passes through the University of Tubingen Robot Lab.

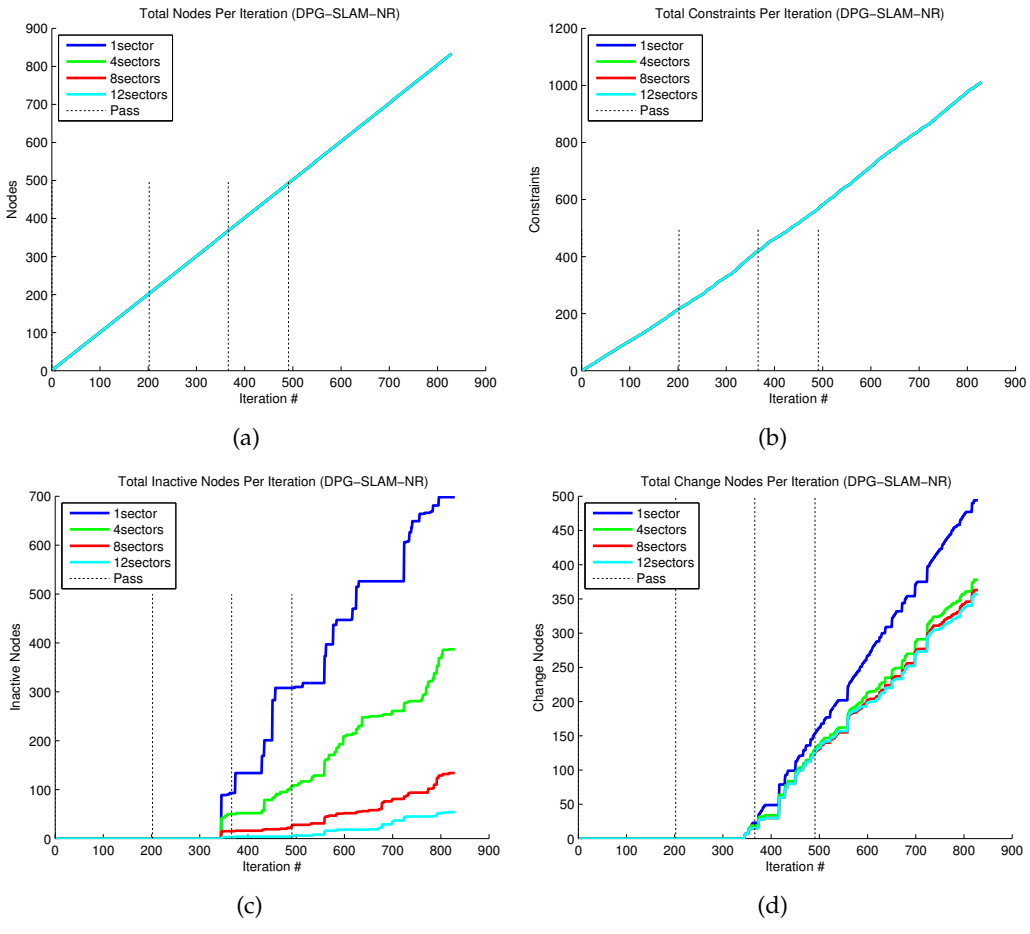


Figure 6-28: Each graph shows the number of nodes, constraints, inactive nodes, and change nodes for the 1, 4, 8, and 12 sector DPG-SLAM-NR experiments.

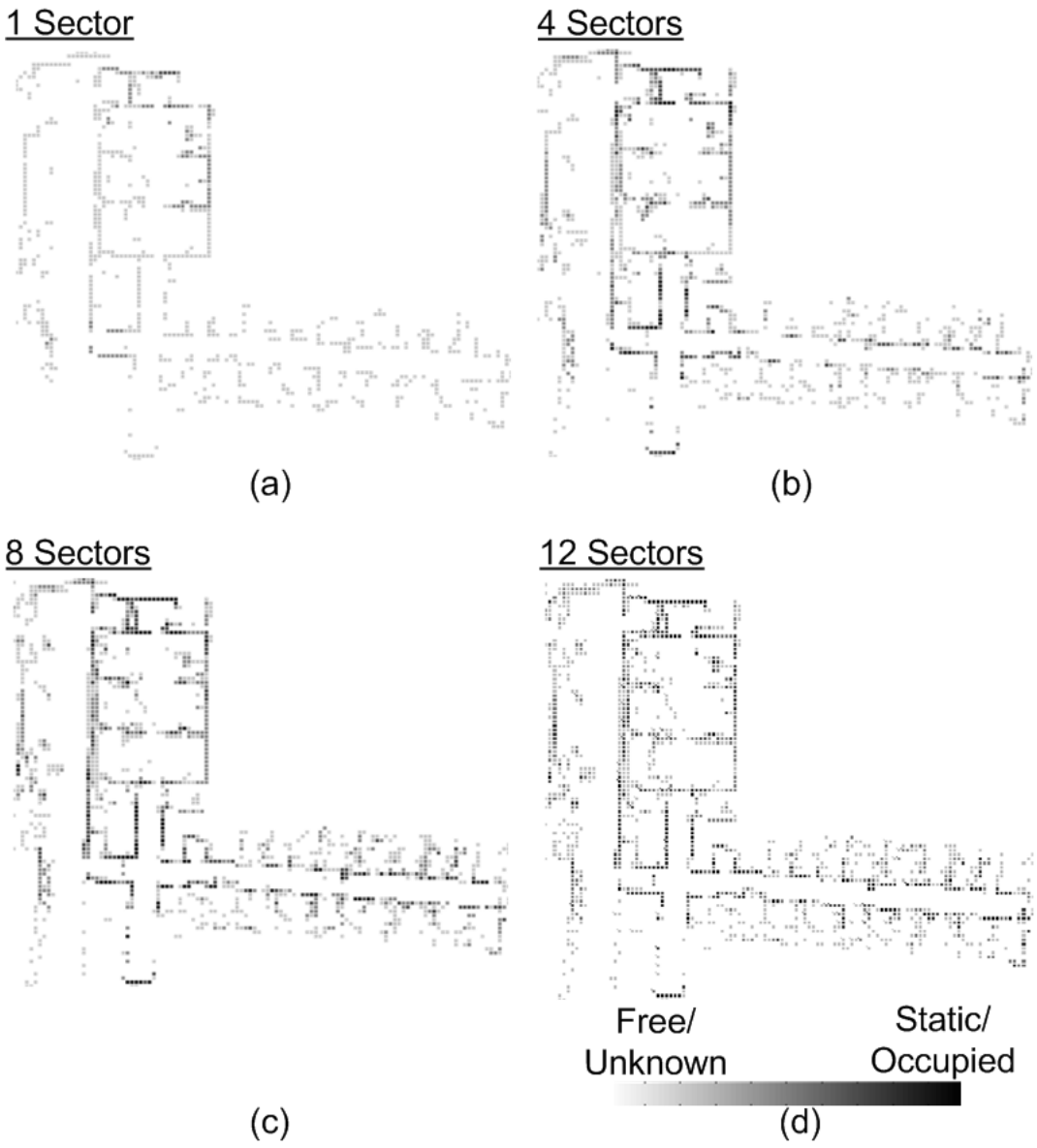


Figure 6-29: The static histograms for the DPG-SLAM-NR 1, 4, 8, and 12 sector experiments. The 8 and 12 sector experiments clearly indicate the walls, except along the left-most corridor where the walls are at half intensity.

DPG-SLAM Results

The active maps generated by DPG-SLAM are shown in Figure 6-30. Due to edge and node removal, pose estimates can change slightly and possibly become less accurate. For example, there is a minor visible difference between the four sectors active map with DPG-SLAM (Figure 6-30c), and the four sectors active map with DPG-SLAM-NR (Figure 6-27). While the remaining DPG-SLAM active maps are comparable to their DPG-SLAM-NR maps.

The totals of each type of nodes and the number of constraints at the completion of four passes through the Univ. of Tubingen Robot Lab using DPG-SLAM are given Table 6.8 and Figure 6-31. It is clear that 1 sector experiments contain the greatest number of inactive nodes. However, 1 sector has the least removed nodes and constraints. As a result, 1 sector has the largest DPG compared to 4, 8, and 12 sectors. Intuitively, the number of inactive nodes should be close to the number of removed nodes. As mentioned earlier 1 sector scenarios are particularly sensitive to potentially having an entire pass removed, and thus the pose chain is too long for removal. More specifically, all nodes in a pass become inactive in the 1 sector case. Given our criteria, we do not remove all nodes and edges from a pass, as a result these inactive nodes remain in the DPG.

Table 6.8: Totals for each type of node after the robot makes four passes through the University of Tubingen Robot Lab DPG-SLAM was applied.

Sectors	Nodes	Constraints	Change	Inactive	Rem Nodes	Rem Constraints
1	785	960	489	696	48	72
4	496	853	370	390	337	217
8	712	855	356	136	121	190
12	765	916	360	60	68	106

Moreover, the graphs in Figure 6-31(a) and 6-31(e) show that nodes begin to be removed during pass three. Figure 6-2a and b, illustrate why node removal does not begin until pass three. The notable changes between pass one and pass two is the added object referred to in Figure 6-2b. The object is added and no other low-dynamic objects are moved or removed, and the object does not cause a significant change; thus, the object remains in the map and nodes are not removed.

Finally, the static histograms in Figure 6-32 depict how well the walls are shown in the map. As in the DPG-SLAM-NR results, the walls along the left corridor only have small portions that are clearly static. The eight and twelve sector experiments yield the best static histograms.

6.4.3 Long-term Operations, Sixty Passes

In this section we present the long-term operation experiments to illustrate the efficacy of DPG-SLAM on data collected over five weeks in an indoor environment, the University of Tubingen Robot Lab [9]. Results for sixty passes over the time period are shown for both DPG-SLAM-NR and DPG-SLAM. The experiments included nodes with 4, 8, 12, and 16 sectors. The robot traveled a total of 7,373 meters. Figure 6-33 and Figure 6-34 show the resulting map and pose graph after applying Pose Graph SLAM using iSAM. The pose graph contains a total of 8,392 nodes and 11,350 constraints. The aim is to show that with a large number of edges being removed with DPG-SLAM the accuracy of active map as

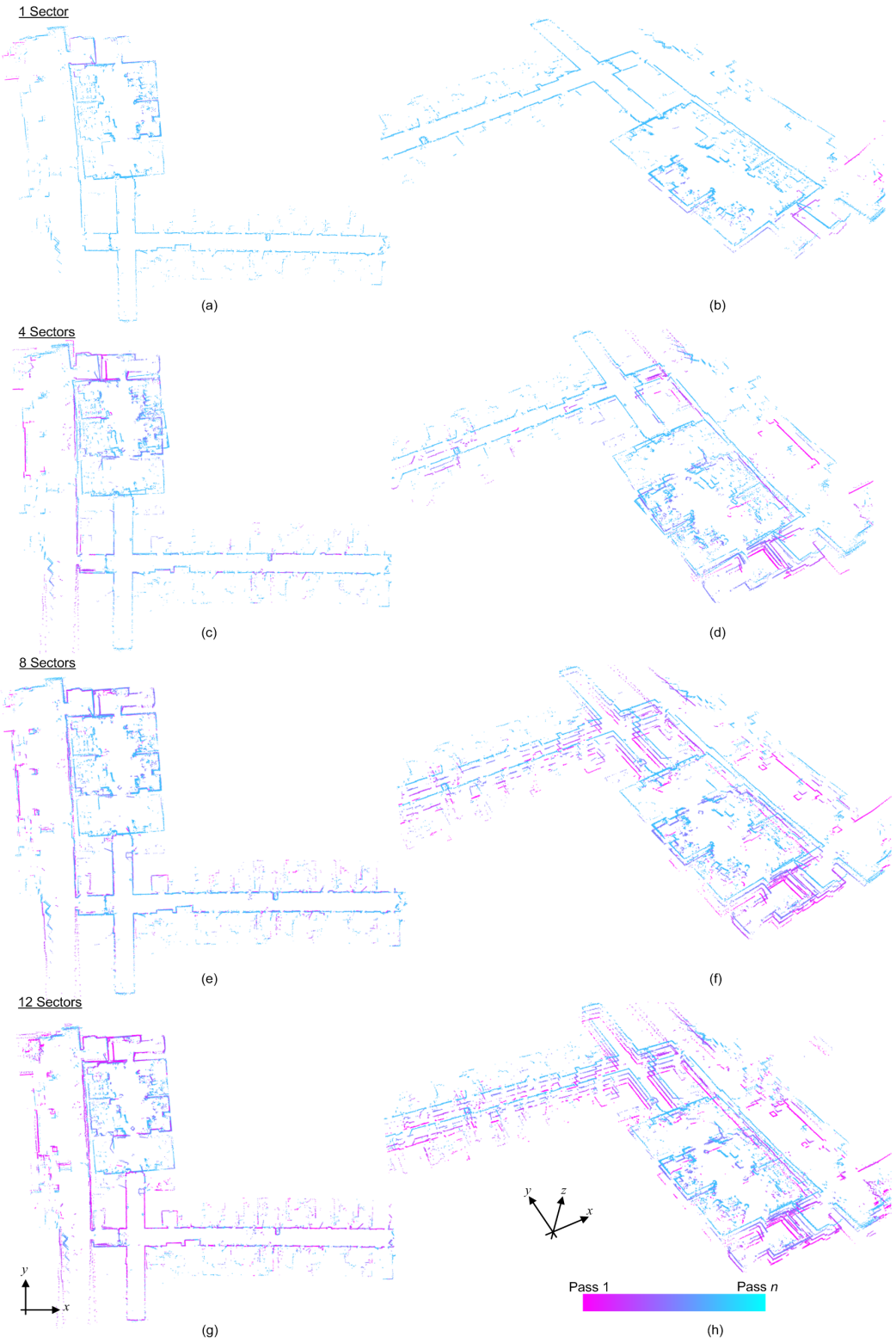


Figure 6-30: The active maps generated after applying DPG-SLAM where nodes and edges are removed.

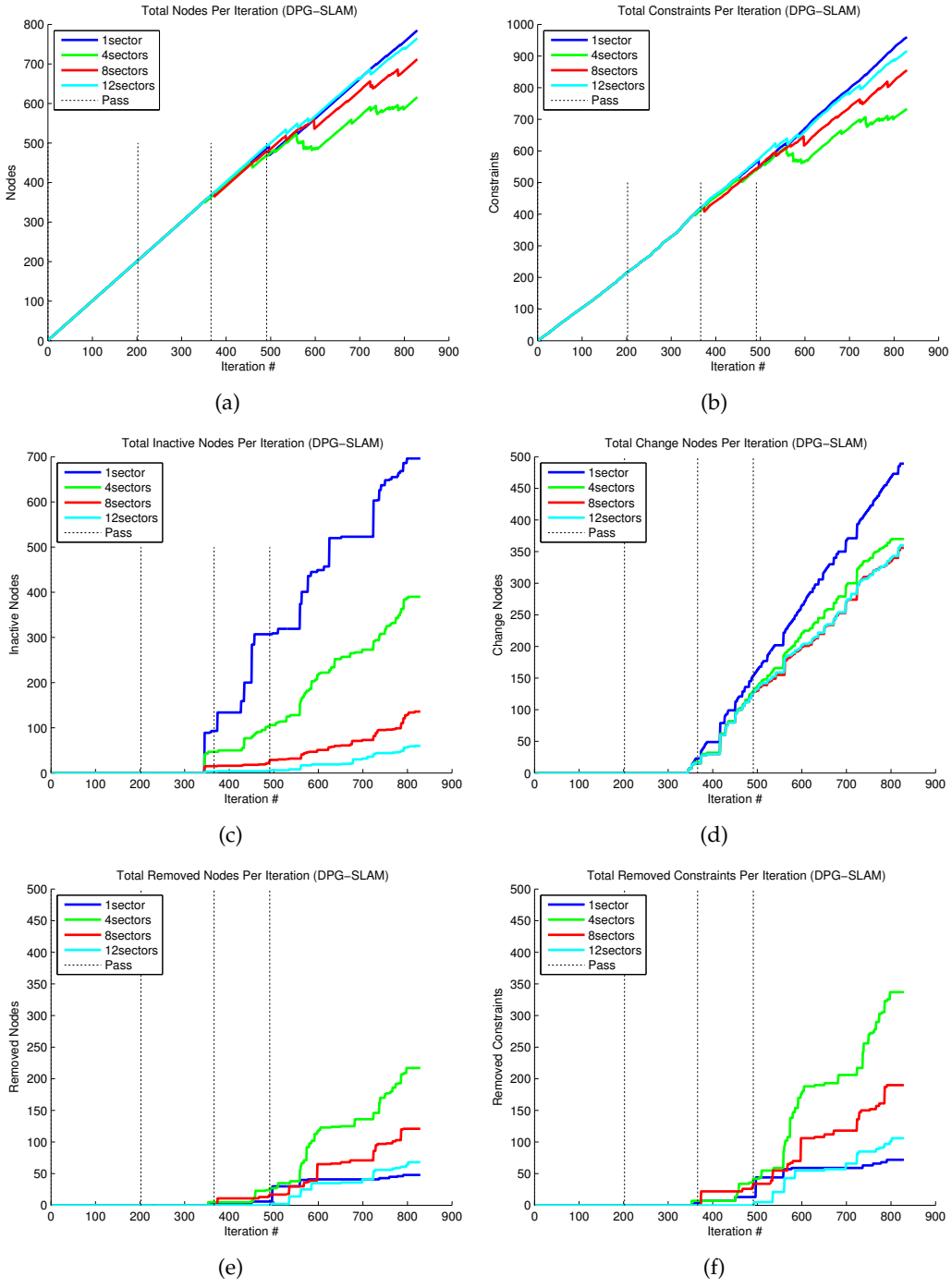


Figure 6-31: The graphs depict the growth of the number of constraints and node types over four passes from the Robot Lab, where DPG-SLAM is applied.

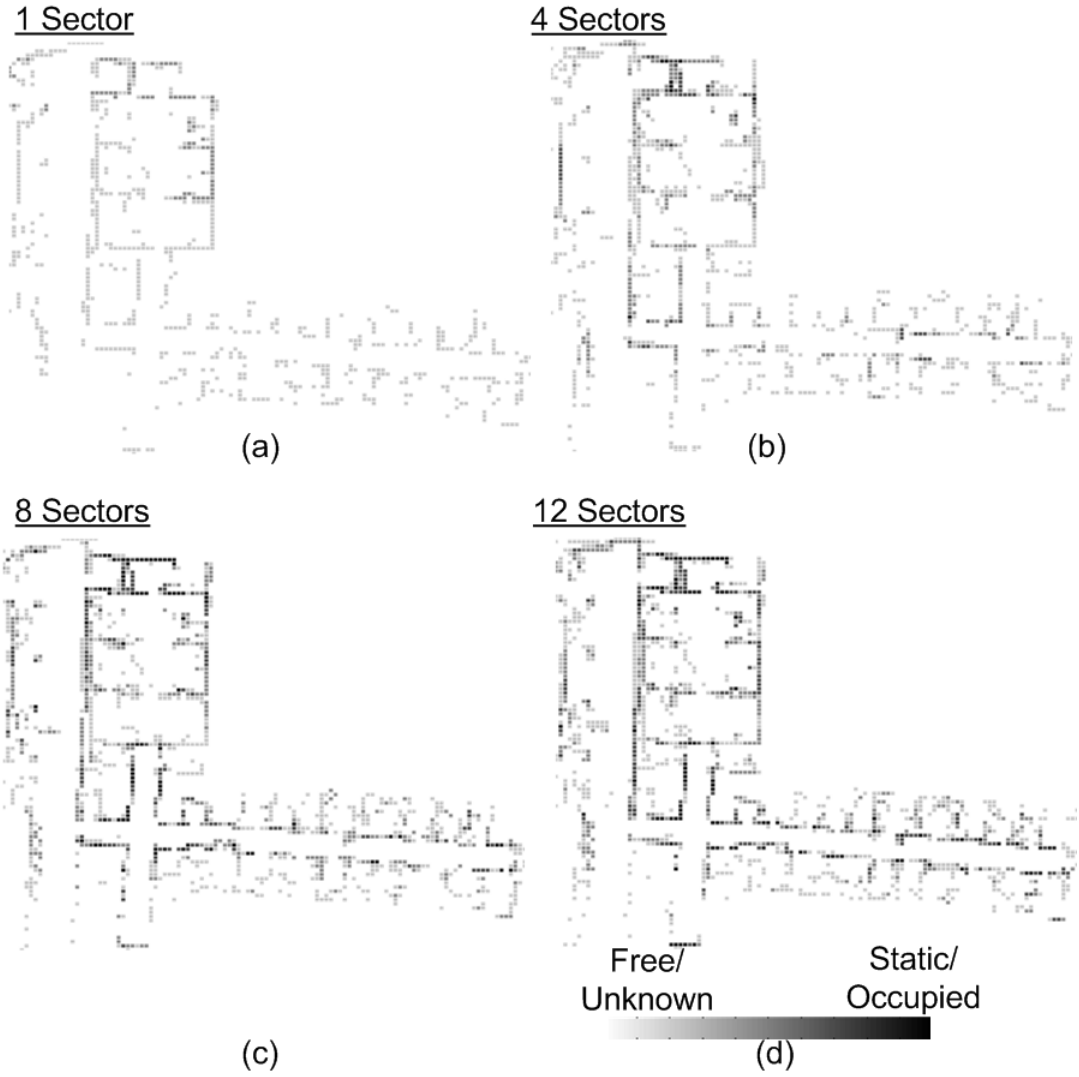


Figure 6-32: The static histograms of 1, 4, 8, and 12 sector experiments with four passes through the Univ. of Tubingen Robot Lab.

well as the static histogram is comparable to the results of DPG-SLAM-NR where nodes and constraints are not removed.

In Figure 6-33 and Figure 6-34 it is apparent that there are numerous spurious laser range points that derive from high-dynamic objects, such as people walking. As mentioned in Figure 6-2, the right-side corridor contains student offices and is a highly populated and trafficked office area. Recall that the percentage of change in these experiments is set to 30%. Therefore, the amount of spurious points has to cause a change greater than 30% in order to be filtered out of the active map. This is evident in the active maps generated by DPG-SLAM-NR and DPG-SLAM below. In general, existing SLAM methods that address the problem of SLAM in populated environments or SLAM and tracking would detect and filter out or track these high-dynamic objects [1, 3, 21, 32, 54, 66, 70, 73, 75]. In this work, we focus on the long-term unpredictable changes resulting from low-dynamic objects.

DPG-SLAM-NR Results

The DPG-SLAM active maps for 4, 8, 12, and 16 sectors is given in Figure 6-35. Both the twelve and sixteen sector active maps retain range points from most of the passes. While the four and the eight sectors have mainly ranges from the most recent passes (all of the earlier passes shown in a lighter magenta color, and are at the bottom of the 3D side-view maps). It makes sense as the many spurious points would quickly cause the sectors to be turned off and thus the node would become inactive quickly (particularly in the highly populated right-corridor).

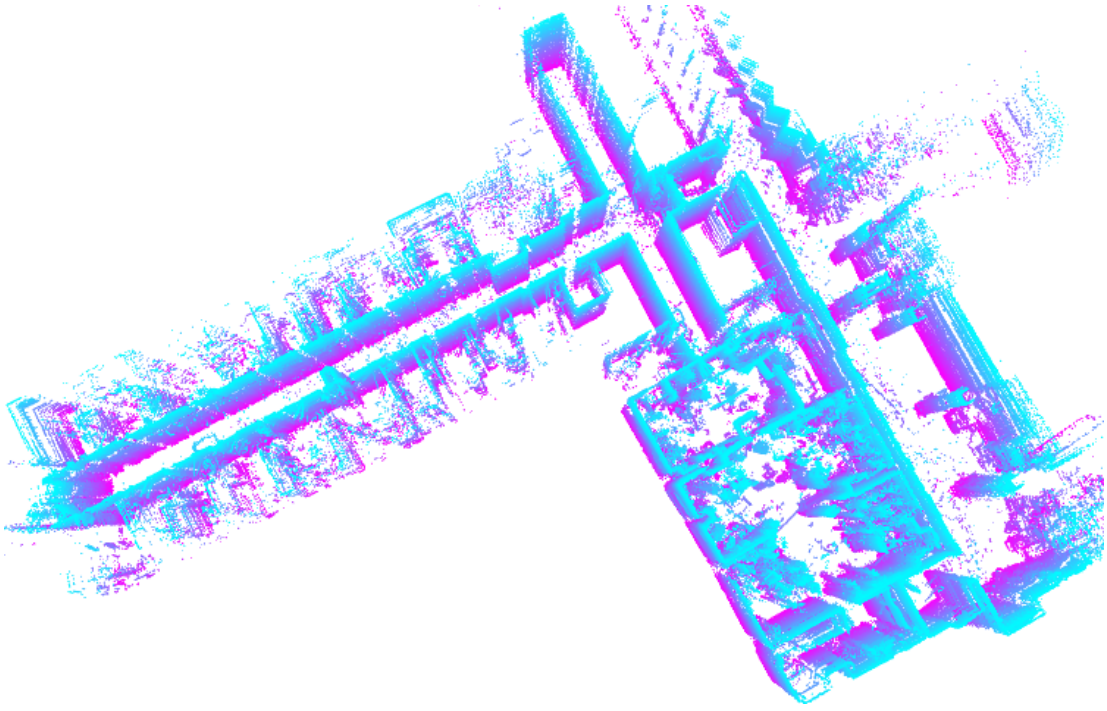
The total number of change nodes and inactive nodes at the completion of the sixty passes is given in Table 6.9. The number of change nodes are comparable for 8, 12, and 16 sectors, and are slightly greater for 4 sectors. In addition, as expected, the number of inactive nodes decreases as the number of sectors increase. Figure 6-36 shows the rate at which the number of constraints and each type of nodes grows with each iteration and pass. For example, the rate of grown for the number of change nodes (Figure 6-36(d)) is quite similar in all the DPG-SLAM-NR experiments.

Table 6.9: Total number of change nodes and inactive nodes after the robot makes sixty passes through the Univ. of Tubingen Robot Lab and DPG-SLAM-NR was applied. A total of 8,392 nodes and 11,350 constraints were added to each of the 4, 8, 12, and 16 sectors DPGs.

Sectors	Change	Inactive
4	5,019	6,604
8	4,577	4,513
12	4,523	3,018
16	4,853	2,036

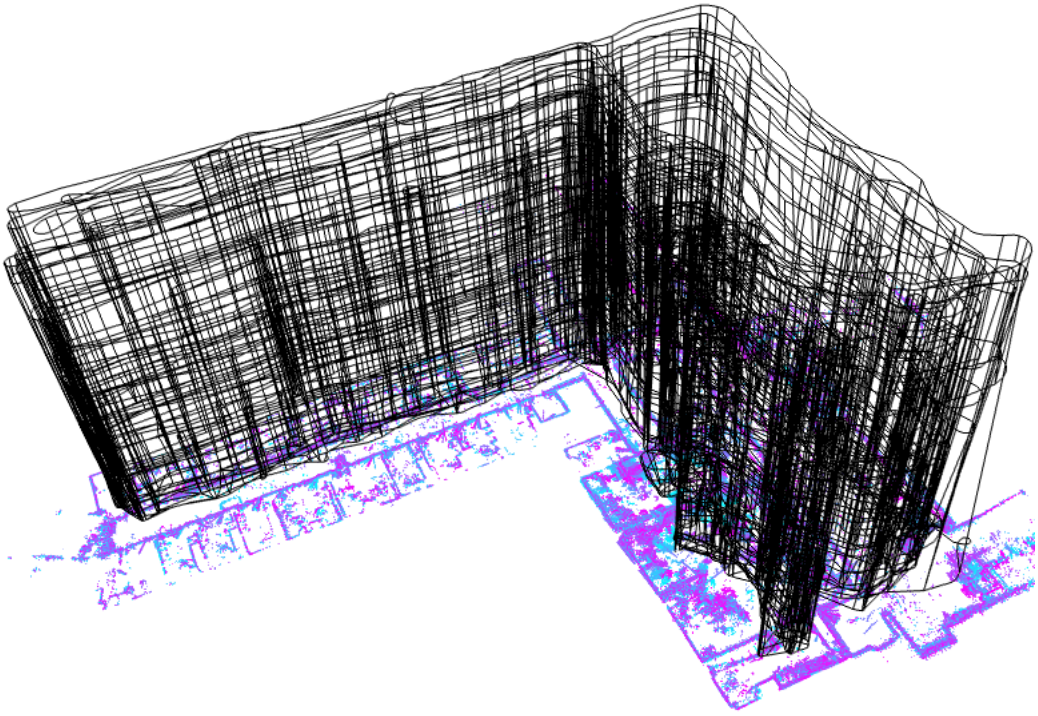
The static histograms from the DPG-SLAM-NR sixty passed are depicted in Figure 6-37. In general each of the histograms yield accurate results, where the walls are darker with each increasing number of sectors. The sixteen sectors static histogram contains the darkest outline of the static parts of the environment.

(a)

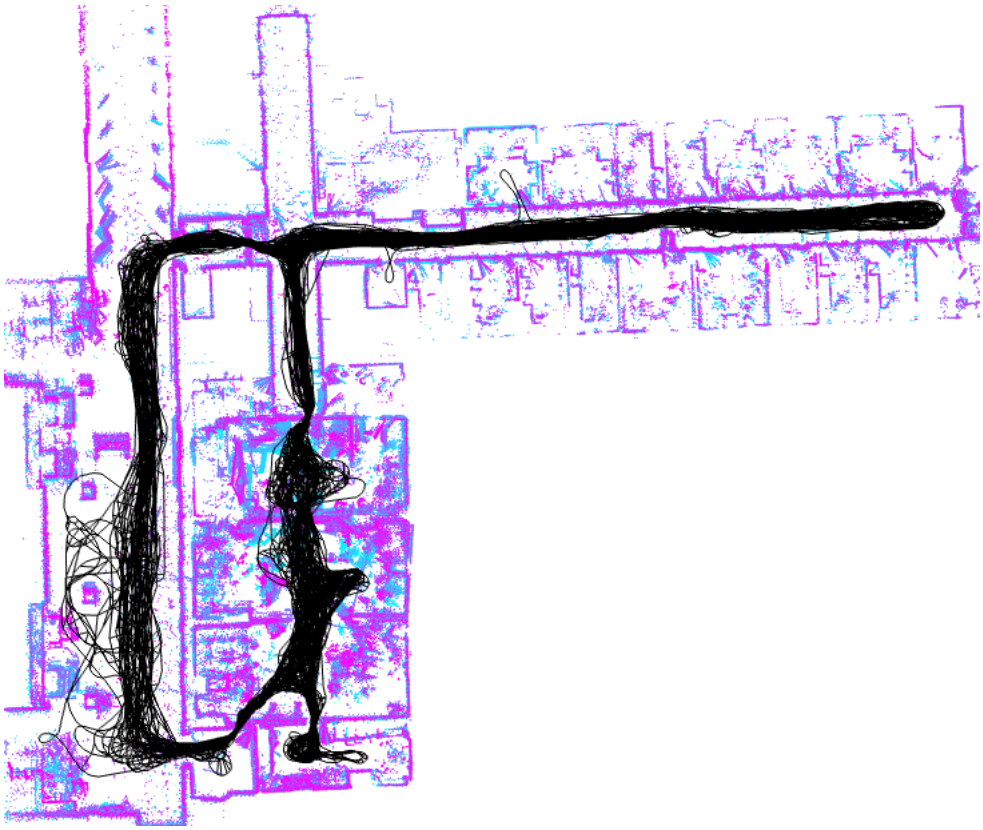


(b)

Figure 6-33: Maps of the orthogonal projection, (a), and three-dimensional side-view, (b), generated from pose graph SLAM applied to 60 passes through the University of Tubingen Robot Lab.



(a)



(b)

Figure 6-34: In Figure 6-34(a) the pose graph created over 60 passes is shown. The height of each edges is depicted as a function of time. Figure 6-34(b) is an orthogonal projection of the map and the pose graph.

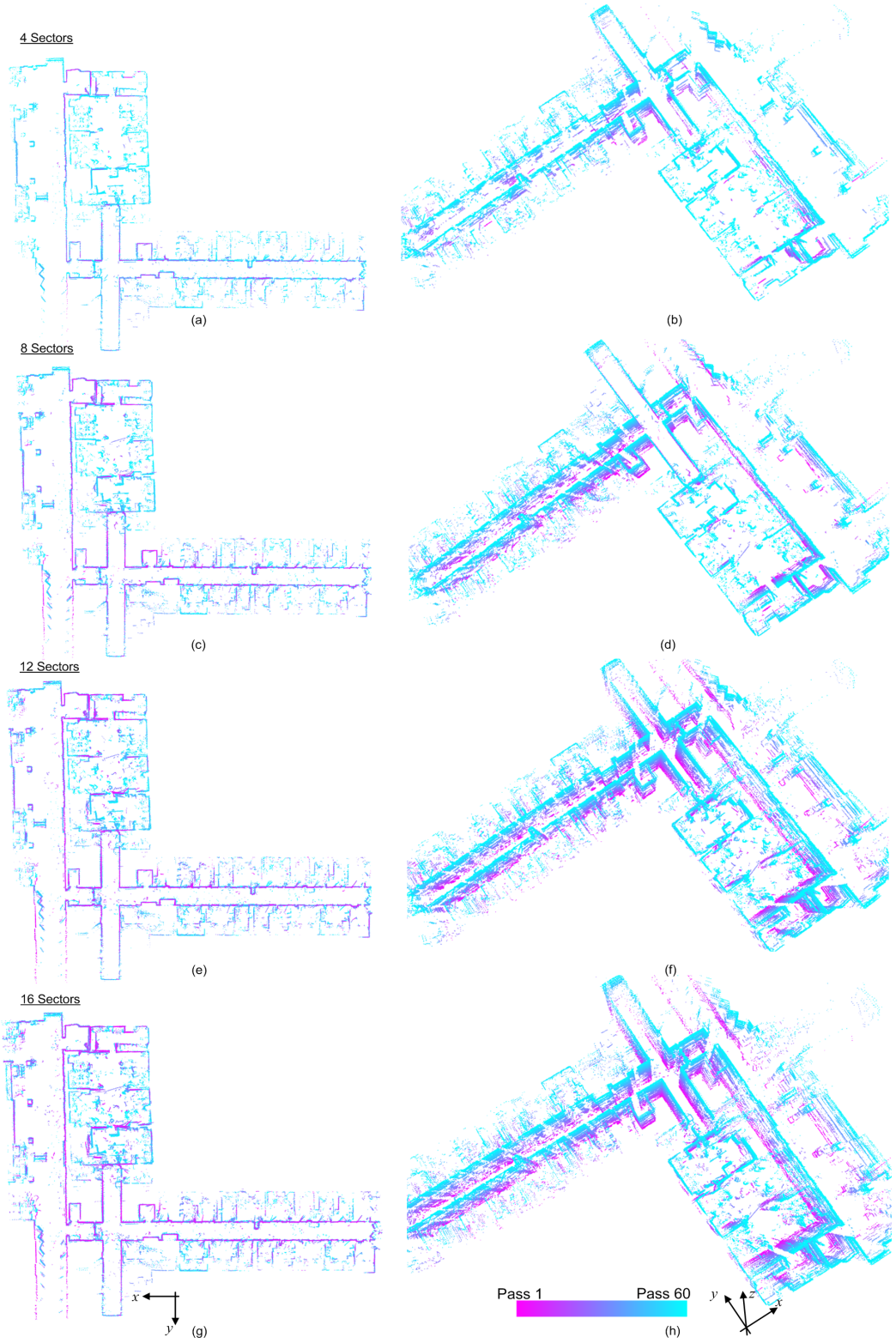


Figure 6-35: The active maps generated after applying DPG-SLAM-NR for sixty passes through the Univ. of Tubingen Robot Lab.

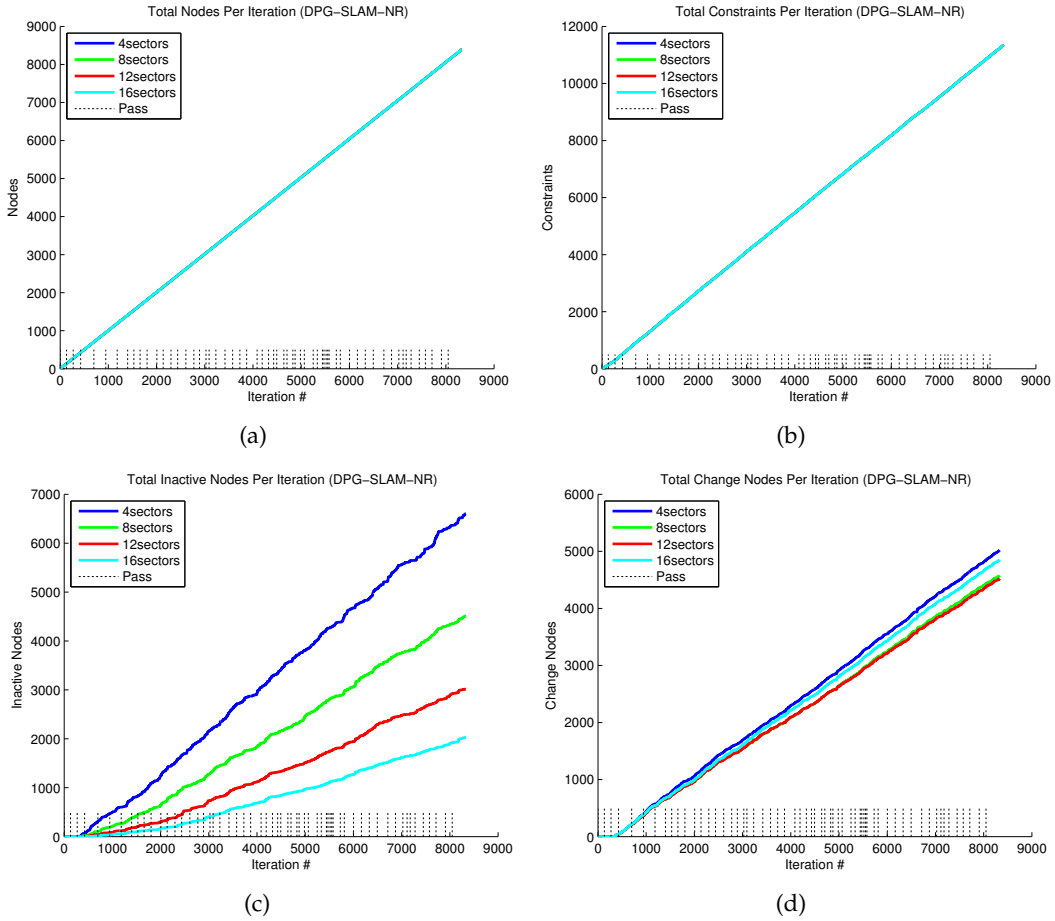


Figure 6-36: Graphs depicting the number of nodes and constraints per iteration (and per pass). The DPG-SLAM-NR algorithm was applied and the robot made 60 passes through the University of Tubingen Robot Lab.

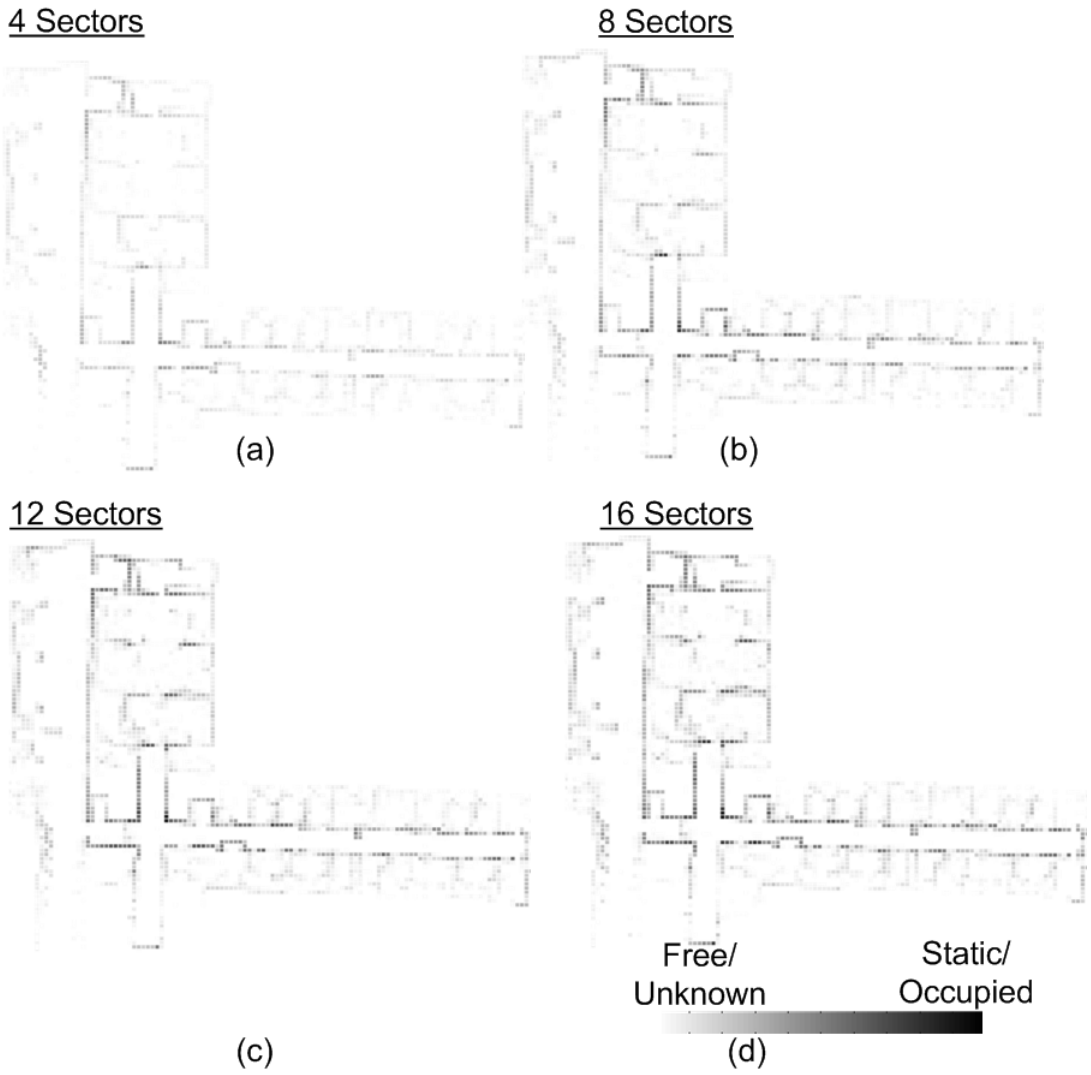


Figure 6-37: The static histograms generated from the active maps resulting from DPG-SLAM-NR applied to 60 passes through the Robot Lab.

DPG-SLAM Results

As stated, the analysis of the performance of DPG-SLAM-NR and DPG-SLAM for long-term operation enables us to interpret the performance of both algorithms on a large, long-term data set. In particular, we are interested in comparing and contrasting the accuracy and the computational efficiency resulting from both algorithms.

The active maps from the DPG-SLAM experiments are shown in Figure 6-38. Like the DPG-SLAM-NR active maps, the twelve and sixteen sectors DPG-SLAM active maps contain range points from many more passes than the four and eight sectors (see Figure 6-38b,d,f, and h).

Table 6.10: Total number of nodes and constraints remaining in each DPG after the robot makes sixty passes through the Univ. of Tubingen Robot Lab and DPG-SLAM-NR was applied.

Sectors	Nodes	Constraints	Change	Inactive	Rem Nodes	Rem Constraints
4	3,891	4,904	5,807	5,667	4,501	8,130
8	4,511	5,809	5,058	3,908	3,881	7,324
12	5,485	7,202	4,788	2,620	2,907	5,558
16	6,142	8,187	4,813	1,649	2,250	4,264

Table 6.10 presents the DPG-SLAM total nodes, change nodes, inactive nodes and removed nodes; as well as, the total constraints and removed constraints. The growth of each of the number of nodes and constraints over all the passes is shown in Figure 6-39. The four sectors experiment had the greatest number of constraints removed, and thus, contain the smallest DPG at the end of the passes. Given the total nodes and constraints, the final size of each DPG increases with the number of sectors. In addition, the number of change nodes seen in Table 6.10 are comparable, with the smallest number of sectors having the greatest number of change nodes.

The static histograms for the DPG-SLAM experiments are shown in Figure 6-40. In the four sectors it is very difficult to see the walls and other static objects. This shows that overall the active map does not include much, if any, of the ranges from all the passes.

6.5 Summary

This chapter presented analysis and results of both the DPG-SLAM-NR algorithm where no nodes or constraints are removed from the Dynamic Pose Graph, and the DPG-SLAM algorithm where nodes and constraints are removed from the DPG. Two distinct collections of data were used to explore the efficacy of the algorithms: the CSAIL Reading Room data set and the University of Tubingen Robot Lab data set. The CSAIL Reading Room is a smaller controlled environment where there were two low-dynamic objects, boxes, that were moved, removed or added for up to 20 passes. The University of Tubingen Robot Lab data set is the largest long-term mobile robot data set that was available at the time of this work. It contained data collected from operating a mobile robot at different times throughout the day over a 5 week period. We hand labeled distinguishing changes in the short-term data set in order to analyze the performance of the DPG-SLAM method. For the long-term operation, data for 60 passes through the environment over the 5 weeks was used. In each of the data sets, a B21 mobile robot equipped with a forward-facing laser range finder was used. Our results show that removing nodes and constraints in a

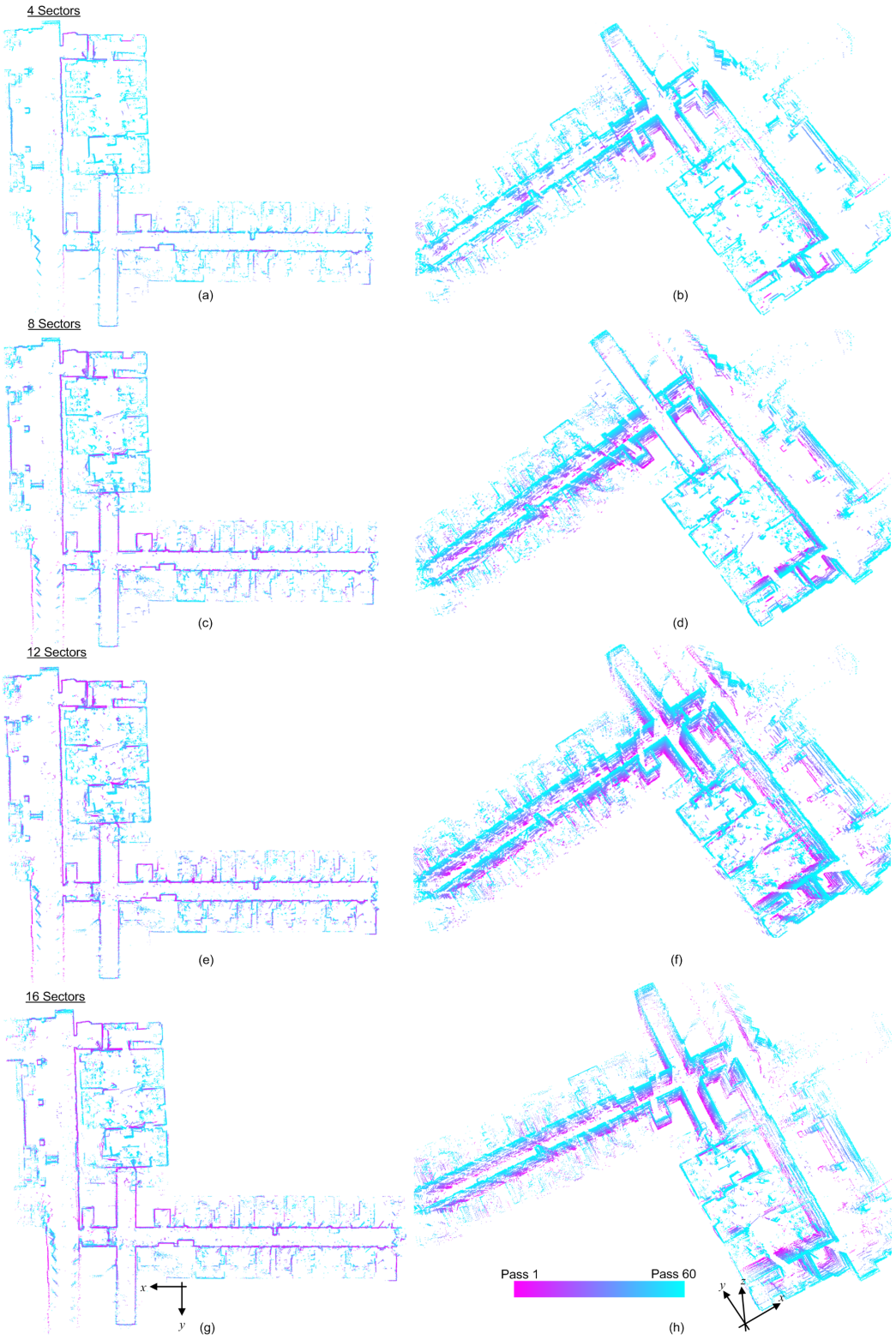


Figure 6-38: Images of the active maps created by executing DPG-SLAM for sixty passes through the Univ. of Tubingen Robot Lab.

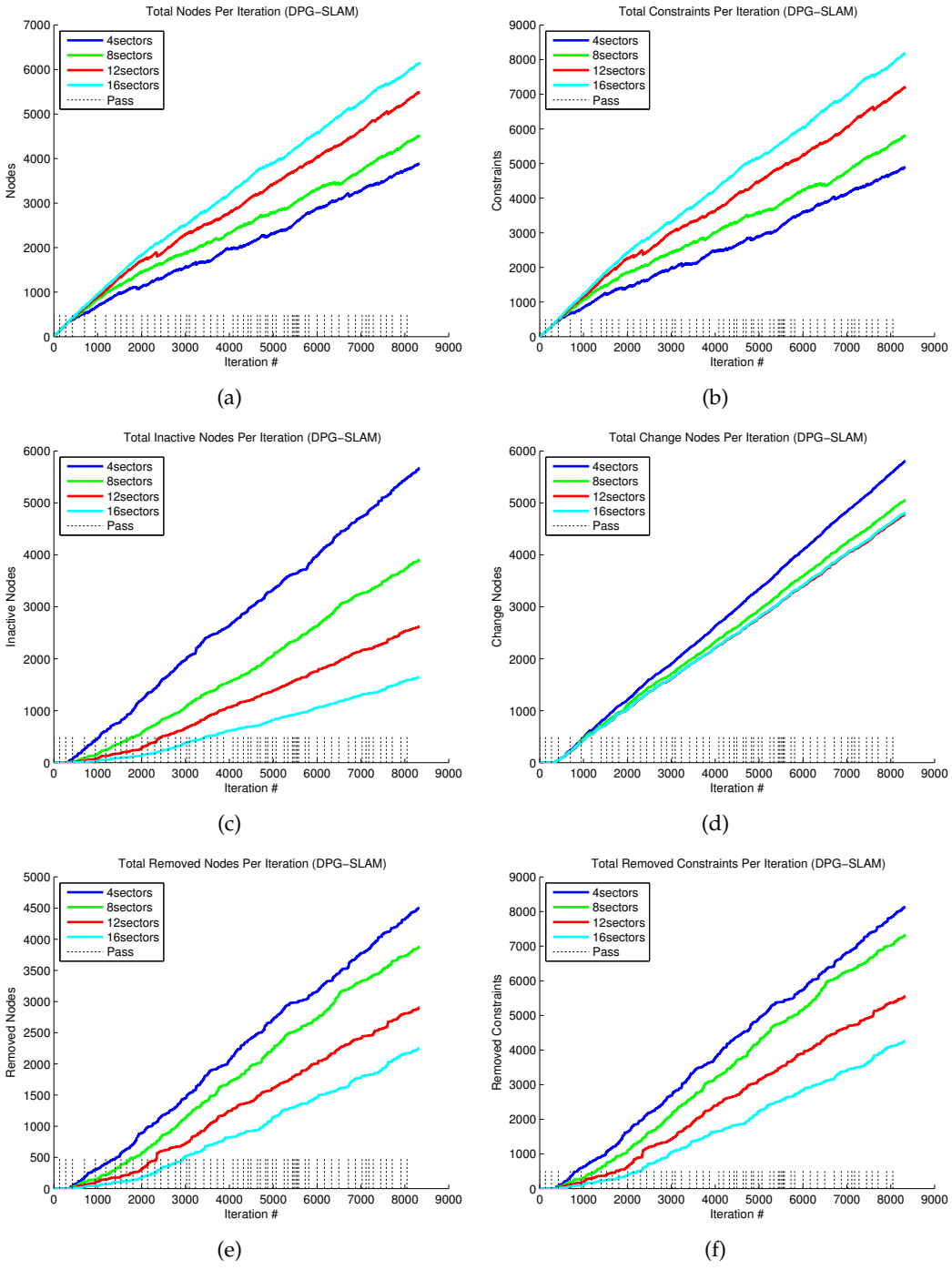


Figure 6-39: Each graph depicts the growth of the number of node types and constraints as the number of iterations increases and DPG-SLAM is applied. The robot made a total of 60 passes through the Robot Lab.

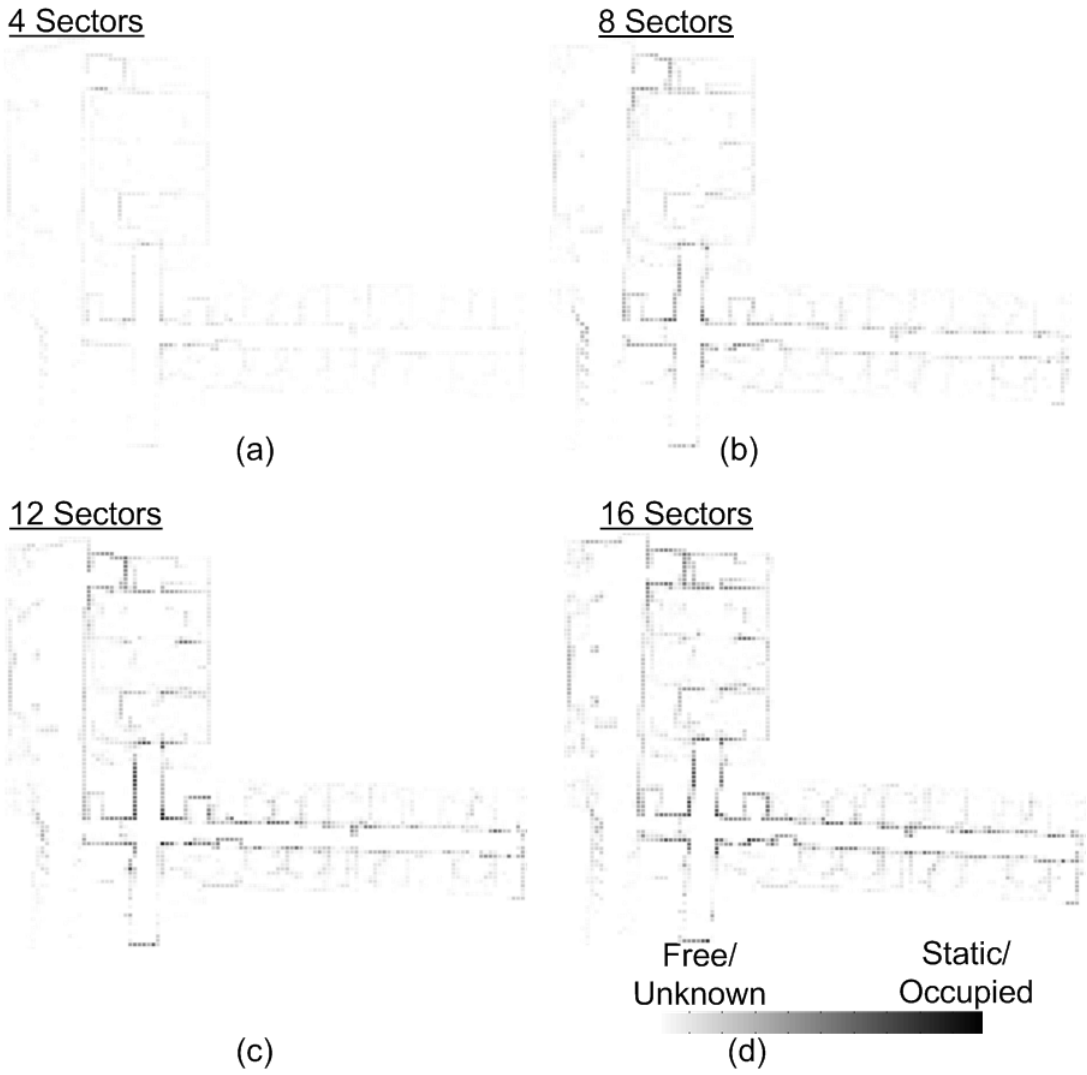


Figure 6-40: The static histograms generated from the active maps of DPG-SLAM applied to 60 passes through the University of Tubingen Robot Lab.

Dynamic Pose Graph effectively improves tractability of pose graph optimization while maintaining an accurate map. As a result, our method is suitable for mobile robots operating for long periods in low-dynamic environments.

Chapter 7

Conclusion

7.1 Discussion

There are a number of applications that benefit from a mobile robot capable of operating in a dynamic environment over extended periods, such as weeks, months, or even years. For example, in transportation security such as in airports or train stations, a security robot can be patrolling the environment looking for unexpected changes and alerting security personal. Also, in the case of natural disasters, such as after an earthquake or tsunami, a robot can be placed in a damaged building with an original map of the building and be able to detect changes in the structure as it moves or shifts over time.

Developing technologies that address the long-term SLAM problem is critical for robots to operate robustly over the long term with minimal human intervention. Particularly, to safely navigate, make decisions, and carry out long-term plans, the robot must be able to maintain an up-to-date and accurate map of the environment. Recall that our long-term mapping goals were to 1) continuously incorporate new information, 2) represent the environment history, 3) detect changes and update map online, and 4) address the problem of tractability as pose graphs grow. This thesis presented, DPG-SLAM, a novel method for a persistent mobile robot operating for long periods of time in a dynamic environment. Our research focused on the environments with low-dynamic and static objects. To detect when these objects are moved, removed or added to the environment, the robot must be able to makes numerous passes and updates its representation, the active and dynamic maps.

We presented results from two algorithms, DPG-SLAM-NR and DPG-SLAM. The first algorithm DPG-SLAM-NR executed the DPG-SLAM method, but nodes and constraints were not removed. The second algorithm DPG-SLAM addressed the tractability problem by removing nodes and constraints while maintaining up-to-date active and dynamic maps. Our results show that there was minimal trade-off in accuracy between the two algorithms, with a great benefit of computation time for the DPG-SLAM algorithm where the DPG size is reduced.

There are two recent related works relevant to the long-term mapping approach presented in this thesis. The first is the work by Biber and Duckett [9,10] where they collected a long-term data set over 5 weeks. To our knowledge, this was the first long-term operation experiment for a mobile robot mapping an indoor dynamic environment. The robot constructed an initial SLAM map from one pass, and then maintained a sample-based dynamic map while localizing of the initial SLAM map. The multiple maps were maintained

at different times scales and points in each of the map faded (were removed) according to their age. This work however suffers from executing localization after constructing one map. The initial SLAM map could be noisy and additional information acquired by the robot as it makes several passes through the environment should be used to improve the initial SLAM map. In addition, it is difficult to determine the number of maps to maintain as well as the timing for each of the time-scales on each map. Our work performs continuous SLAM on this data set and maintains two maps, an active and dynamic map. In addition, the Biber and Duckett method does not specifically identify changes in the environment, unlike the DPG-SLAM method the provides the dynamic map which maintains a representation of the history of environment changes over time.

The second related work is by Konolige and Bowman [39]. In their work they use multi-session vision for pose graph SLAM and cluster images at nodes in the pose graph. of the method on Changes are not specifically detected, rather like images are clustered and stored at nodes in the graph. They also address the problem of relocation by creating "weak links", edges in the graph, when the robot starts at a different position and attempts to localize itself in the map. These links can potentially be removed if the robot is not able to localize on its current pass. Their method is demonstrated on a few passes through an environment. Thus, it remains to show the scalability and efficacy of the method as it applies to long-term operation and scenarios with a number of low-dynamic objects.

7.1.1 Limitations

There are some limitation of the DPG-SLAM method that were evident in our results. First is the affect of ghosting where not all stale range points are removed from the active map as shown in Figure 7-1. A second limitation of DPG-SLAM is the potential for the dynamic map to have false positives. That is, points that are labeled added or removed, but the points represent measurements from static objects such as walls (see Figure7-2). A third limitation of our method is that there may be constraints removed from the graph that greatly affect the position estimates in a pass. For example, in Figure 7-3, constraints are removed leaving a sequence of nodes and edges that are not bounded by a loop constraint. Though this error is localized to one part of one pass, and with the DPG-SLAM method the DPG was able to maintain accurate pose estimates at later passes.

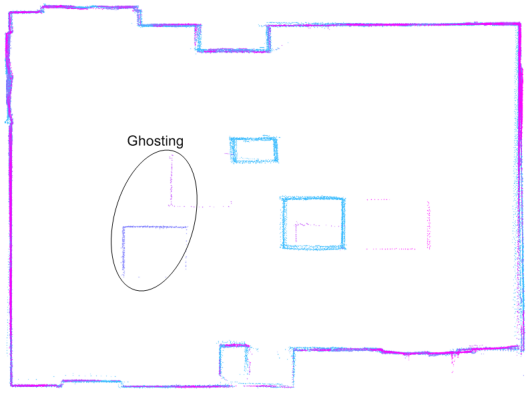


Figure 7-1: Example of ghosting.

Figure 7-2: Walls should not be considered changes, but as the pose estimates are updates with the additions and removal of constraints some measurement from walls will not coincide. As a result, changes are detected.

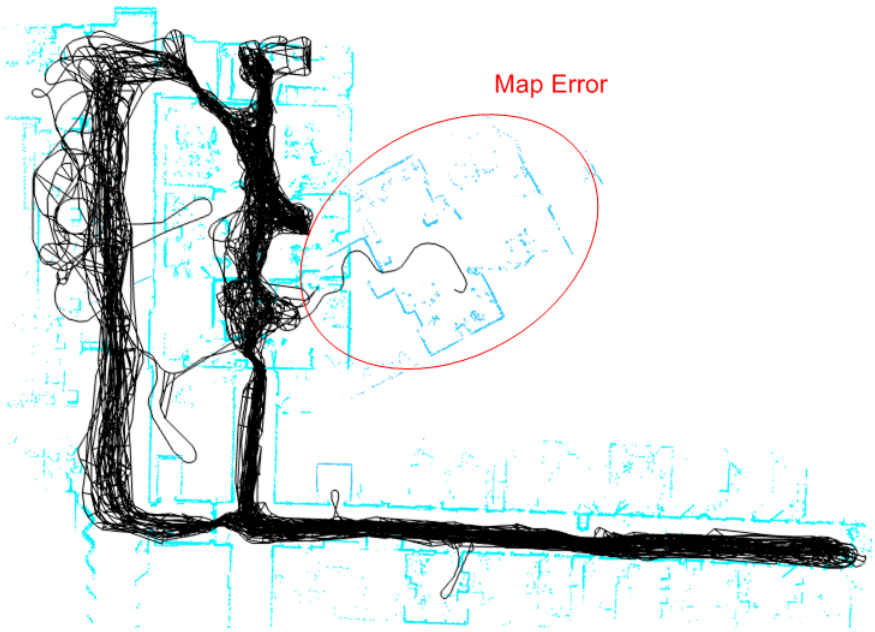


Figure 7-3: Example of when constraints are removed leaving a sequence of nodes, which have error that accrue and are not reduced due to no loop constraints.

7.2 Future Work

There are a number of exciting directions where this research can be furthered. We outline a few of our future research directions below.

A key part to the DPG-SLAM method is the ability to remove nodes and constraints from the Dynamic Pose Graph. In this work our criteria for reducing the size of the pose graph focused on minimizing the amount of information lost by removing nodes and constraints. Another criteria that we are exploring is how much the nodes in the DPG move, more specifically the difference in the positions of the nodes before and after the size of the DPG is reduced. The purpose of this is to understand how much the pose estimates are affected by have the selected nodes constraints are removed. Other criteria could be proposed that account for the covariances of the nodes as a result of node removal. In addition, we attempt to remove inactive nodes, if they meet the criteria, at one point in time during each iteration. As a result, there often remain inactive nodes in the graph that may later meet the criteria and can be removed at a later pass where DPG-SLAM is applied. Thus, future work for caching inactive nodes that are not immediately removed, and attempting to remove that are later points in time should be explored.

There are a number of areas for future research. First, a more efficient method for updating the map can be applied by integrating path planning into the map update process. The path planner could extract information from the map (dynamic pose graph) to decide which locations the robot should revisit and plan paths based on the potential for data at these locations to change. Second, with the collection of large amounts of spatiotemporal data a number of pattern recognition techniques can be applied in order to learn probabilistic model for change at various locations. Thirdly, the map representation can be improved to be able to handle user queries about the environment.

Another key area for future research is in active exploration and map maintenance of a dynamic environment. The robot will need to trade-off exploration of new areas in the environment or returning to previously visited area [63]. Developing a policy to actively explore a dynamic environment given the dynamic map would allow the robot to maintain an up-to-date map and use the active map to localize off the ”more static” parts of the environment. A probabilistic model to represent changes in the dynamic map can be used to determine which places the robot revisits. Given the current environment model and the registration policy, a measure of ”freshness” of the active and dynamic maps can be given. Freshness, a term adopted from web search engines [13, 14], and in mobile robot mapping it can be used to denote the percentage of the active map that are up-to-date within a given period.

Appendix A

Trajectory Estimation Problem Formulation

In feature-based SLAM, the map is represented as a set of features or landmarks. Features can be extracted from measurements and are denoted d_i, the set of features detected from measurement z_i. Determining the correspondences between measurements and features is called the data association problem [5, 56] and is not addressed in this thesis. The probabilistic formulation for the full SLAM posterior is given as,

$$P(X_{1:t}, M, D_{1:t}|Z_{1:t}, U_{1:t-1}). \tag{A.1}$$

This sections summarizes the least-squares problem formulation for computing the full state history. The aim is to compute the maximum-likelihood estimate for the trajectory given the measurements and control inputs.

The posterior for the robot state history is,

$$P(X_{1:t}|Z_{1:t}, U_{1:t-1}). \tag{A.2}$$

We apply Bayes' theorem to Equation (A.2) to get the following,

$$\begin{aligned} P(X_{1:t}|Z_{1:t}, U_{1:t-1}) &= \frac{P(z_t|Z_{1:t-1}, U_{1:t}, X_{1:t})P(X_{1:t}|Z_{1:t-1}, U_{1:t})P(Z_{1:t-1}, U_{1:t})}{P(z_t|Z_{1:t-1}, U_{1:t})P(Z_{1:t-1}, U_{1:t})} \\ &= \eta P(z_t|Z_{1:t-1}, U_{1:t}, X_{1:t})P(X_{1:t}|Z_{1:t-1}, U_{1:t}), \end{aligned}$$

where $\eta = 1/P(Z_t|Z_{1:t-1}, U_{1:t})$. We can apply the Markov assumption, commonly used in SLAM [4], for the first term,

$$P(z_t|Z_{1:t-1}, U_{1:t}, X_{1:t}) = P(z_t|X_{1:t}). \tag{A.3}$$

The second term in A.3 is factored and becomes,

$$P(X_{1:t}|Z_{1:t-1}, U_{1:t}) = P(x_t|X_{1:t-1}, Z_{1:t-1}, U_{1:t})P(X_{1:t-1}|Z_{1:t-1}, U_{1:t}). \tag{A.4}$$

The Markov assumption is applied to get,

$$P(x_t|X_{1:t-1}, Z_{1:t-1}, U_{1:t}) = P(x_t|x_{t-1}, u_t). \tag{A.5}$$

Then Equation A.4 becomes,

$$P(X_{1:t}|Z_{1:t-1}, U_{1:t}) = P(X_t|x_{t-1}, u_t)P(X_{1:t-1}|Z_{1:t-1}, U_{1:t}). \tag{A.6}$$

By combining the above equations we are able to get the following recursive formula,

$$P(X_{1:t}|Z_{1:t}, U_{1:t}) = \eta P(Z_t|X_{1:t})P(X_t|x_{t-1}, u_t)P(X_{1:t-1}|Z_{1:t-1}, U_{1:t}). \tag{A.7}$$

Then we apply induction over time to Equation (A.7), and get

$$P(X_{1:t}|Z_{1:t}, U_{1:t}) = \eta P(x_1) \prod_t P(x_t|u_t, x_{t-1}), \tag{A.8}$$

Oftentimes the prior on the initial pose, x_1, is dropped from Equation (A.8), resulting in

$$P(X_{1:t}|Z_{1:t}, U_{1:t}) = \eta \prod_t P(x_t|u_t, x_{t-1}). \tag{A.9}$$

Additionally, $P(x_t|u_t, x_{t-1})$ is known as the posterior distribution for the robot motion model. We adopt the common SLAM assumption to formulate the robot's motion model [4, 36] as the following Gaussian noise model, where w_t is zero-mean Gaussian noise with covariance Q,

$$x_t \leftarrow f_t(x_{t-1}, u_t) + w_t. \tag{A.10}$$

Applying the Gaussian assumption to Equation (A.10) results in the following posterior probability [4],

$$P(x_t|u_t, x_{t-1}) \propto exp[-1\frac{1}{2}(x_t - f_t(x_{t-1}, u_t))\lambda_t{}^{-1}(x_t - f_t(x_{t-1}, u_t))]. \tag{A.11}$$

As a result, the full trajectory posterior given in Equation (A.11) is a function of the robot motion model which adheres to a zero-mean Gaussian noise model. In the following section, we outline the method used to compute the posterior by adopting a least squares problem formulation [16, 42, 69].

A.1 Trajectory Estimation as Least Squares Formulation

The pose graph SLAM problem computes the minimum of the sum of all constraints in the pose graph. The constraints are nonlinear, as a result or the robot's rotation. Thus non-linear least squares methods are applied to find the minimum, globally consistent solution. Here we show the least squares problem formulation for the trajectory estimation as follows,

$$X^*_{1:t} = \arg\max_{X_{1:t}} P(X_{1:t}|Z_{1:t}, U_{1:t}). \tag{A.12}$$

Then we can expand Equation (A.12) and substitute in Equation (A.8) to get the following,

$$\begin{aligned}
&\arg\max_{X_{1:t}} P(X_{1:t}|Z_{1:t}, U_{1:t}) \\
&\qquad = \arg\min_{X_{1:t}} -logP(X_{1:t}|Z_{1:t}, U_{1:t}) \\
&\qquad = \arg\min_{X_{1:t}} -\log\eta \prod_t P(x_t|u_t, x_{t-1}) \\
&\qquad = \arg\min_{X_{1:t}} [-\log\eta + \sum_t -\log P(x_t|u_t, x_{t-1}].
\end{aligned}$$

Finally, we substitute in the motion model which results in the following,

$$X^*_{1:t} = constant + \arg\min_{X_{1:t}} [\sum_t (x_t - f_t(x_{t-1}, u_t)){\lambda_t}^{-1}(x_t - f_t(x_{t-1}, u_t))] \tag{A.13}$$

If we linearize the motion model then we form a linear least squares problem.

Bibliography

[1] Defense Advanced Research Projects Agency. DARPA announces third grand challenge: Urban challenge moves to the city, May 2006.

[2] J. Andrade-Cetto and A. Sanfeliu. Concurrent map building and localization on indoor dynamic environments. *International Journal of Pattern Recognition and Artificial Intelligence*, 16(3):361–374, 2002.

[3] D. Arbuckle, A. Howard, and M. J. Mataric. Temporal occupancy grids: a method for classifying spatio-temporal properties of the environment. In *Proc. of the IEEE-RSJ Int. Conf. on Intelligent Robots and Systems, Lausanne,(CH)*, 2002.

[4] T. Bailey and H. Durrant-Whyte. Simultaneous localisation and mapping (SLAM): Part i the essential algorithms. *Robotics and Automation Magazine*, 2006.

[5] T. Bailey, E. M. Nebot, J. K. Rosenblatt, and H. F. Durrant-whyte. Data association for mobile robot navigation: a graph theoretic approach. In *in Proc. IEEE Int. Conf. Robotics and Automation*, pages 2512–2517, 2000.

[6] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot. Consistency of the EKF-SLAM algorithm. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 3562 –3568, October 2006.

[7] P. Besl and N. McKay. A method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 14(2):239–256, 1992.

[8] P. Biber. The normal distribution transform: A new approach to laser scan matching, 2003.

[9] P. Biber and T. Duckett. Dynamic maps for long-term operation of mobile service robots. *Proc. of Robotics: Science and Systems (RSS)*, 2005.

[10] P. Biber and T. Duckett. Experimental analysis of Sample-Based maps for Long-Term SLAM. *The International Journal of Robotics Research*, 28(1):20–33, 2009.

[11] R. Biswas, B. Limketkai, S. Sanner, and S. Thrun. Towards object mapping in non-stationary environments with mobile robots. In *Intelligent Robots and System, 2002. IEEE/RSJ International Conference on*, volume 1, pages 1014–1019 vol.1, 2002.

[12] M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller. An atlas framework for scalable mapping. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 2, pages 1899–1906 vol.2, 2003.

[13] B. Brewington and G. Cybenko. How dynamic is the web? pages 257–276, 2000.

[14] J. Cho and H. Garcia-Molina. Effective page refresh policies for web crawlers. Technical Report 2003-44, Stanford InfoLab, December 2003.

[15] M. Darms, P. Rybski, C. Baker, and C. Urmson. Obstacle detection and tracking for the urban challenge. *Trans. Intell. Transport. Sys.*, 10:475–485, September 2009.

[16] F. Dellaert and M. Kaess. Square root sam: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, 25(12):1181–1203, 2006.

[17] B. Douillard, D. Fox, and F. Ramos. Laser and vision based outdoor object mapping. In *Proceedings of Robotics: Science and Systems IV*, Zurich, Switzerland, June 2008.

[18] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989.

[19] J. Folkesson and H. Christensen. Closing the loop with graphical SLAM. *Robotics, IEEE Transactions on*, 23(4):731 –741, August 2007.

[20] J. Folkesson, P. Jensfelt, and H.I. Christensen. Graphical SLAM using vision and the measurement subspace. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 325–330, 2005.

[21] D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journalof Artificial Intelligence Research*, 11, 1999.

[22] D. Fox, W. Burgard, S. Thrun, and A. B. Cremers. Position estimation for mobile robots in dynamic environments. In *In Proc. of the National Conference on Artificial Intelligence (AAAI*, 1998.

[23] L. Freda, F. Loiudice, and G. Oriolo. A randomized method for integrated exploration. In *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, pages 2457–2464, 2006.

[24] L. Freda and G. Oriolo. Frontier-Based probabilistic strategies for Sensor-Based exploration. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 3881–3887, 2005.

[25] M. Fu. Interpolation apprrach to h estimation and its interconnection to loop transfer recovery. *Systems and Control Letters*, 1991.

[26] K. Granstrom, J. Callmer, F. Ramos, and J. Nieto. Learning to detect loop closure from range data. In *Proceedings of the 2009 IEEE international conference on Robotics and Automation*, ICRA'09, pages 1990–1997, Piscataway, NJ, USA, 2009. IEEE Press.

[27] G. Grisetti. Improving grid-based SLAM with rao-blackwellized particle filters by adaptive proposals and selective resampling. In *In Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 2443–2448, 2005.

[28] G. Grisetti, R. Kummerle, C. Stachniss, and W. Burgard. A tutorial on graph-based SLAM. *Intelligent Transportation Systems Magazine, IEEE*, 2(4):31–43, 2010.

[29] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23:2007, 2007.

[30] J-S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *International Symposium on Computational Intelligence in Robotics and Automation*, 1999.

[31] D. Hahnel, D. Schulz, and W. Burgard. Map building with mobile robots in populated environments. In *Intelligent Robots and System, 2002. IEEE/RSJ International Conference on*, volume 1, pages 496–501 vol.1, 2002.

[32] D. Hahnel, R. Triebel, W. Burgard, and S. Thrun. Map building with mobile robots in dynamic environments. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 2, pages 1557–1563 vol.2, 2003.

[33] L. Haines. Spirit rover clocks up six years on mars, January 2010.

[34] F. Harary. *Graph Theory.* Reading, MA: Addison-Wesley, 1994.

[35] X. Ji, H. Zhang, D. Hai, and Z. Zheng. A decision-theoretic active loop closing approach to autonomous robot exploration and mapping. In Luca Iocchi, Hitoshi Matsubara, Alfredo Weitzenfeld, and Changjiu Zhou, editors, *RoboCup 2008: Robot Soccer World Cup XII*, volume 5399 of *Lecture Notes in Computer Science*, pages 507–518. Springer Berlin / Heidelberg, 2009.

[36] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: incremental smoothing and mapping. *Robotics, IEEE Transactions on*, 24(6):1365–1378, 2008.

[37] K. Konolige. Large-scale map-making. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, San Jose, CA, 2004. AAAI.

[38] K. Konolige and M. Agrawal. FrameSLAM: From bundle adjustment to real-time visual mapping. *IEEE Transactions on Robotics*, 24(5):1066–1077, 2008.

[39] K. Konolige and J. Bowman. Towards lifelong visual maps. In *IROS - best paper finalist*, pages 1156–1163, 2009.

[40] K. Konolige, J. Bowman, J. D. Chen, P. Mihelich, M. Calonder, V. Lepetit, and P. Fua. View-based maps. In *Proceedings of Robotics: Science and Systems*, Seattle, USA, June 2009.

[41] K. Konolige, G. Grisetti, R. Kummerle, W. Burgard, B. Limketkai, and R. Vincent. Sparse pose adjustment for 2d mapping. In *IROS*, Taipei, Taiwan, October 2010.

[42] H. Kretzschmar. Life-long map learning for graph-based simultaneous localization and mapping. Master's thesis, University of Freiburg, 2009.

[43] B. Kuipers and Y. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. pages 47–63, 1993.

[44] J. Leonard and H. Feder. Experimental analysis of adaptive concurrent mapping and localization using sonar. In *The Sixth International Symposium on Experimental Robotics VI*, pages 297–306, London, UK, 2000. Springer-Verlag.

[45] J. J. Leonard and H. F. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *In Proc. IEEE Int. Workshop on Intelligent Robots and Systems*, pages 1442–1447, Osaka, Japan, 1991. MIT Press.

[46] X. Li and A. Zell. Filtering for a mobile robot tracking a free rolling ball. In Gerhard Lakemeyer, Elizabeth Sklar, Domenico Sorrenti, and Tomoichi Takahashi, editors, *RoboCup 2006: Robot Soccer World Cup X*, volume 4434 of *Lecture Notes in Computer Science*, pages 296–303. Springer Berlin / Heidelberg, 2007.

[47] G. Lidoris, F. Rohrmuller, D. Wollherr, and M. Buss. The autonomous city explorer (ace) project: mobile robot navigation in highly populated urban environments. In *Proceedings of the 2009 IEEE international conference on Robotics and Automation*, ICRA'09, pages 2238–2244, Piscataway, NJ, USA, 2009. IEEE Press.

[48] F. Lu and E. Milios. Robot pose estimation in unknown environments by matching 2d range scans. *Journal of Intelligent and Robotic Systems*, 18:249–275, 1994.

[49] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Auton. Robots*, 4(4):333–349, 1997.

[50] M. Maimone, Y. Cheng, and L. Matthies. Two years of visual odometry on the Mars Exploration Rovers. *Journal of Field Robotics*, 24(3):169–186, 2007.

[51] M. Milford and G. Wyeth. Persistent navigation and mapping using a biologically inspired SLAM system. *Int. J. Rob. Res.*, 29:1131–1153, August 2010.

[52] N. C. Mitsou and C. S. Tzafestas. Temporal occupancy grid for mobile robot dynamic environment mapping. In *Control & Automation, 2007. MED '07. Mediterranean Conference on*, pages 1–8, 2007.

[53] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Canada, 2002. AAAI.

[54] L. Montesano, J. Minguez, and L. Montano. Modeling dynamic scenarios for local sensor-based motion planning. *Auton. Robots*, 25(3):231–251, 2008.

[55] H. P. Moravec. Sensor Fusion in Certainty Grids for Mobile Robots. *AI Magazine*, 9(2):61–74, 1988.

[56] J. Neira and J. D. Tards. Data association in stochastic mapping using the joint compatibility test, 2001.

[57] E. Olson. Real-time correlative scan matching. In *ICRA'09: Proceedings of the 2009 IEEE international conference on Robotics and Automation*, pages 1233–1239, Piscataway, NJ, USA, 2009. IEEE Press.

[58] E. Olson, J. Leonard, and S. Teller. Fast iterative alignment of pose graphs with poor initial estimates. *Proceedings of the IEEE International Conference on Robotics and Automation*, page 22622269, 2006.

[59] G. Oriolo, M. Vendittelli, L. Freda, and G. Troso. The SRT method: randomized strategies for exploration. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 5, pages 4688–4694 Vol.5, 2004.

[60] J. Pineau, M. Montemerlo, M. Pollack, N. Roy, and S. Thrun. Towards robotic assistants in nursing homes: Challenges and results. *Special issue on Socially Interactive Robots, Robotics and Autonomous Systems*, 42(1):271 – 281, 2003.

[61] N. Roy, G. Baltus, D. Fox, F. Gemperle, J. Goetz, T. Hirsch, D. Margaritis, M. Montemerlo, J. Pineau, J. Schulte, and S. Thrun. Towards personal service robots for the elderly. In *Carnegie Mellon University*, 2000.

[62] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *International Conference on 3-D Digital Imaging and Modeling*, 2001.

[63] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, second edition, 2003.

[64] R. Siegwart, K. Arras, S. Bouabdallah, D. Burnier, G. Froidevaux, X. Greppin, B. Jensen, A. Lorotte, L. Mayor, and M. Meisser. Robox at expo. 02: A large-scale installation of personal robots. *Robotics and Autonomous Systems*, 42(3-4):203–222, 2003.

[65] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. *Autonomous Robot Vehicles*, 1:167–193, 1990.

[66] C. Stachniss and W. Burgard. Mobile robot mapping and localization in non-static environments. In *AAAI'05: Proceedings of the 20th national conference on Artificial intelligence*, pages 1324–1329. AAAI Press, 2005.

[67] S. Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

[68] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz. Probabilistic algorithms and the interactive museum tour-guide robot minerva. *Int. J. Robotics Research*, 19(11):972–999, 2000.

[69] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT Press, 2005.

[70] S. Thrun and T. M. Mitchell. Lifelong robot learning. *Robotics and Autonomous Systems*, 15(1-2):25–46, 1995.

[71] B. Tovar, L. Muoz-Gmez, R. Murrieta-Cid, M. Alencastre-Miranda, R. Monroy, and S. Hutchinson. Planning exploration strategies for simultaneous localization and mapping. *Robotics and Autonomous Systems*, 54(4):314–331, 2006.

[72] M. R. Walter, R. M. Eustice, and J. J. Leonard. Exactly sparse extended information filters for feature-based SLAM. *International Journal of Robotics Research*, 26(4):335–359, April 2007.

[73] C.-C. Wang and C. Thorpe. Simultaneous localization and mapping with detection and tracking of moving objects. In *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 3, pages 2918–2924, 2002.

[74] C.-C. Wang, C. Thorpe, and S. Thrun. Online simultaneous localization and mapping with detection and tracking of moving objects: theory and results from a ground vehicle in crowded urban areas. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 1, pages 842–849 vol.1, 2003.

[75] C-C. Wang, C. Thorpe, S. Thrun, M. Hebert, and H. Durrant-Whyte. Simultaneous localization, mapping and moving object tracking. *The International Journal of Robotics Research*, 26(9):889–916, 2007.

[76] O. Wijk. *Triangulation Based Fusion of Sonar Data with Application in Mobile Robot Mapping and Localization*. PhD thesis, Royal Institute of Technology, Stockholm, Sweden, 2001.

[77] D. Wolf and G. S. Sukhatme. Online simultaneous localization and mapping in dynamic environments. In *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 2, pages 1301–1307 Vol.2, 2004.

[78] D. F. Wolf and G. S. Sukhatme. Towards mapping dynamic environments. *In Proceedings of the International Conference on Advanced Robotics (ICAR)*, page 594600, 2003.

[79] D. F. Wolf and G. S. Sukhatme. Mobile robot simultaneous localization and mapping in dynamic environments. *Autonomous Robots*, 19(1):53–65, 2005.

[80] G. Wyeth and M. Milford. Towards Lifelong Navigation and Mapping in an Office Environment. *Proceedings of the 14th International Symposium of Robotics Research (ISRR)*, 2009.

[81] G. Wyeth and M. Milford. Towards lifelong navigation and mapping in an office environment. *International Symposium of Robotics Research*, 2009.

[82] W. Xin and J. Pu. An improved ICP algorithm for point cloud registration. In *Computational and Information Sciences (ICCIS), 2010 International Conference on*, pages 565 –568, December 2010.

[83] B. Yamauchi and R. Beer. Spatial learning for navigation in dynamic environments. *Systems, Man and Cybernetics, Part B, IEEE Transactions on*, 26(3):496–505, 1996.