Long-Term Simultaneous Localization and Mapping in Dynamic Environments

by

Nicholas D. Carlevaris-Bianco

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Electrical Engineering: Systems)
in The University of Michigan
2015

ACKNOWLEDGEMENTS

This thesis was made possible by: least squares; my wife Amy and my parents Chris and Umberto; my adviser Ryan Eustice; my thesis committee members, Prof. Grizzle, Prof. Hero and Prof. Kuipers; everyone who passed through PeRL while I was there including, Ayoung, Gaurav, Jeff, Paul, Wolcott, Steve, GT, Alex, Enric, Schuyler, Arash, Jie, Steven, Josh and Vittorio; Jim and the folks at Ford; Ed, Andrew and Joho in April; the National Science Foundation under award IIS-0746455; the Office of Naval Research under awards N00014-12-1-0092 and N00014-12-1-0093; the Naval Sea Systems Command through the Naval Engineering Education Center under award N65540-10-C-0003; Ford Motor Company via the Ford-UM Alliance under award N015392; and viewers like you.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

LIST OF ABBREVIATIONS

CLT	Chow-Liu tree
CMLP	convolutional multi-layer perceptron
CPU	central processing unit
DOF	degree of freedom
EIF	extended information filter
EKF	extended Kalman filter
FOG	fiber optic gyro
GLC	generic linear constraint
GPU	graphics processing unit
HAUV	hovering autonomous underwater vehicle
IMU	inertial measurement unit
iSAM	incremental smoothing and mapping
KLD	Kullback-Leibler divergence
LIDAR	light detection and ranging
MAP	maximum *a posteriori*
MC	measurement composition
MLP	multi-layer perceptron
MRF	Markov random field
RANSAC	random sample consensus

SIFT	scale invariant feature transform
SLAM	simultaneous localization and mapping
SURF	speeded up robust features
t-SNE	t-distributed stochastic neighbor embedding

ABSTRACT

One of the core competencies required for autonomous mobile robotics is the ability to use sensors to perceive the environment. From this noisy sensor data, the robot must build a representation of the environment and localize itself within this representation. This process, known as simultaneous localization and mapping (SLAM), is a prerequisite for almost all higher-level autonomous behavior in mobile robotics. By associating the robot's sensory observations as it moves through the environment, and by observing the robot's ego-motion through proprioceptive sensors, constraints are placed on the trajectory of the robot and the configuration of the environment. This results in a probabilistic optimization problem to find the most likely robot trajectory and environment configuration given all of the robot's previous sensory experience. SLAM has been well studied under the assumptions that the robot operates for a relatively short time period and that the environment is essentially static during operation. However, performing SLAM over long time periods while modeling the dynamic changes in the environment remains a challenge.

The goal of this thesis is to extend the capabilities of SLAM to enable long-term autonomous operation in dynamic environments. The contribution of this thesis has three main components: First, we propose a framework for controlling the computational complexity of the SLAM optimization problem so that it does not grow unbounded with exploration time. Second, we present a method to learn visual feature descriptors that are more robust to changes in lighting, allowing for improved data association in dynamic environments. Finally, we use the proposed sparse-approximate marginalization and learned visual features in a SLAM system that explicitly models the dynamics of the environment in the map by representing each location as a set of example views that capture how the location changes with time.

We experimentally demonstrate that the proposed methods enable long-term SLAM in dynamic environments using a large, real-world vision and LIDAR dataset collected bi-weekly over the course of more than a year. This dataset, collected specifically for this research, captures a wide variety of dynamics: from short-term scene changes including moving people, cars, changing lighting, and weather conditions; to long-term dynamics including seasonal conditions and structural changes caused by construction.

CHAPTER I

Introduction

As robotic applications transition from engineered environments, such as factories, to the real world, one of the core competencies required is the ability to answer “what does the environment ‘look’ like?” and “where am I within the environment?” This problem, known as SLAM [7, 47], is critical to any robotic application that requires a mobile robot to operate in an uncontrolled environment. Fundamentally, SLAM is an estimation problem in which noisy observations of the environment, collected by the robot’s sensors, are used to estimate a map of the environment and the robot’s trajectory within the map.

SLAM has been well studied over the last several decades, and there now exists a mature set of probabilistic tools for interpreting the robot’s sensor data and solving SLAM’s underlying estimation problem. However, most implementations are only successful in the short-term and in environments that can reasonably be assumed to be static during the duration of robotic operation.

A complete SLAM system consists of two key components: the front-end, that extracts measurements from the raw sensor data, and the back-end, that finds the optimal configuration of historic robot poses and map features given the current measurements. Long-term applications of SLAM complicate both the back-end and front-end of the system.

The primary concern when developing a SLAM back-end for long-term applications is computational complexity. Optimization methods are the state-of-the-art tools for many large scale SLAM applications. However, some important drawbacks remain. Formulations that save all previous robot poses (important to maintain problem sparsity) do not scale well temporally as new poses (i.e., latent variables) must be added to the graph in order for the robot to stay localized, even if the robot continuously explores an already mapped location.

An appropriate front-end for long-term SLAM must be able to extract measurements from sensor data collected in the presence of large dynamic changes. The effect of dynamic changes on data association is dependent on the sensing modality. For example, light detection and ranging (LIDAR) is mostly affected by dynamic occlusion and longer-term structural changes to the environment, while vision is strongly affected by changes in lighting caused by the time of day and weather, as well as longer-term structural and seasonal changes.

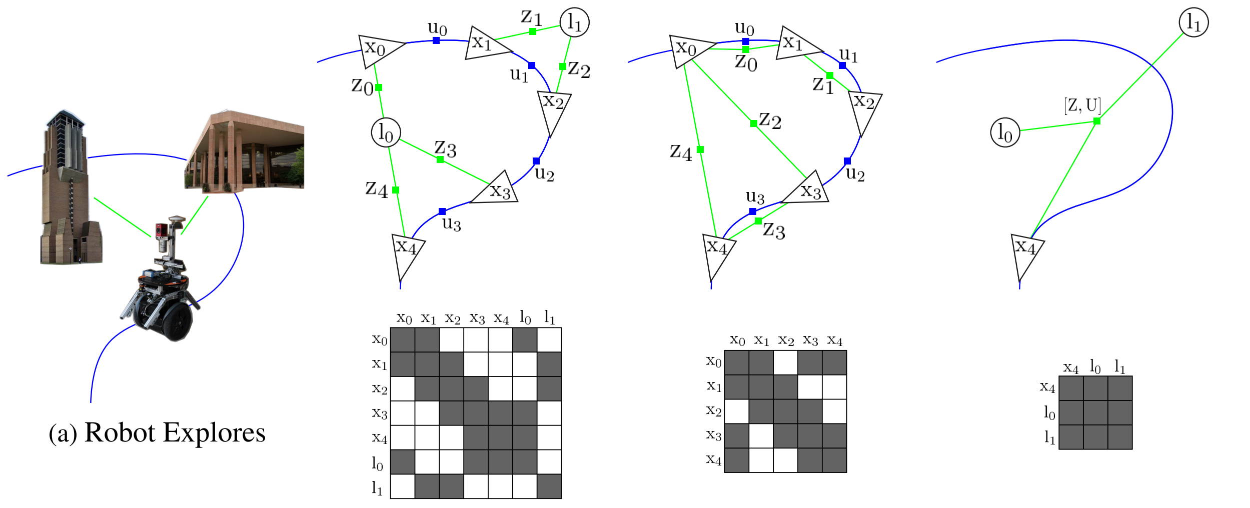
This thesis focuses on extending the state-of-the-art to account for the challenges of performing SLAM in dynamic environments over long periods of time. Our proposed methods will allow for SLAM systems that model and adapt to dynamic changes in the environment and remain accurate and computationally feasible in the long-term.

1.1 SLAM Background

We first provide an overview of SLAM. Readers familiar with SLAM may wish to skip directly to the literature review in §1.2.

1.1.1 SLAM Formulation

Figure 1.1 Factor graphs for common SLAM formulations. In (a), a robot explores an environment observing two landmarks. The full SLAM formulation, (b), estimates the location of historical poses and the landmarks given measurements and motion constraints. The pose SLAM formulation, (c), estimates the location of historical poses using sensor observations to produce constraints directly between pose nodes. The landmark SLAM formulation, (d), estimates the current pose of the robot, and all landmark locations given measurements and motion constraints. The sparsity pattern of the associated information matrix is shown for each formulation.



(a) Robot Explores

(b) Full SLAM

(c) Pose SLAM

(d) Landmark SLAM

Metric SLAM is commonly described by one of three formulations: full SLAM, pose SLAM and landmark SLAM (Fig. 1.1). In the full SLAM problem (Fig. 1.1(a)), the robot's trajectory is represented by a set of poses, $\mathbf{x}_i \in \mathbf{X}$, where each element, $\mathbf{x}_i$, is a vector that commonly represents the pose of the robot, but may contain additional elements to be estimated, such as velocities or other system parameters. The position and parameters of observed landmarks are represented as vectors $\mathbf{l}_j \in \mathbf{L}$. Noisy observations of the robots motion, $\mathbf{u}_i \in \mathbf{U}$, and observations of landmarks in the environment, $\mathbf{z}_k \in \mathbf{Z}$, are used to find the maximum *a posteriori* (MAP) estimate of the robot trajectory and the locations of the landmarks,

$$\hat{\mathbf{X}}, \hat{\mathbf{L}} = \underset{\mathbf{X},\mathbf{L}}{\operatorname{argmax}}\, p(\mathbf{X}, \mathbf{L} | \mathbf{U}, \mathbf{Z}). \tag{1.1}$$

For some sensing modalities, it may be less practical to repeatedly detect and associate landmarks in the environment. Instead, it may be possible to register two observations of the environment in order to directly measure the motion of the robot. This is referred to as pose SLAM (Fig. 1.1(c)) and can intuitively be thought of as marginalizing out the landmark poses from the full SLAM problem

$$\hat{\mathbf{X}} = \underset{\mathbf{X}}{\operatorname{argmax}}\, p(\mathbf{X} | \mathbf{U}, \mathbf{Z}). \tag{1.2}$$

Additionally, it is possible to estimate only the most recent pose of the robot, $\mathbf{x}_t$, and the landmark positions. We refer to this as landmark SLAM (Fig. 1.1(d)) and it can be thought of as the full SLAM problem with all previous robot poses marginalized out

$$\hat{\mathbf{x}}_t, \hat{\mathbf{L}} = \underset{\mathbf{x_t},\mathbf{L}}{\operatorname{argmax}}\, p(\mathbf{x}_t, \mathbf{L} | \mathbf{U}, \mathbf{Z}). \tag{1.3}$$

The choice of SLAM formulation is tightly coupled with the method used to solve the optimization problem. As we will discuss in §1.2.2, landmark SLAM was developed in the context of recursive filtering, while the full SLAM problem is associated with optimization-based smoothing techniques. Pose SLAM falls somewhere between, and has been solved with both filtering and smoothing methods.

By assuming that measurements are made under Gaussian noise, finding the optimum value in (1.1), (1.2) and (1.3) involves solving a nonlinear least squares optimization problem. For simplicity of notation, let all types of nodes be represented as $\boldsymbol{\mathcal{X}} = \mathbf{X} \cup \mathbf{L}$, and all observed variables be represented as $\boldsymbol{\mathcal{Z}} = \mathbf{Z} \cup \mathbf{U}$. Then

$$\hat{\boldsymbol{\mathcal{X}}} = \underset{\boldsymbol{\mathcal{X}}}{\operatorname{argmax}}\, p(\boldsymbol{\mathcal{X}}|\boldsymbol{\mathcal{Z}}) = \underset{\boldsymbol{\mathcal{X}}}{\operatorname{argmin}} \; -\ln p(\boldsymbol{\mathcal{X}}|\boldsymbol{\mathcal{Z}}) = \underset{\boldsymbol{\mathcal{X}}}{\operatorname{argmin}} \sum_i \|h_i(\boldsymbol{\mathcal{X}}) - \mathbf{z}_i\|^2_{\Sigma_i}, \tag{1.4}$$

where h_i is the measurement model that predicts the measurement given the current state estimate, and Σ_i is the covariance matrix associated with the measurement noise. To solve (1.4), we linearize the measurement models about the current estimate using a first-order Taylor series expansion. This yields a linear least squares problem

$$\underset{\boldsymbol{\mathcal{X}}}{\operatorname{argmin}} \sum_i \|\mathrm{H}_i \boldsymbol{\mathcal{X}} - \mathbf{z}_i\|^2_{\Sigma_i}, \tag{1.5}$$

where H_i is the Jacobian of the measurement model with respect to the state. The optimal solution to the linear problem is found by solving the normal equations. Note that in full SLAM and pose SLAM, each measurement function, $h_i(\boldsymbol{\mathcal{X}})$, is independent and will only depend on a small subset of $\boldsymbol{\mathcal{X}}$. This results in an inherent sparsity in the linear least squares problem, which is exploited by modern solvers to greatly reduce the computational complexity of the optimization problem (§1.2.2). To solve the full nonlinear problem, (1.4) is repeatedly linearized and solved until convergence. It is important to note that this does not guarantee a global optimum—if the initial linearization point is not sufficiently close to the global optimum the optimization may converge to a local minima. However, in practice, a good initial linearization point is usually available from odometry and by incrementally building and solving the SLAM problem as the robot explores.

1.1.1.1 Graphical Representation

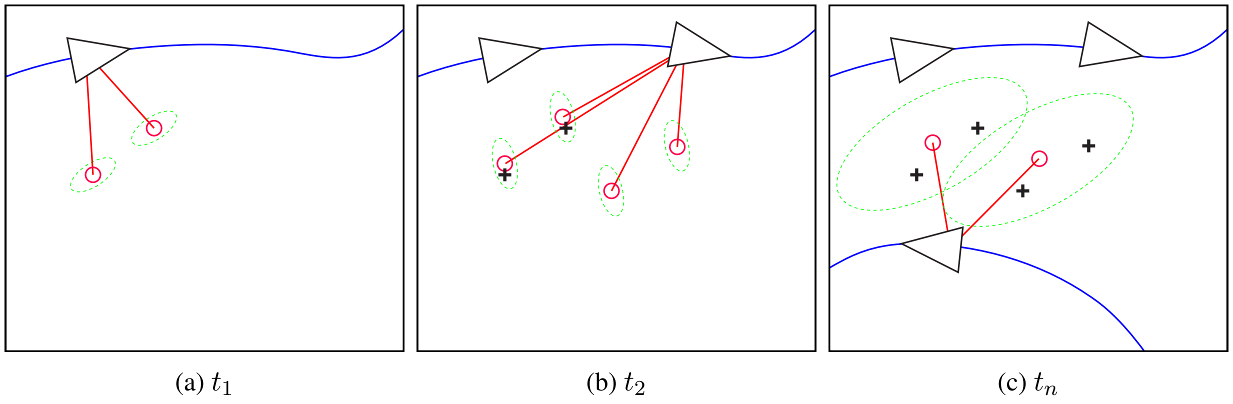
The SLAM problem is commonly represented as one of three graphs, each of which emphasize different aspects of the problem [45]. The first is a directed acyclic graph referred to as a Bayes net. The Bayes net encodes the conditional dependencies in the SLAM problem and is commonly associated with filtering techniques, as the conditional dependencies arise from temporal Markov properties. The second is an undirected graph, the Markov random field (MRF). The connectivity of the MRF also encodes some of the conditional independence properties of the distribution. Additionally, for Gaussian distributions, the MRF illustrates the adjacency structure of the associated information matrix, and therefore, provides insight into the sparsity structure of the problem. The third is the factor graph, a bipartite graph consisting of nodes that represent variables, and factors that represent measurement potentials over the variables. Like an MRF, a factor graph encodes the conditional independence properties of the distribution and illustrates the adjacency structure of the information matrix. Additionally, it explicitly defines a factorization of the distribution, clearly illustrating which nodes support each measurement observation function h_i. If we represent each factor in the graph as $\psi_i(\mathbf{x}_i)$, where $\mathbf{x}_i \subset \boldsymbol{\mathcal{X}}$ is the subset of variables that support the factor ψ_i, then

$$\hat{\boldsymbol{\mathcal{X}}} = \underset{\boldsymbol{\mathcal{X}}}{\operatorname{argmin}} \sum_i \|h_i(\boldsymbol{\mathcal{X}}) - \mathbf{z}_i\|^2_{\Sigma_i} = \underset{\boldsymbol{\mathcal{X}}}{\operatorname{argmin}} \sum_i \psi_i(\mathbf{x}_i). \tag{1.6}$$

In this thesis we focus on the factor graph representation of the SLAM problem as it aids the presentation of the proposed algorithms.

1.1.2 Data Association and the "Front-End"

Figure 1.2 Illustration of data association and loop-closure in the full SLAM formulation. At time t_1, the robot observes and instantiates two new landmarks. At time t_2, the robot observes four landmarks—two of which can be associated with existing landmarks, while the other two instantiate new landmarks. At time t_n, the robot revisits the previously explored area and observes two landmarks. Because a large amount of uncertainty has accumulated between the robot's pose and the landmark locations (illustrated by the green ellipses) the robot is unsure which landmarks it is observing. This problem is often referred to as loop-closure.


(a) t_1 (b) t_2 (c) t_n

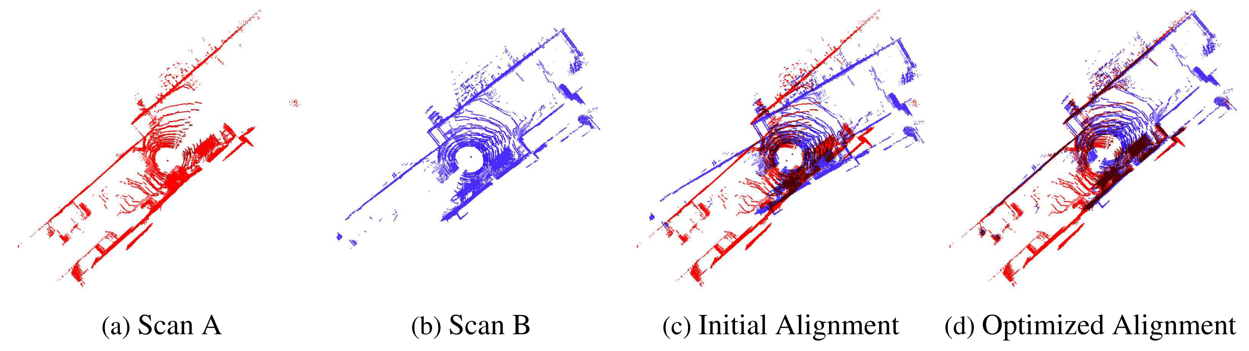
So far we have focused on the underlying estimation problem in SLAM. A complete SLAM system, however, must also generate the constraints in the optimization problem from noisy sensory observations of the environment. This is often referred to as the SLAM front-end, with the associated optimization referred to as the back-end.

Unlike the SLAM back-end, which can be generic, the SLAM front-end is highly dependent on the application and the sensing modality. In the full SLAM formulation, the front-end must repeatably extract landmarks from raw data and be able to establish correspondence between previously-viewed landmarks. In the pose SLAM formulation, the front-end must identify pairs of poses with overlapping views of the environment and then register the views to produce a relative constraint. The process of associating extracted landmarks, or associating views, is referred to as data association, and is a critical function of the front-end.

Data association in the full SLAM formulation is illustrated in Fig. 1.2. When a good motion prior is available from the current SLAM estimate (Fig. 1.2(b)), data association is often straightforward as false matches can be easily rejected based upon the motion prior (e.g., Mahalanobis gating [127]). However, when a strong prior is not available, which often happens when the robot returns to a previously explored location after a long period of time, data association becomes much more challenging. This is referred to as the loop-closure problem (Fig. 1.2(c)). Even though we illustrate the data association problem using the full SLAM formulation, similar challenges are also present in the pose and landmark SLAM formulations. In this thesis, we focus on SLAM front-ends that use two sensing modalities, LIDAR and vision, both of which are commonly used in robotics.

1.1.2.1 LIDAR Scan Matching

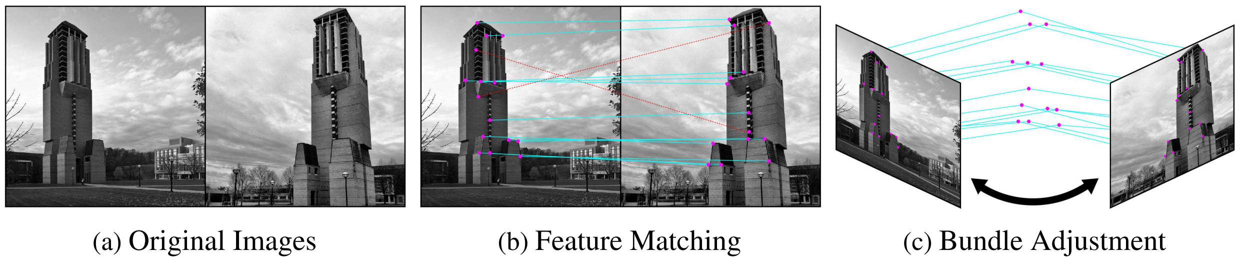
Figure 1.3 LIDAR scan matching illustration. Two 3D LIDAR scans collected by the robot as it moves through the environment are shown in (a) and (b). An initial guess of the robot's motion between scans is then used to roughly align the scans (c). The optimal estimate of the robot's motion is then found by registering the two scans using iterative methods (d).


(a) Scan A (b) Scan B (c) Initial Alignment (d) Optimized Alignment

LIDAR uses time-of-flight laser measurements to sample the 3D structure of an environment. LIDAR can be used to capture a 2D slice of the environment or a full 3D point cloud depending upon the sensor configuration. Two scans taken at different locations can then be registered using iterative methods such as [13, 148]. This produces a relative-pose constraint that can be used in the pose SLAM formulation. Iterative LIDAR registration methods require a good initial guess of the relative transform between the two poses as convergence to an incorrect local minimum can be a problem. An illustration of this process is shown in Fig. 1.3.

1.1.2.2 Vision

Figure 1.4 Image registration illustration. Two images to register are shown in (a). Feature points are extracted from both images, shown as magenta points in (b). These features are matched based on their vector descriptors (lines in (b)) and outliers are rejected based on geometric verification (red lines in (b)). Finally, bundle adjustment is used to optimize the robot's motion and the 3D location of the feature points (c).

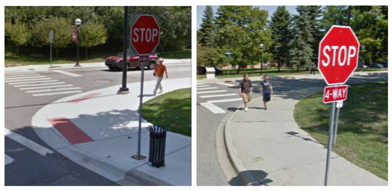


Vision-based SLAM front-ends commonly perform feature-based image registration in order to produce constraints for both full SLAM and pose SLAM. A good overview of standard feature-based image registration is available in [68]. Given a set of images to register, feature points that can be repeatedly detected under scale, rotation, and viewpoint changes [9, 104], are first extracted from each image. The visual appearance of each feature point is described as a vector in a high dimensional space. Feature matching is then performed using nearest-neighbors in feature space. The initial feature matches are geometrically verified to remove outliers, often by using random sample consensus (RANSAC) [55] to fit a projective model. In the case of full SLAM, the location of triangulated image features can be used as landmarks. This SLAM formulation is also referred to as bundle adjustment [68, 167] and is well studied in the computer vision community. In the case of pose SLAM, a small bundle adjustment problem is often used over a pair of images as a final refinement stage to produce a relative constraint between robot poses. An illustration of this process is shown in Fig. 1.4.

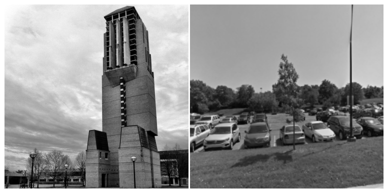
1.1.2.3 Data Association Challenges

Data association is a core challenge of all SLAM systems. Some of the most common challenges for visual data association are illustrated in Fig. 1.5—though not universal, many of the same challenges occur for other sensing modalities. Data association is further complicated by the fact that, because Gaussian noise models are assumed, the SLAM optimization problem is highly sensitive to outlier measurements. Therefore, it is very important that data association avoid false positive matches.

Figure 1.5 Data association challenges. In short-term SLAM applications, the front-end must overcome several challenges including perceptual aliasing, varying measurement saliency and short-term dynamic objects like people and cars. Long-term SLAM is further complicated by longer-term dynamics like lighting, changing seasons, and even structural changes caused by construction. All of these examples occur in the North Campus dataset (Appendix A).

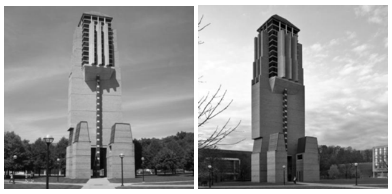


(a) Perceptual Aliasing

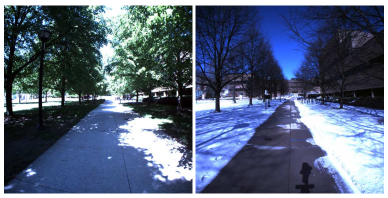


(b) Saliency

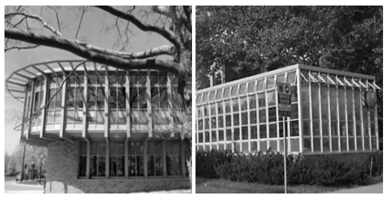
(c) Dynamic Objects



(d) Changing Lighting



(e) Changing Seasons



(f) Structural Changes

In short-term SLAM applications, the front-end must overcome several challenges. The front-end must be robust to perceptual aliasing, i.e., the fact that different places may have a similar appearance and could be confused based on their appearance alone (Fig. 1.5(a)). Additionally, not all sensory information is equally valuable or "salient" for map building and localization—an image of the North Campus clock tower is very useful for localization, while an ambiguous image of a parking lot provides very little help if you are lost (Fig. 1.5(a)). Perceptual aliasing and saliency are especially challenging when solving the loop-closure problem, as loop-closure requires that the appearance of the environment be sufficiently informative so that the true association can be identified and incorrect associations rejected. Finally, fast moving objects like people and cars (Fig. 1.5(c)) violate the static world assumption in most SLAM systems and can temporarily occlude the sensor's field of view.

Long-term SLAM is further complicated by medium- and long-term dynamics including changes in lighting (Fig. 1.5(e)), changing seasons (Fig. 1.5(d)), and even structural changes caused by construction (Fig. 1.5(f)). The North Campus dataset (Appendix A) contains many examples of each of these challenges, making it a valuable tool to evaluate the algorithms proposed in this thesis.

1.2 Literature Review

In this section, we provide a review of the literature related to this thesis, including some of the many applications of SLAM, how the SLAM problem has been formulated and solved, and works that seek to extend the capabilities of SLAM systems in challenging scenarios. Additionally, we consider work related to LIDAR- and vision-based SLAM front-ends.

1.2.1 Applications of SLAM

Many of the most promising robotic applications seek to explore and manipulate environments that are difficult or dangerous for humans to access. Examples include scientific data collection in the ocean [53, 82, 179] or outer space [61], autonomous inspection of critical infrastructure [71, 89, 136, 144], or search and rescue after a disaster [1, 39, 147]. In other applications, we simply want robotics to assist us in our daily lives, from self-driving cars [6, 16, 102, 107, 109, 115, 163, 169, 183], to assistive robots for the elderly or disabled [123, 124, 141, 155], and even automated vacuums [33, 76].

The role of SLAM varies in each of these applications. In some cases, SLAM runs online as the robot explores the environment [71, 89, 136]. In others, SLAM is used to post-process the data collected by the robot to produce environment models and visualizations [53, 82, 179]. In the case of self-driving cars, modern systems [107, 183] use SLAM to build a prior map offline. Then, during operation, the autonomous car localizes itself into the prior map [29, 103, 182], allowing the vehicle to access prior information such as the road network topology, speed limits, and the location of traffic signals. Regardless of how it is used, SLAM is a key component in these applications and many more like them.

Beyond robotics, bundle adjustment [167]—a variant of SLAM where poses represent the position of cameras and the features are triangulated image key points—has been well studied in the computer vision community where it is used for many tasks including large-scale reconstructions from internet photo collections [3, 58, 131, 153, 154].

1.2.2 Solving SLAM

Over the last several decades many methods have been proposed for solving the various formulations of SLAM illustrated in Fig. 1.1.

1.2.2.1 Filtering Methods

Early SLAM solutions sought to solve the landmark SLAM formulation (Fig. 1.1(d)) in a recursive Bayesian estimation framework. These filtering methods, typically based on the extended Kalman filter (EKF) [17, 43, 152], were shown to be successful in small environments with few landmarks. However, as the number of landmarks in the environment grows, maintaining the mean vector and (dense) covariance matrix of the state quickly becomes computationally intractable. Careful landmark selection, as proposed by Davison and Kita [42], can delay computational issues and allow for real-time operation in small environments. Dividing the environment into computationally feasible submaps [17] was also proposed as a solution at the expense of additional complexity of maintaining multiple maps and estimating the relative positions of each map.

Careful consideration of the SLAM problem revealed that in almost all applications, the relationship between robot states and map landmarks is inherently sparse. Sensors used to perceive the environment have a finite field of view and landmarks in different parts of the environment are only weakly correlated through the robot's previous poses. This is referred to as sparsity, because in the natural parametrization of the Gaussian distribution, which we refer to as the information form, the information matrix will have many off-diagonal values very close to zero. In the case of landmark SLAM (Fig. 1.1(d)), these off-diagonal entries will not be exactly zero, and therefore, the information matrix is not truly sparse. Thrun et al. [161] proposed to force some elements to be exactly zero, inducing sparsity in the information matrix and allowing efficient optimization using an extended information filter (EIF), the information-form dual of the EKF. Unfortunately, this sparsification method causes the resulting estimate to be overconfident as shown by Eustice et al. [50]. For feature-based SLAM, Walter et al. [176] ensure sparsity in an EIF by dropping odometry constraints and re-localizing based on features.

With this focus on sparsity, it was soon realized that it is the marginalization of past poses in the landmark SLAM formulation (Fig. 1.1(d)) that causes the loss of sparsity. Both full SLAM and pose SLAM are naturally sparse because past poses are maintained in the optimization problem. This is clearly reflected in the factor graphs of the formulations, depicted in Fig. 1.1. In the context of filtering, this is exploited by Eustice et al. [52] in a pose SLAM EIF framework, which greatly increased the possible scale of filtering-based SLAM solutions compared to those solvable with a dense EKF.

Even with the increased scale provided by sparse EIF methods, all filtering methods still suffer from a significant short coming—as each new measurement is incorporated into the filter they commit to a static linearization point. This can cause the solution to diverge as linearization errors accumulate [8, 31, 84].

1.2.2.2 Optimization-based methods

Most recently, optimization-based methods [45, 46, 64, 65, 77, 86, 88, 96, 99, 105, 129, 130, 134, 140] have been proposed that explicitly treat the SLAM problem as a large, sparse, nonlinear least-squares problem. These methods can efficiently solve both the full SLAM (Fig. 1.1(b)) and pose SLAM (Fig. 1.1(c)) formulations by exploiting state-of-the-art sparse linear algebra solvers. By maintaining all nonlinear measurements, these algorithms can relinearize repeatedly as new observations are made. By exploiting the inherent sparsity of the optimization problem, these methods are capable of efficiently solving very large SLAM problems. These methods represent the current state-of-the-art for large-scale, metric SLAM.

Given all the measurements and an initial guess of the parameters, many of these methods treat SLAM as a large batch optimization problem [45, 99, 105, 129, 130, 134]. This is a fast and efficient method of solving the problem, but it is best suited for offline SLAM as the whole batch optimization must be repeatedly performed as new measurements are obtained for online SLAM.

To better adapt optimization-based SLAM for online applications, several method have been proposed [64, 65, 86, 88, 96, 140] that incrementally update the least-squares solution. This allows for faster access to the SLAM solution after adding new measurements and is most appropriate for applications that require online SLAM.

For the majority of these methods, an efficient solution is possible because of the inherent sparsity in many real-world SLAM problems. To address the problem of graphs with many loop-closures, and therefore reduced sparsity, Dellaert et al. [46] and Jian and Dellaert [77] have proposed methods that first solve a sparse graph preconditioner, and then use an iterative method, like conjugate gradients, to solve the full, less-sparse, problem.

Ni et al. [130] and Ni and Dellaert [129] present methods to divide the graph into subgraphs that can be optimized independently before optimizing the relationships between each graph. This allows for the optimization of very large graphs as the entire graph does not need to be loaded into memory at once.

One important consideration in all SLAM systems is the parameterization of the variables to be estimated. The most common parameterization represents each pose and landmark in a common "world" or "local" reference frame. Several methods have proposed a relative parameterization [110, 134, 150], reporting improved scalability and speed of convergence during optimization. Kim et al. [91] propose the use of "anchor nodes," which encode the transformation between local frames. Introducing anchor nodes to the graph allows multiple maps—from different robots, or different mapping sessions—to be parameterized in their own local frame, while still allowing for observations between nodes in different frames.

1.2.2.3 Additional SLAM Formulation and Methods

Several other important classes of SLAM systems have been proposed beyond the metric filtering- and optimization-based SLAM solutions that we focus on in this thesis.

Monte Carlo Methods One important class of methods used to solve SLAM are Monte Carlo methods employing particle filters, such as [119, 121]. These methods can model multi-modal probability distributions, and therefore, allow multiple hypothesis tracking. However, the number of particles required to sufficiently sample the state space increases with the scale of the environment and it is unclear how to ensure a sufficient level of particle variety through the mapping process.

Occupancy Grids Many indoor applications elect to use an occupancy grid [49, 122] representation of the environment during SLAM. In this formulation, the environment is discretized into a grid and a binary variable is estimated indicating if a cell is free or occupied. The main benefit of this approach is that the map can then immediately be used for planning and other higher-level tasks. Unfortunately, discretized grids do not scale well to large environments.

Topological SLAM Several authors have proposed SLAM systems that do not attempt to optimize a fully metric map. These methods instead attempt to estimate a topological or hybrid metric-topological representation of the environment [10, 38, 98, 114]. Topological maps are especially well-suited for applications that require a more semantic understanding of the environment, i.e., the user tells the robot to go to the end of the hall and turn right. However, they are not sufficient for tasks where a metric understanding of the environment is important, for example, area coverage tasks like [32, 90], spatial motion planning, or geo-referencing scientific data.

Continuous-Time SLAM In situations where sensor data is collected at a high rate, it may be computationally infeasible to represent the robot's trajectory using a set of discrete poses. In continuous-time SLAM methods [4, 62], the trajectory is represented by a weighted set of basis functions. This allows one to make measurements at any point in time while still only optimizing over the finite set of basis weights.

1.2.3 Robust Optimization for SLAM

Standard SLAM optimization methods are highly sensitive to outlier measurements because the assumed Gaussian noise models assign extremely low probability to measurements in the "tails" of the distribution. Under these assumptions, it is very important that data association have zero false positive matches. Clearly, this is an unrealistic requirement, especially in the long-term where the changes of making an erroneous loop-closures grows with time.

Recently, outlier-robust SLAM optimization has received a good deal of attention with several methods [2, 133, 158] producing very promising results. Sunderhauf and Protzel [158] proposed the addition of binary "switchable constraint" variables, which estimate if each link is an inlier or outlier. This method produces good results at the expense of introducing many additional variables to the SLAM problem. By marginalizing out the switching variables, Agarwal et al. [2] proposed an M-estimator [74] referred to as "dynamic covariance scaling" that also provides similar performance without the additional optimization variables. Olson and Agarwal [133] propose modeling constraints with a max-mixture of Gaussian distributions. Like [2, 158], max-mixtures allow one to model binary inlier-outlier factors, with one component representing the measure and another representing a high-variance null hypothesis. However, unlike [2, 158], max-mixtures can also model other more complex multi-modal measurements.

In extreme cases, these methods can converge to the correct graph in the presence of almost as many outliers as inliers. However, this is not normally necessary in practice. Instead, these methods allow one to relax the precision requirements on the front-end so that in the small, yet non-zero, chance an outlier measurement is integrated in the graph, it will not corrupt the solution.

1.2.4 Long-Term SLAM in Dynamic Environments

Long-term applications of SLAM complicate both the back-end and front-end of the SLAM system. In terms of the back-end, the primary concern is computational complexity, while for the front-end the primary concern is data association in the presence of large dynamic changes.

1.2.4.1 Controlling the Computational Complexity of Long-Term SLAM

Many methods have been proposed that seek to reduce or bound the computational complexity of the SLAM optimization problem. Methods have been proposed that enforce sparsity [161], slow the rate of graph growth by only adding the most informative nodes and edges [75], and avoid adding new nodes in previously explored locations [78]. Node removal, through marginalization or an approximation of marginalization, has been proposed in [48, 56, 73, 94, 97, 108, 174, 178].

These works are discussed in detail in the context of our proposed node removal method in Chapter II and our proposed complexity management schemes in Chapter IV.

1.2.4.2 Dealing with Dynamic Environments in SLAM

Many methods have been proposed that try to filter out the dynamic elements of the environment while maintaining the assumption that the underlying environment is static, for example [22, 57, 67, 177], among others. It has also been shown that it is possible to explicitly identify and track dynamic objects in the environment, either for specific classes of objects, such as people [120], or *a-priori*-unknown dynamic objects [118]. Most promising are methods that explicitly model the dynamic environment in the map representation [14, 15, 36, 44, 94, 156, 164]. Among these, the "exemplar"-based methods [36, 94, 156], which capture a location's change in appearance using a set of example views, are most relevant to this thesis. We discuss these works in detail in the context of the proposed SLAM systems in Chapter IV.

1.2.5 Data Association for Vision and LIDAR

The methods used to derive measurements from sensory data vary significantly between different modalities. Here, we provide a brief overview of methods used to extract measurements from cameras and LIDARs.

1.2.5.1 Deriving Observations from Computer Vision

As discussed in §1.1.2.2, two or more images of a scene can be used estimate the structure of the scene and the motion of the camera between images. It is important to note that because cameras only capture a 2D representation of the world, the geometric registration will not constrain the scale of the scene unless additional information is provided, e.g., from odometry or a known stereo baseline [68].

Visual Feature Descriptors A wide variety of visual feature descriptors have been proposed including SIFT [104], SURF [9] and DAISY [165], among many others. Additionally, many authors have proposed methods that leverage machine learning to improve the performance of visual feature descriptors [5, 20, 72, 142, 168, 180, 181]. We discuss these works in detail in Chapter III, where we propose a method to learn visual feature descriptors that are more robust to changes in lighting.

Place Recognition Proposing sets of images that may be of the same location can be done using the current SLAM state estimate. However, for loop-closure, place recognition [38, 113, 126, 132, 151] is commonly used to propose loop-closures based strictly on the visual content of the image. Many of these methods [38, 132, 151] are based on the "bag-of-words" model, which represents each image as a histogram of visual-word occurrence counts over a quantized visual vocabulary. Recently, several methods [112, 126] have produced very good results on challenging datasets by exploiting the coherence of temporal sequences to boost place recognition performance.

Visual Data Association in Dynamic Environments Several methods have been proposed to improve the robustness of visual data association specifically in dynamic environments [24, 37, 80, 81, 94, 100, 128, 142]. We discuss these works in detail in the context of our proposed robust visual feature descriptors in Chapter III.

1.2.5.2 Deriving Observations from LIDAR

Unlike vision, LIDAR allows the robot to directly observe the 3D structure of the environment. Additionally, because LIDAR is an active sensor, it is not affected by changes in lighting. Unfortunately, LIDAR scanners are currently much more expensive than cameras.

In this thesis, we use iterative alignment methods [13, 106, 148] to derive measurements for LIDAR scans. These methods start with an initial guess of the rigid-body transform between the two scans. This transform is used to assign correspondence between the points in the two scans. Based on these correspondences, the transform that best aligns the scans is found. A new set of correspondences is then found and the process is repeated until convergence. These iterative methods are sensitive to their initialization and can get stuck in local minima. However, if well-initialized, they provide very accurate measurements.

1.3 Thesis Objective and Contributions

The objective of this thesis is to extend the capabilities of SLAM systems operating autonomously for long-term periods of time in dynamic environments—to do so requires addressing two core problems:

1. The computational complexity of the SLAM optimization problem must not grow unbounded with time. As described in §1.1, many state-of-the-art graph SLAM formulations require that nodes be continuously added to the graph for the robot to stay localized and preserve problem sparsity.

2. The SLAM front-end must continue to function as the environment changes with time. While certain short-term and small-scale dynamic changes can be considered as noise within the SLAM front-end, truly long-term SLAM requires a front-end that explicitly accounts for dynamic changes in the environment. This is especially true for vision-based SLAM where even the change in lighting between morning and evening may be enough to break state-of-the-art systems.

Toward this objective we have produced the following contributions:

1. Collected a long-term dataset sufficient to evaluate the proposed methods.
2. Developed a mathematical method to control the computational complexity of the SLAM optimization problem through node removal.
3. Presented a method to learn visual feature descriptors that are more robust to dynamic changes in lighting.
4. Developed and experimentally validated SLAM systems using LIDAR and vision front-ends capable of long-term exploration of dynamic environments.

Each of these contributions is described in the following sections.

1.3.1 Long-Term Dataset Collection

Evaluation of the proposed SLAM system requires a long-term dataset with significant dynamic variation. An appropriate dataset is not currently available to the research community, therefore, we have collected a challenging dataset on the University of Michigan's North Campus. We collected imagery and LIDAR data from January 2012 to April 2013 using our Segway robotic platform (Fig. A.1(a)). In addition to allowing us to throughly evaluate the proposed algorithms throughout this thesis, we plan to release this dataset to the community. **The North Campus dataset is described in detail in Appendix A**.

1.3.2 Complexity Control Through Node Removal

Though other works have proposed using node removal to control the computational complexity of SLAM, nearly all rely on measurement composition to produce a new set of factors over the marginalization clique [48, 79, 94, 97, 174, 178]. This is problematic for two reasons. First, the new factors produced through measurement composition are not independent in general and treating them as such produces inconsistent estimates. Second, for heterogeneous graphs with low-rank constraints (e.g., range-only or bearing-only measurements), measurement composition may not be well-defined. **In Chapter II, we propose a factor-based method for node removal in graph SLAM that addresses the shortcomings of measurement composition**. The proposed method, which we refer to as generic linear constraints (GLCs), is able to produce a new set of constraints over the marginalization clique that can represent either the true marginalization, or a sparse approximation of the true marginalization.

1.3.3 Learning Robust Visual Feature Descriptors

In many robotic applications, especially long-term outdoor deployments, the success or failure of feature-based image registration is largely determined by changes in lighting. **In Chapter III, we present a method to learn visual feature point descriptors that are more robust to changes in scene lighting than standard hand-designed features.** We demonstrate that, by tracking feature points in time-lapse videos, one can easily generate training data that captures how the visual appearance of interest points changes with lighting over time. This training data is used to learn feature descriptors that map the image patches associated with feature points to a lower-dimensional feature space where Euclidean distance provides good discrimination between matching and non-matching image patches.

1.3.4 Long-Term SLAM in Dynamic Environments

In Chapter IV, we propose LIDAR- and vision-based SLAM systems capable of long-term operation in dynamic environments. These systems leverage the proposed GLC node removal to control the computational complexity of the graph over time and to actively preserve a set of example views for each location. The vision-based system additionally uses the learned feature descriptors to better constrain the SLAM graph.

CHAPTER II

Generic Linear Constraint Node Removal

Standard graph-based simultaneous localization and mapping (SLAM) formulations are not ideal for use in the long-term as the size of the graph, and therefore the computational complexity of its associated optimization problem, grows with exploration time, regardless of the size of the environment. In this chapter, we present a factor-based method for node removal in graph SLAM, which can be used to control its computational complexity. The proposed method is able to produce a new set of constraints over the elimination clique, which can represent either the true marginalization or a sparse approximation of the true marginalization. The proposed algorithm improves upon commonly used node-removal methods in two key ways: First, it is not strictly limited to full-state relative-pose constraints and works equally well with other low-rank constraints, such as those produced by monocular vision. Second, the new factors are produced in a way that accounts for inter-measurement correlation, a problem in other methods that rely upon measurement composition.

We propose several alternatives for the sparse approximation of the dense potentials induced by node marginalization in SLAM factor graphs. First, we present the Chow-Liu tree approximation, which provides the minimum Kullback-Leibler divergence (KLD) but may be overconfident. We then present a collection of optimization-based methods for producing a guaranteed-conservative sparse approximation. These methods start with a sparse, but potentially overconfident, Chow-Liu tree approximation of the marginalization potential, and then adjust the approximation with the objective of achieving a low KLD from the true marginalization potential subject to a constraint that the approximation is conservative.

We evaluate the proposed node removal methods over multiple real-world SLAM graphs and show that they outperform commonly used methods in terms of Kullback-Leibler divergence. This chapter is based on our work published in [25, 27, 30].

2.1 Introduction

Graph based SLAM [45, 52, 86, 93, 105, 134, 160] has been demonstrated successfully over a wide variety of applications. Unfortunately, the standard graph SLAM formulation, which maintains *all* past robot states, is not ideal for long-term applications. The robot must continually add nodes and measurements to stay localized even if it is working in a finite region, causing the size of the graph to grow with both spatial extent and exploration time (Fig. 1.1(b) and (c)).

In this chapter, we address this challenge by developing a principled and generic method that allows one to arbitrarily remove nodes from the graph, thereby reducing inference complexity and allowing for graph maintainability. We refer to this method as generic linear constraints (GLCs). The method is illustrated in Fig. 2.1. The proposed algorithm was designed to address the pitfalls of existing node removal and sparsification techniques—particularly those based on measurement composition [48, 79, 94, 97, 174, 178]. The algorithm was designed so that it meets the following criteria:

- The algorithm works equally well with non-full-state constraints. Constraints with lower degree of freedom (DOF) than full-state (e.g., bearing-only, range-only and partial state constraints) are handled under the same framework as full-state constraints, without special consideration.

- The new factors are produced in a way that does not double count measurement information. As we will show in §2.2, methods based on the pairwise composition of measurements produce pairwise constraints that are not independent, which leads to inconsistency in the graph.

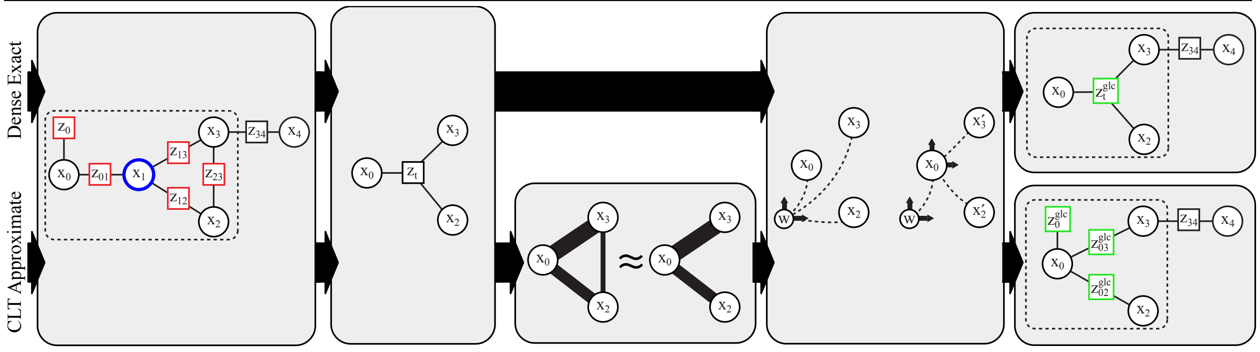
- The algorithm produces a new set of independent factors using the current graph factors as input. The method does not require the linearized information matrix of the entire graph as input.

- The algorithm is able to produce constraints that can represent exact node marginalization or a sparse approximation of the dense marginal using either a Chow-Liu tree (which minimizes the KLD from the dense marginal but may be slightly overconfident), or a guaranteed-conservative tree (which provides a low KLD while preventing overconfidence).

- The computational complexity of the algorithm is dependent only upon the number of nodes and factors in the elimination clique, not on the size of the graph beyond the clique.

- The algorithm does not require committing to a world-frame linearization point, rather, the new factors are parametrized in such a way as to use a local linearization that is valid independent of the global reference frame. This allows for the exploitation of methods that re-linearize during optimization (e.g., [45, 86, 134]).

Figure 2.1 The GLC node removal algorithm. The dense exact version follows the top row, while the sparse approximate versions follows the bottom row. A sample factor graph where node $\mathbf{x}_1$ is to be removed is shown in (a). Here $\mathbf{X}_m = [\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3]$. The factors $\mathbf{Z}_m = [\mathbf{z}_0, \mathbf{z}_{01}, \mathbf{z}_{12}, \mathbf{z}_{13}, \mathbf{z}_{23}]$ (highlighted in red in (a)) are those included in calculating the target information, Λ_t, which defines a linear potential, $\mathbf{z}_t$, over the marginalization clique $\mathbf{X}_t = [\mathbf{x}_0, \mathbf{x}_2, \mathbf{x}_3]$ (b). In the case of sparse-approximate node removal, the original distribution associated with the target information, $p(\mathbf{X}_t|\mathbf{Z}_m)$, is approximated using a sparse spanning tree (computed using either the Chow-Liu approximation or the proposed guaranteed-conservative approximations) as $p(\mathbf{x}_0|\mathbf{Z}_m)p(\mathbf{x}_2|\mathbf{x}_0, \mathbf{Z}_m)p(\mathbf{x}_3|\mathbf{x}_0, \mathbf{Z}_m)$ (c). Optionally, the potentials are reparameterized with respect to $\mathbf{x}_0$ to avoid linearization in the world-frame (d). New GLC factors are computed and inserted into the graph replacing $\mathbf{Z}_m$ (highlighted in green in (e)). Note that node removal only affects the nodes and factors within the Markov blanket of $\mathbf{x}_1$ (dashed box).



(a) Original Graph
(b) Target Info.
(c) Sparse Approximation
(d) Reparameterize (optional)
(e) Final Graph

2.1.1 Related Work

A wide variety of methods have been proposed to control the computational complexity of long-term SLAM—ranging from methods that simply slow the growth of the graph, to methods that actively remove edges (sparsification) and nodes (marginalization).

2.1.1.1 Slowing the Growth of Long-Term SLAM

Several prior methods have been proposed to slow the rate of growth of the graph. In Ila et al. [75], an information-theoretic approach is used to add only non-redundant nodes and highly-informative measurements to the graph. This slows the rate of growth but does not bound it. In Johannsson et al. [79], new constraints are induced between existing nodes when possible, instead of adding new nodes to the graph. In this formulation the number of nodes grows only with spatial extent, not with mapping duration—though the number of factors and connectivity density within the graph still grow with time.

2.1.1.2 Graph Sparsification: Removing Edges from the Graph

Graph sparsification methods that seek to remove edges from the graph in order to increase its sparsity include [161, 173, 176]. These methods work directly on the linearized information matrix and are therefore, best suited for filtering-based SLAM solutions. In Thrun et al. [161], weak links between nodes are removed to enforce sparsity. Unfortunately, this removal method causes the resulting estimate to be overconfident [50]. In Walter et al. [176], odometry links are removed in order to enforce sparsity in feature-based SLAM. Recently, Vial et al. [173] proposed an optimization-based method that minimizes the KLD of the information matrix while enforcing a sparsity pattern and the requirement that the estimated information is conservative. This method performs favorably in comparison with [161] and [176], but requires a large matrix inversion in order to reduce the scope of the optimization problem, limiting its online utility.

2.1.1.3 Controlling Computational Complexity by Removing Nodes

Recently, many works have proposed removing nodes from the SLAM graph as a means to control the computational complexity of the associated optimization problem [48, 94, 97, 174, 178]. In Konolige and Bowman [94], the environment is spatially divided into neighborhoods and then a least-recently-used criteria is used to remove nodes with the goal of keeping a small set of example views that capture the changing appearance of the environment. In Kretzschmar and Stachniss [97], nodes that provide the least information to an occupancy grid are removed. In Eade et al. [48], nodes without associated imagery are removed. In Walcott-Bryant et al. [174], "inactive" nodes that no longer contribute to the laser-based map (because the environment has changed) are removed. Finally, in Wang et al. [178], nodes are removed based on an approximation of their information contribution to the graph.

Each of the methods described in [48, 79, 94, 97, 174, 178] provides insight into the question of which nodes should be removed from the graph; however, they all rely upon pairwise measurement composition over full-state constraints, as described in Smith et al. [152], to produce a new set of factors over the elimination clique after a node is removed from the graph. Unfortunately, pairwise measurement composition has two key drawbacks when used for node removal. First, it is not uncommon for a graph to be composed of many different types of low-rank constraints, such as bearing-only, range-only and other partial-state constraints. In these heterogeneous cases, measurement composition, if even possible, quickly becomes complicated as the constraint composition rules for all possible pairs of measurement types must be well-defined. Second, the new constraints created by measurement composition are often correlated. Ignoring this correlation leads to inconsistent estimates because measurements are double counted. In some cases it is possible to avoid double counting measurements by discarding some of the composed measurements; however, this comes at the cost of information loss. Additionally, for general graph topologies, double counting measurements may be unavoidable when using a pairwise composition scheme as illustrated by the simple graph in Fig. 2.2.

The exact procedure for measurement-composition-based node removal varies amongst existing methods. In [48, 94, 97] the correlation between composed measurements is ignored. In Konolige and Bowman [94], all composed constraints are kept, causing fill-in within the graph. In order to preserve sparsity, a subset of the composed edges are pruned by Eade et al. [48] using a heuristic based on node degree. In Kretzschmar and Stachniss [97], composed-edge removal is guided by a Chow-Liu tree calculated over the conditional information of the elimination clique. To avoid measurement double counting, Johannsson et al. [79] discard an odometry link and performs re-localization (along the lines of [176]). Similarly, Walcott-Bryant et al. [174] use a maximum of two newly composed constraints at the beginning and end of a "removal chain" (a sequence of nodes to remove) to ensure connectivity without double counting measurements. In Wang et al. [178], only the two sequential odometry constraints are compounded, which also avoids double counting measurements.

Methods that remove nodes without measurement composition have been proposed in [25, 26, 56, 60, 73]. These methods are based upon replacing the factors in the marginalization clique with a linearized potential or a set of linearized potentials, instead of potentials produced by measurement composition. In Folkesson and Christensen [56], these linearized potentials are referred to as "star nodes." The dense formulation of our proposed GLC [25] is essentially equivalent to "star nodes" while the sparse approximate GLC replaces the dense n-nary connectivity with a sparse tree structure. In Frese [60], linearized potentials are used to remove nodes in cliques within the author's Treemap [59] algorithm. The method recently proposed in Huang et al. [73] uses dense linear potentials similar to star-nodes and dense-GLCs to remove nodes from the graph, and then later sparsifies the graph using an $\mathcal{L}_1$ regularized optimization.

The recent work by Mazuran et al. [108] replaces the elimination clique factors with a set of nonlinear virtual measurements and then uses a numerical optimization method to find the appropriate measurement noise for each virtual measurement. This method produces similar results to

GLC when removing nodes with a good linearization point, and improves the KLD when removing nodes with a poor linearization, as the measurement can subsequently be re-linearized. The authors also propose adding additional virtual factors in the elimination clique beyond a pairwise tree, which can further reduce the KLD. One limitation of the method is that it requires the specification of the virtual measurements and their Jacobians such that their rank is appropriate for the information in the marginalization potential. This is straightforward in homogeneous graphs with full-rank constraints; however, it is not clear how one would specify the virtual measurements in heterogeneous graphs with low-rank constraints. GLC automatically determines an appropriate linear model for each measurement, and works on heterogeneous graphs with low-rank constraints without special consideration.

Linearized potentials representing the result of marginalization are also used in several works [40, 166, 175] to reduce bandwidth while transmitting graphs between robots in a multi-robot distributed estimation framework. Nodes that are not part of the interaction between the robots' graphs are marginalized, producing linearized potentials. These linearized potentials are transmitted between robots.

2.1.1.4 Guaranteed-Conservative Graph Sparsification

Removing edges from the SLAM graph prevents the graph from capturing correlation between variables that may be correlated in the true distribution. This can result in estimates that are overconfident [50]. In most SLAM applications, conservative estimates are strongly preferred to overconfident estimates. During map building, overconfidence can adversely affect data association, causing the system to miss valid loop-closures. Additionally, when using the resulting map, overconfident estimates can lead to unsafe path planning and obstacle avoidance [162].

Recently, Vial et al. [173] and Huang et al. [73] have proposed methods that explicitly ensure conservative approximations during graph sparsification. In Vial et al. [173], an optimization-based method is proposed that, given a desired sparsity pattern, minimizes the KLD between the sparsified distribution and the true distribution while ensuring that the sparsified distribution is conservative. This method performs favorably in comparison with Thrun et al. [161] and Walter et al. [176], however, as Vial et al. [173] acknowledge, the computational cost of the optimization grows quickly with the size of the matrix being sparsified. To avoid this, they propose a problem reduction that allows their method to be applied to a subset of the graph's variables. The problem reduction still involves the expensive inversion of the block of the information matrix associated with the entire graph beyond the subproblem's Markov blanket, which will also be intractable for large graphs.

The method proposed by Huang et al. [73] performs node marginalization by inducing a densely connected linear factor as in [56] and our proposed dense-exact GLC. To perform edge sparsification, the authors formulate an optimization problem that seeks to minimize the KLD of the approximation while requiring a conservative estimate and encouraging sparsity through $\mathcal{L}_1$ regularization. This optimization problem is then applied to the linearized information matrix associated with the entire graph, which limits its applicability to relatively small problems, and prevents relinearization after sparsification. Using $\mathcal{L}_1$ regularization to promote sparsity is appealing because it does not require the sparsity pattern to be specified—instead, it automatically removes the least important edges. However, because the sparsity pattern produced is arbitrary, it is unclear how the resulting information matrix might be decomposed into a sparse set of factors, which is important if one wishes to exploit existing graph SLAM solvers such as iSAM [86, 87] or g^{2}o [99].

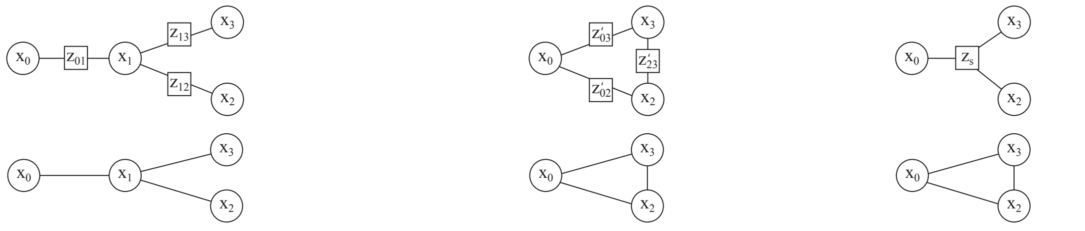
We explore additional optimization-based methods for conservative sparsification of the dense cliques induced by node marginalization. The proposed methods are integrated within the GLC framework and are designed to maintain the advantages of sparse-approximate GLC while addressing some of the aforementioned shortcomings of the methods proposed in Vial et al. [173] and Huang et al. [73]. Specifically, our methods do not require the full linearized information matrix as input nor do they have a computational complexity dependent on the size of the entire graph.

The remainder of this chapter is outlined as follows; In Section 2.2 we discuss the pitfalls associated with the use of measurement composition for node removal. Our proposed method is then described in Section 2.3 and experimentally evaluated in Section 2.4. In Section 2.5 we present methods to ensure conservative estimates during sparse-approximate node removal and evaluate them in Section 2.6. Finally, Sections 2.7 and 2.8 offer a discussion and concluding remarks.

2.2 Pairwise Composition is Not Marginalization

Consider the simple graph depicted in Fig. 2.2(a) where we show both its factor graph and Markov random field (MRF) representations. Suppose that we wish to marginalize node $\mathbf{x}_1$. Using the composition notation of Smith et al. [152], we can compose the pairwise measurements to

Figure 2.2 Measurement composition versus marginalization. Here node $\mathbf{x}_1$ is removed from the original graph (a). The top row shows the factor graph; the bottom row shows its Markov random field.



(a) Original graph (b) Composition (c) Marginalization

produce the graph depicted in Fig. 2.2(b) as follows,

$$\begin{aligned} \mathbf{z}'_{02} &= h_1(\mathbf{z}_{01}, \mathbf{z}_{12}) = \mathbf{z}_{01} \oplus \mathbf{z}_{12}, \\ \mathbf{z}'_{03} &= h_2(\mathbf{z}_{01}, \mathbf{z}_{13}) = \mathbf{z}_{01} \oplus \mathbf{z}_{13}, \\ \mathbf{z}'_{23} &= h_3(\mathbf{z}_{12}, \mathbf{z}_{13}) = \ominus\mathbf{z}_{12} \oplus \mathbf{z}_{13}. \end{aligned} \tag{2.1}$$

These composed measurements are meant to capture the fully connected graph topology that develops in the elimination clique once $\mathbf{x}_1$ has been marginalized. In [48, 97], this composition graph forms the conceptual basis from which their link sparsification method then acts to prune edges and produce a sparsely connected graph. The problem with this composition is that the pairwise edges/factors in Fig. 2.2(b) are assumed to be independent, which they are not.

It should be clear that the composed measurements in (2.1) are correlated, as $\mathbf{z}'_{02}$, $\mathbf{z}'_{03}$ and $\mathbf{z}'_{23}$ share common information (e.g., $\mathbf{z}'_{02}$ and $\mathbf{z}'_{03}$ both share $\mathbf{z}_{01}$ as input), yet, if we treat these factors as strictly pairwise, we are unable to capture this correlation. Now consider instead a stacked measurement model defined as

$$\mathbf{z}_s = \begin{bmatrix} \mathbf{z}'_{02} \\ \mathbf{z}'_{03} \\ \mathbf{z}'_{23} \end{bmatrix} = h\left(\begin{bmatrix} \mathbf{z}_{01} \\ \mathbf{z}_{12} \\ \mathbf{z}_{13} \end{bmatrix}\right) = \begin{bmatrix} \mathbf{z}_{01} \oplus \mathbf{z}_{12} \\ \mathbf{z}_{01} \oplus \mathbf{z}_{13} \\ \ominus\mathbf{z}_{12} \oplus \mathbf{z}_{13} \end{bmatrix}. \tag{2.2}$$

Its first-order uncertainty is given as

$$\Sigma_s = \begin{bmatrix} \frac{\partial \mathbf{z}'_{02}}{\partial \mathbf{z}_{01}} & \frac{\partial \mathbf{z}'_{02}}{\partial \mathbf{z}_{12}} & 0 \\ \frac{\partial \mathbf{z}'_{03}}{\partial \mathbf{z}_{01}} & 0 & \frac{\partial \mathbf{z}'_{03}}{\partial \mathbf{z}_{13}} \\ 0 & \frac{\partial \mathbf{z}'_{23}}{\partial \mathbf{z}_{12}} & \frac{\partial \mathbf{z}'_{23}}{\partial \mathbf{z}_{13}} \end{bmatrix} \begin{bmatrix} \Sigma_{01} & 0 & 0 \\ 0 & \Sigma_{12} & 0 \\ 0 & 0 & \Sigma_{13} \end{bmatrix} \begin{bmatrix} \frac{\partial \mathbf{z}'_{02}}{\partial \mathbf{z}_{01}} & \frac{\partial \mathbf{z}'_{02}}{\partial \mathbf{z}_{12}} & 0 \\ \frac{\partial \mathbf{z}'_{03}}{\partial \mathbf{z}_{01}} & 0 & \frac{\partial \mathbf{z}'_{03}}{\partial \mathbf{z}_{13}} \\ 0 & \frac{\partial \mathbf{z}'_{23}}{\partial \mathbf{z}_{12}} & \frac{\partial \mathbf{z}'_{23}}{\partial \mathbf{z}_{13}} \end{bmatrix}^\top .$$

Here we see that Σ_s will be dense in order to capture the correlation between the compounded measurements. Expressing this correlation requires a trinary factor with support including all three variables. Therefore, the joint composition in (2.2) produces the factor graph shown in Fig. 2.2(c).

It is this inability of pairwise factors to capture correlation between composed measurements that causes simple compounding to be wrong. Note that the graphs in Fig. 2.2(b) and Fig. 2.2(c) have the same Markov random field representation and information matrix sparsity pattern. The difference between the binary and trinary factorization is only made explicit in the factor graph representation. It is also interesting to note that even if we were to approximate the dense connectivity with a spanning tree constructed from binary factors, as in [97], the resulting estimate would still be inconsistent as any pair of factors are correlated.

These two observations; *(i)* that composed measurements are often correlated, and *(ii)* that representing the potential of an elimination clique with n nodes requires n-nary factors, will prove important in the remainder of this chapter.

2.3 Generic Linear Constraint Node Removal

The proposed method, illustrated in Fig. 2.1, is summarized as follows; First, the factors that are supported by the node to be removed and the nodes in its elimination clique (Fig. 2.1(a)) are used to compute the linear potential induced by marginalization over the elimination clique. This potential is characterized by its distribution's information matrix, which we refer to as the target information, Λ_t (Fig. 2.1(b)). Next, we use either *(i)* Λ_t directly to compute an exact n-nary potential that produces a marginalization-equivalent potential over the elimination clique (in the case of dense node removal), or *(ii)* approximate Λ_t as a sparse set of binary potentials that best approximate the true distribution over the elimination clique using a Chow-Liu tree (in the case of sparsified node removal, Fig. 2.1(c)). Before creating new GLC factors, one can optionally reparameterize the variables in each potential so that the constraint will be linearized in a relative frame as opposed to a global frame (Fig. 2.1(d)). Finally, a new GLC factor is created for each potential and we can simply remove the marginalization node from the graph and replace its surrounding factors with the newly computed set (Fig. 2.1(e)).

In the following sections we derive the proposed method and describe each step in detail. This description makes use of many standard concepts from prior work in SLAM including: graphical interpretations of SLAM, the underlying least-squares problem, node removal / marginalization, graph sparsification, the manipulation of information-form multivariate Gaussian distributions, and the representation of robot poses. We recommend [45, 52, 152, 161] as background material to readers who may be less familiar with these concepts.

2.3.1 Building the Target Information

The first step in the algorithm is to correctly identify the target information, Λ_t (Fig. 2.1(b)). Letting $\mathbf{X}_m \subset \mathbf{X}$ be the subset of nodes including the node to be removed and the nodes in its Markov blanket, and letting $\mathbf{Z}_m \subset \mathbf{Z}$ be the subset of measurement factors that only depend on the nodes in $\mathbf{X}_m$, we consider the distribution $p(\mathbf{X}_m|\mathbf{Z}_m) \sim \mathcal{N}^{-1}\big(\boldsymbol{\eta}_m, \Lambda_m\big)$. From Λ_m we can then compute the desired target information, Λ_t, by marginalizing out the elimination node using the standard Schur-complement form. For example, in the graph shown in Fig. 2.1(a), to eliminate node $\mathbf{x}_1$ we would first calculate Λ_m using the standard information-form measurement update equations [50, 161] as

$$\Lambda_m = \mathrm{H}_0^\top \Lambda_0 \mathrm{H}_0 + \mathrm{H}_{01}^\top \Lambda_{01} \mathrm{H}_{01} + \mathrm{H}_{12}^\top \Lambda_{12} \mathrm{H}_{12} + \mathrm{H}_{23}^\top \Lambda_{23} \mathrm{H}_{23} + \mathrm{H}_{13}^\top \Lambda_{13} \mathrm{H}_{13},$$

where H_{ij} are the Jacobians of the observation models for measurements $\mathbf{z}_{ij}$ with information matrices Λ_{ij}, and then compute the target information as

$$\Lambda_t = \Lambda_{\alpha\alpha} - \Lambda_{\alpha\beta}\Lambda_{\beta\beta}^{-1}\Lambda_{\alpha\beta}^\top,$$

where $\Lambda_{\alpha\alpha}$, $\Lambda_{\alpha\beta}$ and $\Lambda_{\beta\beta}$ are the required sub-blocks of Λ_m with $\alpha = [\mathbf{x}_0, \mathbf{x}_2, \mathbf{x}_3]$ and $\beta = [\mathbf{x}_1]$. Note that, though this example only contains unary and binary factors, general n-nary factors are equally acceptable.

While computing Λ_m one could exclude intra-clique factors that are not connected to the marginalization node, for example $\mathbf{z}_0$ and $\mathbf{z}_{23}$ in Fig. 2.1(a), and simply leave them in the graph. In fact, the only strict requirement is that all factors which include the marginalization node be included in Λ_m. However, in §2.3.4 we wish to sparsely approximate the marginalization clique factors, and including all intra-clique factors assures that the resulting connectivity will be sparse. For consistency, we include all intra-clique factors in Λ_m throughout the algorithm and in all experimental results.

The key observation when identifying the target information is that, for a given linearization point, a single n-nary factor can recreate the potential induced by the original pairwise factors by adding the same information (i.e., Λ_m) back to the graph. Moreover, because marginalization only affects the information matrix blocks corresponding to nodes *within* the elimination clique, an n-nary factor that adds the information contained in Λ_t to the graph will induce the same potential in the graph as true node marginalization at the given linearization point.

Note that the target information, Λ_t, *is not* the conditional distribution of the elimination clique given the rest of the nodes, i.e., $p(\mathbf{x}_0, \mathbf{x}_2, \mathbf{x}_3 | \mathbf{x}_4, \mathbf{Z})$, nor is it the marginal distribution of the elimination clique, i.e., $p(\mathbf{x}_0, \mathbf{x}_2, \mathbf{x}_3 | \mathbf{Z})$. Using either of these distributions as the target information results in a wrong estimate as information will be double counted when the n-nary factor is reinserted into the graph.

It is also important to note that the constraints in $\mathbf{Z}_m$ may be purely relative and/or low-rank (e.g., bearing or range-only) and, therefore, may not fully constrain $p(\mathbf{X}_m | \mathbf{Z}_m)$. This can cause Λ_t to be singular. Additionally, some of Λ_t's block-diagonal elements may also be singular. This will require special consideration in subsequent sections.

2.3.2 Generic Linear Constraints

Having defined a method for calculating the target information, Λ_t, we now seek to produce an n-nary factor that captures the same potential. We refer to this new n-nary factor as a generic linear constraint (GLC). Letting $\mathbf{x}_t$ denote a stacked vector of the variables within the elimination clique after node removal, we begin by considering an observation model that directly observes $\mathbf{x}_t$ with a measurement uncertainty that is defined by the target information:

$$\mathbf{z}_t = \mathbf{x}_t + \mathbf{w} \;\text{ where }\; \mathbf{w} \sim \mathcal{N}^{-1}\big(\mathbf{0}, \Lambda_t\big). \tag{2.3}$$

Setting the measurement value, $\mathbf{z}_t$, equal to the current linearization point, $\hat{\mathbf{x}}_t$, induces the desired potential in the graph. Unfortunately, the target information, Λ_t, may not be full rank, which is problematic for optimization methods that rely upon a square root factorization of the measurement information matrix [45, 86]. We can, however, use principle component analysis to transform the measurement to a lower-dimensional representation that is full rank.

We know that Λ_t will be a real, symmetric, positive semi-definite matrix by construction. In general then, it has an eigen-decomposition given by

$$\Lambda_t = \begin{bmatrix} \mathbf{u}_1 & \cdots & \mathbf{u}_q \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \lambda_q \end{bmatrix} \begin{bmatrix} \mathbf{u}_1^\top \\ \vdots \\ \mathbf{u}_q^\top \end{bmatrix} = \mathrm{U}\mathrm{D}\mathrm{U}^\top, \tag{2.4}$$

where U is an orthogonal $p \times q$ matrix, D is a $q \times q$ matrix, p is the dimension of Λ_t, and $q =$

$\mathrm{rank}(\Lambda_t)$. Letting $\mathrm{G} = \mathrm{D}^{\frac{1}{2}}\mathrm{U}^\top$ allows us to write a transformed observation model,

$$\mathbf{z}_{glc} = \mathrm{G}\mathbf{z}_t = \mathrm{G}\hat{\mathbf{x}}_t + \mathbf{w}' \ \text{ where } \ \mathbf{w}' \sim \mathcal{N}^{-1}\big(\mathbf{0}, \Lambda'\big). \tag{2.5}$$

Using the pseudo-inverse [143], $\Lambda_t^+ = \mathrm{U}\mathrm{D}^{-1}\mathrm{U}^\top$, and noting that $\mathrm{U}^\top\mathrm{U} = \mathrm{I}_{q\times q}$, we find that

$$\Lambda' = (\mathrm{G}\Lambda_t^+\mathrm{G}^\top)^{-1} = (\mathrm{D}^{\frac{1}{2}}\mathrm{U}^\top(\mathrm{U}\mathrm{D}^{-1}\mathrm{U}^\top)\mathrm{U}\mathrm{D}^{\frac{1}{2}})^{-1} = \mathrm{I}_{q\times q}.$$

This GLC factor will contribute the desired target information back to the graph, i.e.,

$$\mathrm{G}^\top\Lambda'\mathrm{G} = \mathrm{G}^\top\mathrm{I}_{q\times q}\mathrm{G} = \Lambda_t,$$

but is itself non-singular. This is a key advantage of the proposed GLC method; it automatically determines the appropriate measurement rank such that Λ' is $q \times q$ and invertible, and G is a $q \times p$ new observation model that maps the p-dimensional state to the q-dimensional measurement.

2.3.3 Avoiding World-Frame Linearization in GLC

In the case where the nodes involved are robot poses or landmark locations, GLC, as proposed so far, would linearize the potential with respect to the state variables in the *world-frame*. This may be acceptable in applications where a good world-frame linearization point is known prior to marginalization; however, in general, a more tenable assumption is that a good linearization point exists for the local *relative-frame* transforms between nodes within the elimination clique.

To adapt GLC so that it only locally linearizes the relative transformations between variables in the elimination clique, we first define a "root-shift" function that maps its world-frame coordinates, $\mathbf{x}_t$, to relative-frame coordinates, $\mathbf{x}_r$. Letting $\mathbf{x}_j^i$ denote the j^{th} pose in the i^{th} frame, the root-shift function for $\mathbf{x}_t$ becomes

$$\mathbf{x}_r = \begin{bmatrix} \mathbf{x}_w^1 \\ \mathbf{x}_2^1 \\ \vdots \\ \mathbf{x}_n^1 \end{bmatrix} = r\left(\mathbf{x}_t\right) = r\left(\begin{bmatrix} \mathbf{x}_1^w \\ \mathbf{x}_2^w \\ \vdots \\ \mathbf{x}_n^w \end{bmatrix}\right) = \begin{bmatrix} \ominus\mathbf{x}_1^w \\ \ominus\mathbf{x}_1^w \oplus \mathbf{x}_2^w \\ \vdots \\ \ominus\mathbf{x}_1^w \oplus \mathbf{x}_n^w \end{bmatrix}. \tag{2.6}$$

In this function, the first node is arbitrarily chosen as the root of all relative transforms. The inclusion of the inverse of the root pose, $\mathbf{x}_w^1$, is important as it ensures that the Jacobian of the root-shift operation, R, is invertible, and allows for the representation of target information that is not purely relative.

To derive, instead of starting with a direct observation of the state variables, as in (2.3), we instead start with their root-shifted relative transforms,

$$\mathbf{z}_r = \mathbf{x}_r + \mathbf{w}_r \text{ where } \mathbf{w}_r \sim \mathcal{N}^{-1}\big(\mathbf{0}, \Lambda_r\big). \tag{2.7}$$

Here, the root-shifted target information, Λ_r, is calculated using the fact that the root-shift Jacobian, R, is invertible,

$$\Lambda_r = \mathrm{R}^{-\top}\Lambda_t\mathrm{R}^{-1}. \tag{2.8}$$

Like the original target information, the root-shifted target information, Λ_r, may also be low-rank. Following the same principal component analysis procedure as before, we perform the low-rank eigen-decomposition $\Lambda_r = \mathrm{U}_r\mathrm{D}_r\mathrm{U}_r^\top$, which yields a new observation model,

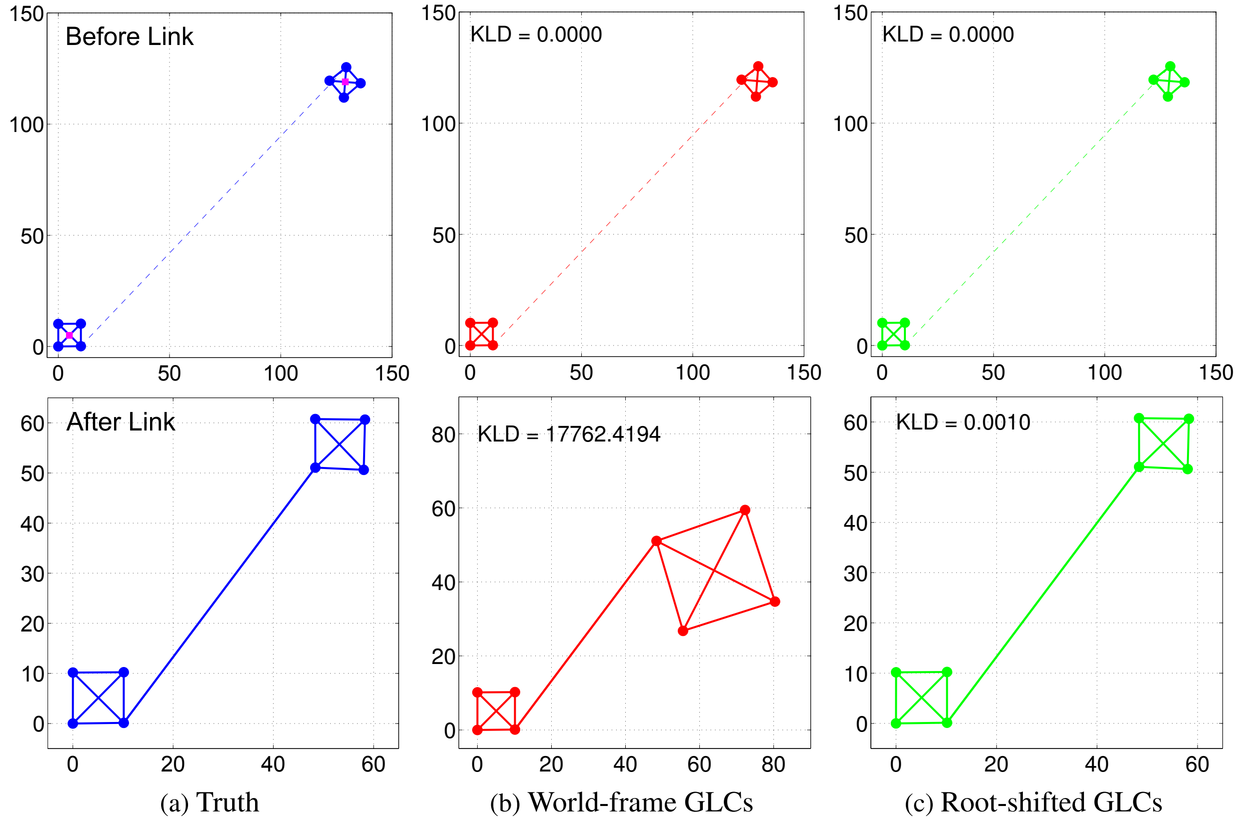
$$\mathbf{z}_{glc_r} = \mathrm{G}_r r(\hat{\mathbf{x}}_t) + \mathbf{w}'_r \text{ where } \mathbf{w}'_r \sim \mathcal{N}^{-1}\big(\mathbf{0}, \Lambda'_r\big), \tag{2.9}$$

where $\mathrm{G}_r = \mathrm{D}_r^{\frac{1}{2}}\mathrm{U}_r^\top$, and measurement information $\Lambda'_r = \mathrm{I}_{q\times q}$. Using the root-shifted linearization point to compute the measurement value, $\mathbf{z}_{glc_r} = \mathrm{G}_r r(\hat{\mathbf{x}}_t)$, will again induce the desired potential in the graph. Now, however, the advantage is that the GLC factor embeds the linearized constraint within a relative coordinate frame defined by the clique, as opposed to an absolute coordinate world-frame. Fig. 2.3 demonstrates this benefit.

It is important to note that this reparameterization step is optional and that it is the only step in GLC that is dependent on the parameterization of the state vector. It is also important to note that reparameterization may not even be necessary if the parameters are already defined in a relative frame as opposed to in the global frame. The root-shift reparameterization defined above is designed for graphs with nodes representing robot poses or landmark locations in the world frame, and is only one example of a possible transformation.

In cases where graph nodes represent other parameters beyond world-frame robot poses or point landmarks, it may be beneficial to reparameterize the variables that support the GLC factor using a different transformation. Any invertible reparameterization of the support variables is acceptable, allowing for large flexibility in designing reparameterizations appropriate for the user's application. In our public implementation [87] we provide a simple callback for user defined reparameterizations. This is exploited in [135], where a reparameterization is defined for use in a more complicated graph with nodes describing a piecewise-planar model of the environment in addition to robot pose nodes.

Figure 2.3 Demonstration of root-shifted versus world-frame GLC factors. Depicted is a simple graph (a) that is initially constructed with two well-connected clusters connected by a highly-uncertain and inaccurate link. The center (magenta) node in each cluster is removed inducing a GLC factor over each cluster. Subsequently, a second measurement is added between the two clusters, correcting the location of the upper-right cluster, and drastically changing its world-frame linearization point. After adding the strong inter-cluster constraint, the graph with the world-frame linearized GLCs fails to converge to the correct optima (b), while the graph with root-shifted GLCs does (c). The KLD from the true marginalization is displayed for each test.



(a) Truth (b) World-frame GLCs (c) Root-shifted GLCs

2.3.4 Sparse Approximate Node Removal

Exact node marginalization causes dense fill-in. As the number of marginalized nodes increases, this dense fill-in can quickly reduce the graph's sparsity and greatly increase the computational complexity of optimizing the graph [45, 86]. In [97], Kretzschmar and Stachniss insightfully propose the use of a Chow-Liu tree (CLT) [35] to approximate the individual elimination cliques as sparse tree structures.

The CLT approximates a joint distribution as the product of pairwise conditional distributions,

$$p(\mathbf{x}_1, \cdots, \mathbf{x}_n) \approx p(\mathbf{x}_1) \prod_{i=2}^{n} p(\mathbf{x}_i | \mathbf{x}_{\mathrm{p}(i)}), \tag{2.10}$$

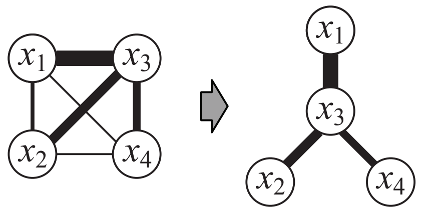
where $\mathbf{x}_1$ is the root variable of the CLT and $\mathbf{x}_{\mathrm{p}(i)}$ is the parent of $\mathbf{x}_i$. The pairwise conditional distributions are selected such that the KLD between the original distribution and the CLT approximation is minimized. To construct it, the maximum spanning tree over all possible pairwise mutual information pairings is found (Fig. 2.4), where the mutual information between two Gaussian random vectors,

$$p(\mathbf{x}_i, \mathbf{x}_j) \sim \mathcal{N}\left(\left[\begin{smallmatrix} \boldsymbol{\mu}_i \\ \boldsymbol{\mu}_j \end{smallmatrix}\right], \left[\begin{smallmatrix} \Sigma_{ii} & \Sigma_{ij} \\ \Sigma_{ji} & \Sigma_{jj} \end{smallmatrix}\right]\right) \equiv \mathcal{N}^{-1}\left(\left[\begin{smallmatrix} \boldsymbol{\eta}_i \\ \boldsymbol{\eta}_j \end{smallmatrix}\right], \left[\begin{smallmatrix} \Lambda_{ii} & \Lambda_{ij} \\ \Lambda_{ji} & \Lambda_{jj} \end{smallmatrix}\right]\right), \tag{2.11}$$

is given by [41]

$$I(\mathbf{x}_i, \mathbf{x}_j) = \frac{1}{2} \log \left(\frac{|\Lambda_{ii}|}{|\Lambda_{ii} - \Lambda_{ij}\Lambda_{jj}^{-1}\Lambda_{ji}|} \right). \tag{2.12}$$

Figure 2.4 Illustration of the Chow-Liu tree approximation. The magnitude of mutual information between variables is indicated by line thickness. The original distribution $p(x_1, x_2, x_3, x_4)$ (left), is approximated as $p(x_1)p(x_3|x_1)p(x_2|x_3)p(x_4|x_3)$ (right).



Like [97], we leverage the CLT approximation to sparsify our n-nary GLC factors; however, our implementation of CLT-based sparsification actually implements the true CLT of the marginalization potential over the elimination clique. In [97], the maximum mutual information spanning tree is computed over the conditional distribution of the elimination clique given the remainder of the graph. This tree is then used as a heuristic to guide which edges should be composed and which edges should be excluded. This presents two issues: First, these composed edges do not actually implement the true CLT. Second, the conditional distribution of the elimination clique is not the distribution that we wish to reproduce by our new factors (see §2.3.1).

We address these issues by computing the CLT distribution (2.10) from the target information, Λ_t, which parameterizes the distribution that we wish to approximate, and then represent the CLT's unary and binary potentials as GLC factors.

2.3.4.1 Chow-Liu Tree Factors

The CLT has two types of potentials, a unary potential on the root node and binary potentials between the rest of the nodes in the tree. We first consider the CLT's binary potentials, $p(\mathbf{x}_i|\mathbf{x}_{\mathrm{p}(i)})$, and in the following use $\mathbf{x}_j = \mathbf{x}_{\mathrm{p}(i)}$ as shorthand for the parent node of $\mathbf{x}_i$. We note that the target-information-derived joint marginal, $p_t(\mathbf{x}_i, \mathbf{x}_j)$, can be computed from Λ_t and written as in (2.11).[1] From this joint marginal, we can then easily write the desired conditional, $p_t(\mathbf{x}_i|\mathbf{x}_j) = \mathcal{N}\big(\boldsymbol{\mu}_{i|j}, \Sigma_{i|j}\big) \equiv \mathcal{N}^{-1}\big(\boldsymbol{\eta}_{i|j}, \Lambda_{i|j}\big)$, and express it as a constraint as

$$\mathbf{e} = \mathbf{x}_i - \boldsymbol{\mu}_{i|j} = \mathbf{x}_i - \Lambda_{ii}^{-1}(\boldsymbol{\eta}_i - \Lambda_{ij}\mathbf{x}_j), \tag{2.13}$$

where $\mathbf{e} \sim \mathcal{N}^{-1}\big(\mathbf{0}, \Lambda_{i|j}\big)$, and with Jacobian,

$$\mathrm{E} = \begin{bmatrix} \frac{\partial \mathbf{e}}{\partial \mathbf{x}_i} & \frac{\partial \mathbf{e}}{\partial \mathbf{x}_j} \end{bmatrix} = \begin{bmatrix} \mathrm{I} & \Lambda_{ii}^{-1}\Lambda_{ij} \end{bmatrix}. \tag{2.14}$$

Therefore, using the standard information-form measurement update, we see that this constraint adds information

$$\mathrm{E}^\top \Lambda_{i|j} \mathrm{E}, \tag{2.15}$$

where $\Lambda_{i|j}$ is simply the sub-block Λ_{ii}.

Treating (2.15) as the input target information, we can produce an equivalent GLC factor for this binary potential using the techniques described in §2.3.2 and §2.3.3. Similarly, the CLT's root unary potential, $p_t(\mathbf{x}_1)$, can also be implemented as a GLC factor by using the target-information-derived marginal information, Λ_{11}, and the same techniques.

[1] In this section, when we refer to marginal and conditional distributions, they are with respect to the target information, Λ_t, *not* with respect to the distribution represented by the full graph.

2.3.5 Implementation Considerations

2.3.5.1 Pseudo-Inverse

As discussed in §2.3.1, the target information, Λ_t, is generally low-rank. This is problematic for the joint marginal (2.11) and conditioning (2.13)–(2.14) calculations used to compute the CLT, as matrix inversions are required. To address this issue, in place of the inverse we use the generalized- or pseudo-inverse [143, §10.5], which can be calculated via an eigen-decomposition for real, symmetric, positive semi-definite matrices. For full-rank matrices the pseudo-inverse produces the same result as the true inverse, while for low-rank matrices it remains well-defined. Calculating the pseudo-inverse numerically requires defining a tolerance below which eigenvalues are considered to be zero. We found that our results are fairly insensitive to this tolerance and that automatically calculating the numerical tolerance using the machine epsilon produced good results. In our experiments we use $\epsilon \times n \times \lambda_{max}$ (the product of the machine epsilon, the size of the matrix, and the maximum eigenvalue) as the numerical tolerance.

2.3.5.2 Pinning

When calculating the pairwise mutual information, the determinants of both the conditional and marginal information matrices in (2.12) must be non-zero, which is again problematic because these matrices are generally low-rank as calculated from the target information, Λ_t. It has been proposed to consider the product of the non-zero eigenvalues as a pseudo-determinant [116, 143] when working with singular, multivariate, Gaussian distributions. Like the pseudo-inverse, this requires determining zero eigenvalues numerically. However, we found that this can cause the mutual information computation to be numerically unstable if the matrices involved have eigenvalues near the threshold. This numerical instability causes the edges to be sorted incorrectly in some cases. This results in a non-optimal structure when the maximum mutual information spanning tree is built and, therefore, a slightly higher KLD from the true marginalization in some graphs.

Instead, we recognize that the CLT's construction requires only the ability to *sort* pairwise links by their relative mutual information (2.12), and not the actual value of their mutual information. A method that slightly modifies the input matrix so that its determinant is non-zero, without significantly affecting the relative ordering of the edges, would also be acceptable. Along these lines we approximate the determinant of a singular matrix using

$$|\Lambda| \approx |\Lambda + \alpha \mathrm{I}|. \tag{2.16}$$

This can be thought of as applying a low-certainty prior on the distribution, and we therefore refer to it as "pinning."[2] Pinning always results in a numerically stable mutual information computation, the only concern is that the relative ordering of the mutual information values remains the same. Experimentally, we found the quality of the results to be less sensitive to the pinning α value than the numerical epsilon in the pseudo-determinant. We, therefore, elected to use pinning with $\alpha = 1$ in our experiments when evaluating the determinants in the pairwise mutual information (2.12).

2.3.6 Computational Complexity

The core operations that GLC relies on, in and of themselves, are computationally expensive. The CLT approximation has a complexity of $\mathcal{O}(m^2 \log m)$, where m is the number of nodes. Matrix operations on the information matrix with n variables, including the eigen-decomposition, matrix multiplication, and inversion operations, have a complexity of $\mathcal{O}(n^3)$. Fortunately, the input size for these operations is limited to the number of nodes within the elimination clique, which in a SLAM graph is controlled by the perceptual radius. In general, the number of nodes and variables in an elimination clique is much less than the total number of nodes in the full graph. We will see in Chapter IV that GLC's calculations are computationally feasible in both offline and real-time online settings.

2.4 Experimental Evaluation of GLC Node Removal

Table 2.1 Experimental Datasets

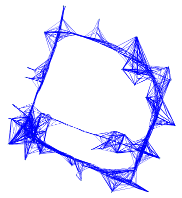
Dataset	Node Types	Factor Types	# Nodes	# Factors
Intel Lab	3-DOF pose	3-DOF odometry, 3-DOF laser scan-matching	910	4,454
Killian Court	3-DOF pose	3-DOF odometry, 3-DOF laser scan-matching	1,941	2,191
Duderstadt Center	6-DOF pose	6-DOF odometry, 6-DOF laser scan-matching	552	1,774
EECS Building	6-DOF pose	6-DOF odometry, 6-DOF laser scan-matching	611	2,134
Victoria Park	3-DOF pose, 2-DOF Landmark	3-DOF odometry, 2-DOF landmark observation	7,120	10,609
USS Saratoga	6-DOF pose	6-DOF odometry, 5-DOF monocular-vision, 1-DOF depth	1,513	5,433

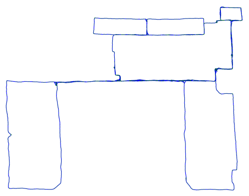
First, we directly evaluate GLC node removal over a variety of SLAM graphs (summarized in Fig. 2.5 and Table 2.1), including:

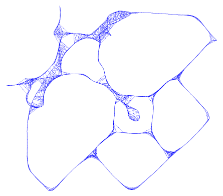
- Two standard 3-DOF pose-graphs, *Intel Lab* and *Killian Court*.

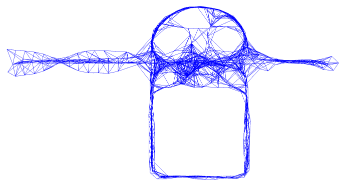
[2]This is related to the derivation of the pseudo-determinant in [116], which uses a similar form in the limit as $\alpha \to 0$.

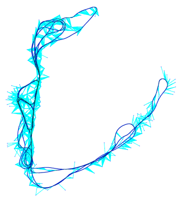
Figure 2.5 Graphs used in GLC's evaluation. Blue links represent full-state (3-DOF or 6-DOF) relative-pose constraints from odometry and laser scan-matching. Red links represent 5-DOF relative-pose constraints modulo-scale from monocular vision. Cyan links represent landmark observation factors.

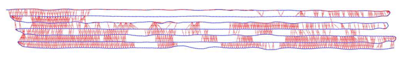

(a) Intel Lab


(b) Killian Court


(c) Duderstadt Center


(d) EECS Building


(e) Victoria Park


(f) USS Saratoga

- Two 6-DOF pose-graphs built using data from a Segway ground robot (Fig. A.1(a)) equipped with a Velodyne HDL-32E laser scanner as the primary sensing modality, *Duderstadt Center* and *EECS Building*.

- The *Victoria Park* 3-DOF graph with poses and landmarks.

- A 6-DOF graph produced by a hovering autonomous underwater vehicle (HAUV) performing monocular SLAM for autonomous ship hull inspection [71], *USS Saratoga*.

The proposed algorithm was implemented using iSAM [85, 86] as the underlying optimization engine. The code is open-source and available for download within the iSAM repository [87]. For comparison, a dense measurement composition (MC) method as described in §2.2, and a sparse MC method based upon CLT-guided node removal, as proposed in [97], were also implemented.

For each graph, the original full graph is first optimized using iSAM. Then the different node removal algorithms are each performed to remove a varying percentage of nodes evenly spaced throughout the trajectory. Finally, the graphs are again optimized in iSAM.

For each experiment the true marginal distribution is recovered by obtaining the linearized information matrix from the full graph about the optimization point and performing Schur-complement marginalization. This provides a ground-truth distribution that we can directly compare the distribution after node removal against.

A summary of our results are provided in Table 2.2, which shows the KLD (normalized by the

DOF of the distribution after node removal) from the true marginalization as an increasing percentage of nodes are removed from the graph. Results for dense-exact and sparse-approximate GLC are provided for all six graphs, while results for dense and sparse-approximate MC are provided only for the pose-graphs with full-state constraints. The Saratoga graph is excluded as it contains 5-DOF monocular relative-pose constraints for which MC is undefined.

Table 2.2 Experimental Normalized KLD from True Marginalization

	Dense GLC		**CLT Sparse GLC**						
% Nodes Removed	25.0 %	33.3 %	25.0 %	33.3 %	50.0 %	66.6 %	75.0 %	83.3 %	87.5 %
Intel Lab	0.002	0.002	0.096	0.110	0.128	0.126	0.131	0.170	0.139
Killian Court	0.001	0.002	0.006	0.008	0.013	0.020	0.023	0.028	0.033
Duderstadt Center	0.001	0.000	0.003	0.003	0.003	0.005	0.008	0.018	0.024
EECS Building	0.003	0.002	0.005	0.005	0.004	0.010	0.017	0.035	0.049
Victoria Park	0.001	0.002	0.005	0.007	0.011	0.017	0.024	0.042	0.057
USS Saratoga	0.016	0.013	0.017	0.015	0.001	0.002	0.001	0.001	0.003

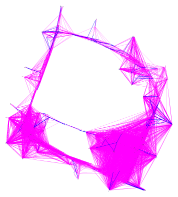
	Dense Pairwise MC		**Sparse Pairwise MC**						
% Nodes Removed	25.0 %	33.3 %	25.0 %	33.3 %	50.0 %	66.6 %	75.0 %	83.3 %	87.5 %
Intel Lab	1.57E3	7.19E5	0.023	0.038	0.108	0.280	0.428	0.800	1.295
Killian Court	0.01	0.02	0.005	0.007	0.013	0.023	0.031	0.042	0.048
Duderstadt Center	0.18	42.69	0.002	0.008	0.008	0.025	0.044	0.070	0.100
EECS Building	160.76	9.32E4	0.003	0.005	0.010	0.027	0.043	0.113	0.170

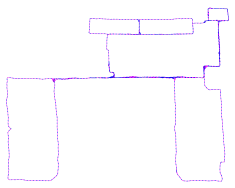
2.4.1 Dense GLC Node Removal

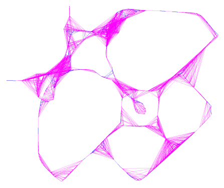
We first consider the results for our method when performing exact node removal with dense fill-in. Visual examples of the resulting dense GLC graphs are shown in Fig. 2.6.

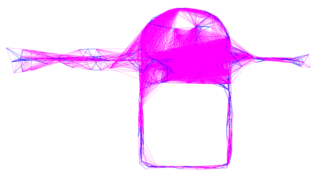
To put dense GLC's KLD values from Table 2.2 into perspective, we look at the case with the highest KLD, which is the Saratoga graph with 25% of nodes removed (i.e., KLD $= 0.016$). Under these conditions, the reconstructed graph has a mean error in translation and rotation of $18.9\ \mathrm{mm}$ and $3.8\ \mathrm{mrad}$, respectively, when compared to the original baseline pose-graph SLAM result. To more systematically investigate the accuracy of GLC's marginal pose uncertainties in Fig. 2.7 we consider the eigenvalues of the difference between the marginal covariances of the GLC-derived and the true distribution, $\mathrm{eig}(\Sigma_{ii}^{\mathrm{GLC}} - \Sigma_{ii}^{\mathrm{TRUE}})$. In the ideal case the eigenvalues of this difference will be zero, indicating perfect agreement between GLC and the true marginalization. Values larger than zero indicate conservative estimates while those less than zero indicate overconfidence. Conservative estimates are generally preferred to overconfident estimates in robotics, as overconfidence can lead to data association failure [127] and unsafe path planning and obstacle avoidance. For dense GLC we see that the eigenvalues are almost zero (note the 10^{-6} scale),

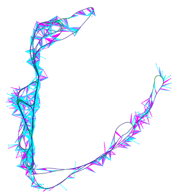
Figure 2.6 Example graphs after dense GLC node removal. New GLC factors are shown in magenta. Note that this is the MRF representation of the graph connectivity. The percentage indicates the percentage of nodes that have been removed.


(a) Intel (33.3%)


(b) Killian (33.3%)


(c) Duderstadt (25.0%)


(d) EECS (25.0%)


(e) Victoria (33.0%)

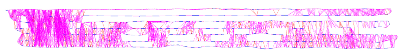

(f) USS Saratoga (25.0%)

Figure 2.7 Accuracy of GLC-derived marginals for the USS Saratoga dataset with 25% of nodes removed. The range of the eigenvalues of the difference between the covariances of the GLC-derived marginals and true marginals, $\text{eig}(\Sigma_{ii}^{\text{GLC}} - \Sigma_{ii}^{\text{TRUE}})$, is shown for both dense and sparse GLC. Note 10^{-6} scale.

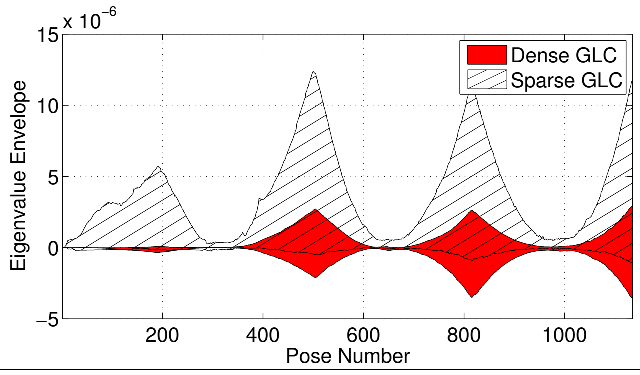
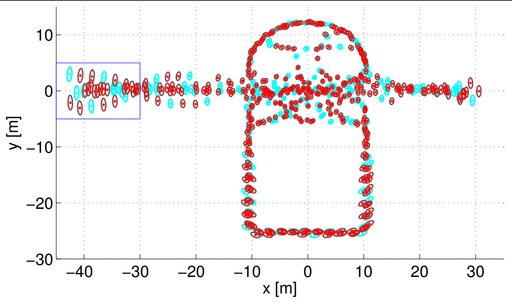
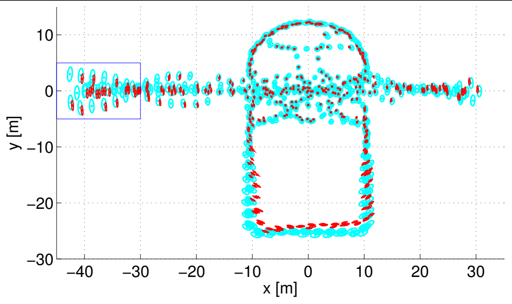


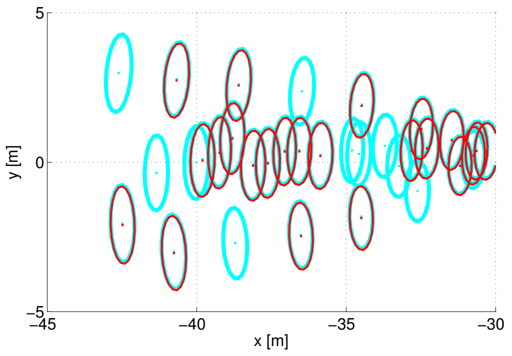
Figure 2.8 Sample 3-σ uncertainty ellipses for the EECS graph and the Intel graphs with 33.3% node removal using dense GLC and dense MC. The true marginalization uncertainties are shown in cyan. Note that fewer red ellipses are plotted than cyan because fewer nodes remain in the graph after node removal.



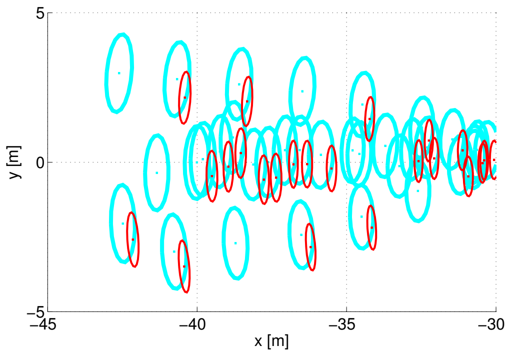
(a) EECS Dense GLC



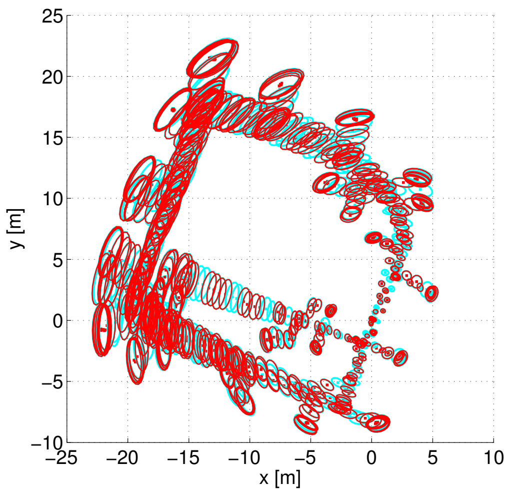
(b) EECS Dense MC



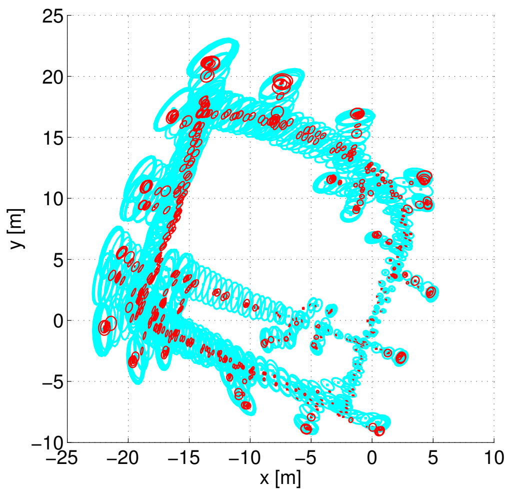
(c) EECS Dense GLC (zoom)



(d) EECS Dense MC (zoom)



(e) Intel Dense GLC



(f) Intel Dense MC

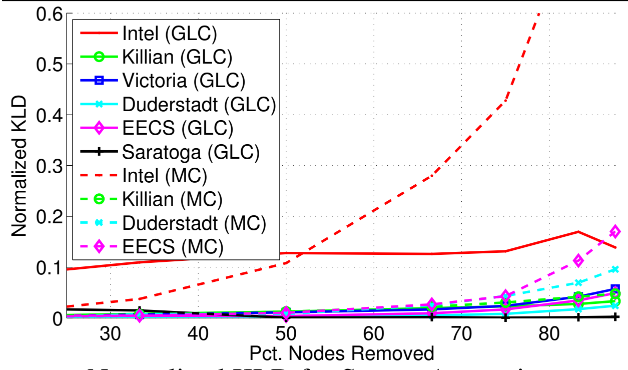
indicating excellent agreement between GLC and the true marginalization. Additionally, visual examples of the marginal covariances for the EECS and Intel graphs are shown in Fig. 2.8(a) and Fig. 2.8(e), respectively.

As more nodes are removed from the graph, dense GLC node removal quickly becomes computationally expensive due to an increase in the size of the elimination cliques. Removing nodes may take on the order of tens of seconds [25] per node. This, combined with increased optimization cost due to the dense connectivity, limits the applicability of dense node removal to applications where only a small percentage (in our experiments around one-third to one-half) of nodes are removed.

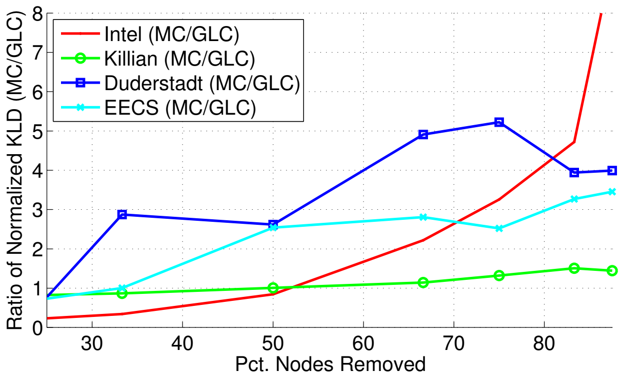
Considering the results for dense MC, Table 2.2 shows that it performs quite poorly—as more nodes are removed, the KLD quickly increases. This is because dense pairwise MC fails to properly track the correlation that develops between composed measurements (as demonstrated in §2.2); thus, the higher the connectivity in the graph, the more measurement information gets double counted when compounding. This results in overconfidence, as well as a shift in the optimal mean (Fig. 2.8(b) and Fig. 2.8(f)).

2.4.2 CLT Sparse-Approximate GLC Node Removal

Figure 2.9 KLD comparison for sparse approximate node removal. The KLD (normalized by the DOF of the distribution after node removal) for the GLC and MC-based sparse approximate node removal methods is shown in (a). In (b) the ratio of KLD between MC and GLC is plotted, highlighting that, in most cases, as more nodes are removed MC induces several times higher KLD than GLC.



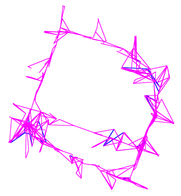
(a) Normalized KLD for Sparse-Approximate Methods

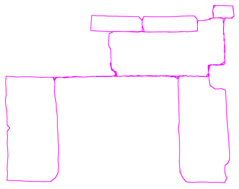


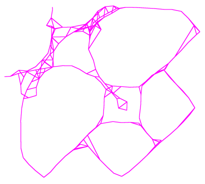
(b) Ratio of Normalized KLD (MC / GLC)

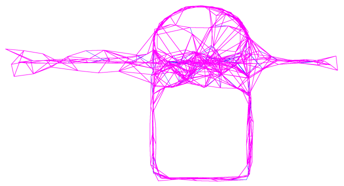
Next, we consider the results for sparse-approximate GLC node removal. Table 2.2 shows that in many graphs, including Killian, Duderstadt, EECS, and USS Saratoga, the KLD for sparse-approximate GLC is only slightly worse than that of dense-exact GLC—indicating that very little graph information is lost due to the CLT approximation. Fig. 2.9 illustrates the KLD for the sparse-

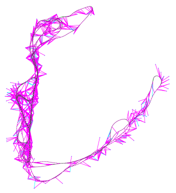
Figure 2.10 Example graphs after CLT sparse GLC node removal. New GLC factors are shown in magenta. The percentage indicates the percentage of nodes that have been removed.


(a) Intel (66.3%)


(b) Killian (83.3%)


(c) Duderstadt (66.6%)


(d) EECS (50.0%)


(e) Victoria (75.0%)

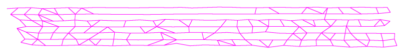
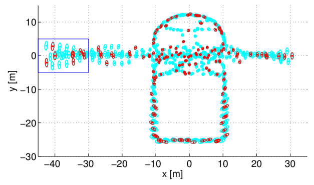
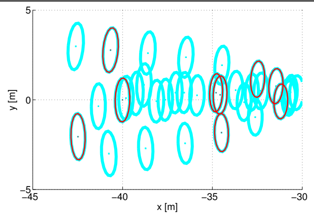
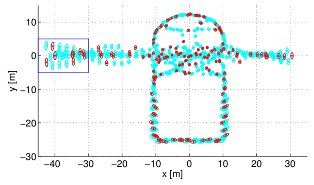
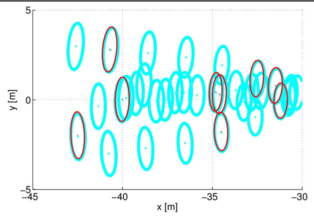

(f) USS Saratoga (87.5%)

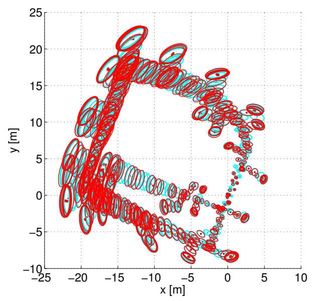
Figure 2.11 Sample 3-σ uncertainty ellipses for the EECS graph with 75% node removal and for the Intel graph with 33.3% and 87.5% node removal using sparse GLC and MC. The true marginalization uncertainties are shown in cyan. Note that fewer red ellipses are plotted than cyan because fewer nodes remain in the graph after node removal. The percentage indicates the percentage of nodes that have been removed.

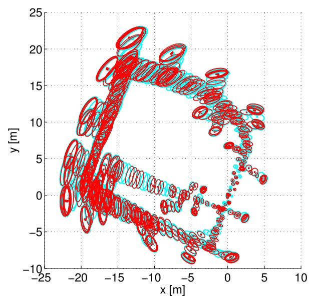

(a) EECS Sparse GLC (75.0%)

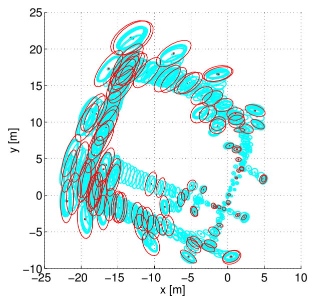

(b) EECS Sparse GLC (75.0%, zoom)

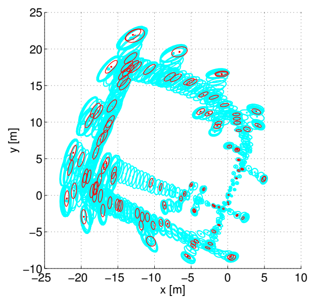

(c) EECS Sparse MC (75.0%)


(d) EECS Sparse MC (75.0%, zoom)


(e) Intel Sparse GLC (33.3%)


(f) Intel Sparse MC (33.3%)


(g) Intel Sparse GLC (87.5%)


(h) Intel Sparse MC (87.5%)

approximate versions of GLC and MC, again normalizing the KLD by the number of degree of freedom in the graph after node removal. Visual examples for sparsification on the graphs are shown in Fig. 2.10. Examples of the marginal covariances for the EECS and Intel graphs are shown in Fig. 2.11.

Considering the results for sparse MC, Table 2.2 shows that, unlike dense MC, sparse MC performs reasonably well when removing a smaller percentage of nodes. This is because information double counting during measurement composition accumulates to a lesser extent than in the dense case because of sparsification. However, as the percentage of removed nodes increases, we see that sparse MC produces less accurate and more inconsistent results than sparse GLC. This is illustrated in Fig. 2.9(b), which highlights the ratio in the normalized KLD between MC and GLC, and in Fig. 2.11 and Fig. 2.12, which compare the marginal covariances of the distributions.

It is important to note that the proposed method is not guaranteed to be conservative. This is due to the fact that the CLT approximation simply seeks to produce the minimum KLD and does not guarantee a conservative approximation. This is addressed in §2.5 where we propose several methods that provide a guaranteed-conservative approximation, while still producing a low KLD.

In the case of the Intel graph, MC achieves a significantly better KLD than GLC when removing a small percentage of nodes. This is due to the fact that when removing a small number of nodes, GLC is slightly conservative (Fig. 2.11(e)), while MC's inconsistency coincidentally yields a less conservative estimate with a better KLD (Fig. 2.11(f)). As more nodes are removed this trend is continued, with GLC remaining conservative and producing a better KLD (Fig. 2.11(g)), while MC becomes very inconsistent (Fig. 2.11(h)).

Unlike dense node removal, sparse GLC maintains graph sparsity and keeps elimination clique size small. This results in fast node removal on the order of tens of milliseconds per node [25, 26].

2.5 Guaranteed Conservative GLC Node Removal

The sparse approximate version of GLC as described thus far accurately implements the CLT approximation without double counting measurement information. The CLT produces an approximation with the lowest KLD among all tree structures. Unfortunately, achieving minimum KLD often requires that the CLT approximation be slightly overconfident with respect to the true distribution, as illustrated in Fig. 2.13(c). In most SLAM applications, conservative estimates are strongly preferred to overconfident estimates. During map building, overconfidence can adversely affect data association, causing the system to miss valid loop-closures. Additionally, when using

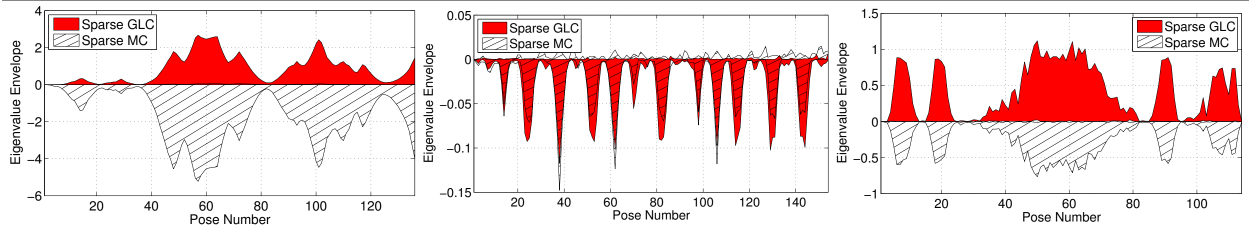


Figure 2.12 Comparison of marginal distributions between sparse GLC and sparse MC, for the Duderstadt, EECS, and Intel graphs. The range of eigenvalues of the difference between the covariances of the approximate-node-removal marginals and true marginals, $\mathrm{eig}(\Sigma_{ii}^{\mathrm{EST}} - \Sigma_{ii}^{\mathrm{TRUE}})$, is shown for both sparse GLC and sparse MC. In the ideal case the range will be zero, indicating perfect agreement between the approximate and the true marginals. Values larger than zero indicate conservative estimates while those less than zero indicate over-confidence. The results show that sparse GLC remains conservative while sparse MC is overconfident for the Duderstadt and Intel graphs. In the case of the EECS graph both methods produce estimates that are occasionally overconfident.

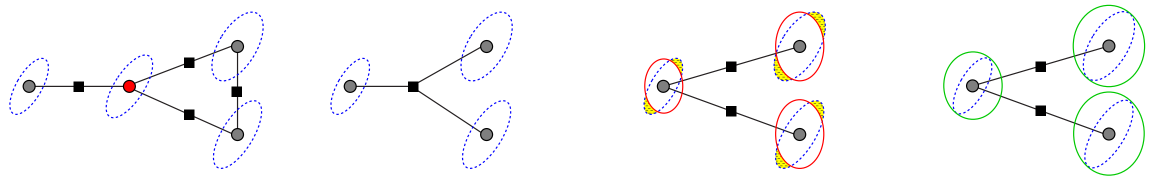
(a) Duderstadt (75.0% Removed) (b) EECS (75.0% Removed) (c) Intel (87.5% Removed)

the resulting map, overconfident estimates can lead to unsafe path planning and obstacle avoidance [162]. Here, as in Vial et al. [173] and Huang et al. [73], we define a conservative estimate as one where the covariance of the sparsified distribution is greater than or equal to that of the true distribution, i.e., $\tilde{\Sigma} \geq \Sigma$.

In this section, we explore optimization-based methods for conservative sparsification of the dense cliques induced by node marginalization. The proposed methods are integrated within the GLC framework and are designed to maintain the advantages of sparse-approximate GLC. The proposed methods start with a sparse, but (potentially) overconfident, Chow-Liu tree approximation of the marginalization potential, and then use optimization-based methods to adjust the approximation so that it is conservative, subject to minimizing the KLD from the true marginalization potential. Our proposed methods address some of the shortcomings of the methods proposed in Vial et al. [173] and Huang et al. [73] as described in §2.1.

- Like [173] and [73] our proposed methods ensure that the sparse approximation remains conservative while providing a low KLD from the true distribution.
- Our methods produce a new set of factors using only the current factors as input, and do not require the full linearized information matrix as input as in [173] and [73].
- The computational complexity of our methods are dependent only upon the size of the elimination clique, and not on the size of the graph beyond the clique. We do not require a large matrix inversion to formulate the subproblem as in [173], nor do we operate over the entire graph as in [73].

Figure 2.13 Guaranteed conservative GLC overview. Starting with the original factor graph (a), the red node is marginalized. This induces a densely connected factor over the marginalization clique (b). The true uncertainty ellipses (dashed blue lines) are not affected by marginalization. The dense marginalization potential is then approximated using a sparse Chow-Liu tree (c). The uncertainty ellipses after the Chow-Liu tree approximation (red lines) are overconfident (note the yellow regions that are no longer probabilistically plausible). The sparse Chow-Liu tree approximation is adjusted so that it is conservative (d), modifying the uncertainty ellipses (green lines).


(a) Original Graph (b) Node Marginalization (c) Chow-Liu Tree Approx. (d) Conservative Approx.

2.5.1 Optimization Formulation

As in Vial et al. [173] and Huang et al. [73], we wish to minimize the KLD of the sparse approximation while producing a consistent estimate. When the distribution means are equal (i.e., $\boldsymbol{\eta}_t = \Lambda_t \boldsymbol{\mu}_t$ and $\tilde{\boldsymbol{\eta}}_t = \tilde{\Lambda}_t \boldsymbol{\mu}_t$), the KLD between the marginalization-induced factor characterized by Λ_t (Fig. 2.1(b)) and its approximation characterized by $\tilde{\Lambda}_t$ is given by

$$\mathcal{D}_{KL}\left(\mathcal{N}^{-1}(\boldsymbol{\eta}_t, \Lambda_t) \| \mathcal{N}^{-1}(\tilde{\boldsymbol{\eta}}_t, \tilde{\Lambda}_t)\right) = \frac{1}{2}\left(\operatorname{tr}(\tilde{\Lambda}_t \Lambda_t^{-1}) + \ln\frac{|\Lambda_t|}{|\tilde{\Lambda}_t|} - \dim(\boldsymbol{\eta}_t)\right). \quad (2.17)$$

Noting that Λ_t and the state dimension are constant, the KLD optimization objective with respect to $\tilde{\Lambda}_t$ can be written as

$$f_{KL}(\tilde{\Lambda}_t) = \operatorname{tr}(\tilde{\Lambda}_t \Lambda_t^{-1}) - \ln|\tilde{\Lambda}_t|. \quad (2.18)$$

However, as discussed in §2.3.1, Λ_t will, in general, be low-rank, making the KLD ill-defined. In Vial et al. [173], a full-rank subproblem is defined, but its implementation requires inverting the information matrix associated with the rest of the graph beyond the subproblem's Markov blanket. In Huang et al. [73], optimization is performed over the full information matrix, which will always be full-rank for well-posed SLAM graphs.

We know that Λ_t will be a real, symmetric, positive semi-definite matrix due to the nature of its construction. In general then, it has an eigen-decomposition given by

$$\Lambda_t = \begin{bmatrix} \mathbf{u}_1 & \cdots & \mathbf{u}_q \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \lambda_q \end{bmatrix} \begin{bmatrix} \mathbf{u}_1^\top \\ \vdots \\ \mathbf{u}_q^\top \end{bmatrix} = \mathrm{UDU}^\top, \quad (2.19)$$

where U is a $p \times q$ orthogonal matrix, D is a $q \times q$ matrix, p is the dimension of Λ_t, and $q = \operatorname{rank}(\Lambda_t)$. Noting that the KLD is invariant under parameter transformations, we rewrite the KLD objective as

$$f_{KL}(\tilde{\Lambda}_t) = \operatorname{tr}(\mathrm{U}^\top \tilde{\Lambda}_t \mathrm{U} \mathrm{D}^{-1}) - \ln |\mathrm{U}^\top \tilde{\Lambda}_t \mathrm{U}|. \tag{2.20}$$

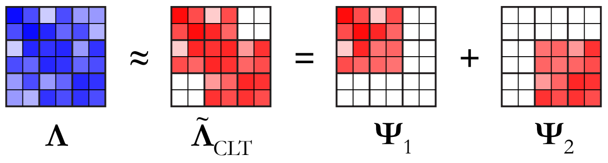
When Λ_t is full-rank

$$f_{KL}(\tilde{\Lambda}_t) = \operatorname{tr}(\tilde{\Lambda}_t \mathrm{U} \mathrm{D}^{-1} \mathrm{U}^\top) - \ln |\mathrm{U}^\top||\tilde{\Lambda}_t||\mathrm{U}| = \operatorname{tr}(\tilde{\Lambda}_t \Lambda_t^{-1}) - \ln |\tilde{\Lambda}_t|,$$

which is exactly equivalent to (2.18). When Λ_t is low-rank (2.20) computes the KLD over the subspace where Λ_t is well-defined. This allows us to work with the low-rank target information and limit the extent of the optimization problem to the elimination clique. Intuitively, this parameter transformation can be thought of as using the pseudo-inverse [143] of Λ_t to compute the KLD. However, it is important that the transformation be applied to $\tilde{\Lambda}_t$ so that, during optimization, $\ln |\mathrm{U}^\top \tilde{\Lambda}_t \mathrm{U}|$ is evaluated instead of $\ln |\tilde{\Lambda}_t|$, which will be undefined because the optimal $\tilde{\Lambda}_t$ will also be low-rank.

2.5.1.1 Chow-Liu Tree Approximation

Figure 2.14 Illustration of the CLT approximation's information matrix for the sample graph in Fig. 2.13(c).



The original version of sparse-approximate GLC approximated the marginalization-induced factor using a Chow-Liu tree. The CLT produces the minimum KLD among all trees by computing the pairwise mutual information between all nodes, and building the maximum mutual information spanning tree. The CLT can be expressed as

$$\mathcal{N}^{-1}(\boldsymbol{\eta}_t, \Lambda_t) \approx \mathcal{N}^{-1}(\tilde{\boldsymbol{\eta}}_t, \tilde{\Lambda}_{\mathrm{CLT}}) = \prod_i p(\mathbf{x}_i | \mathbf{x}_{\mathrm{p}(i)}), \tag{2.21}$$

where $\mathbf{x}_{\mathrm{p}(i)}$ is the parent of $\mathbf{x}_i$, and for the root of the CLT $p(\mathbf{x}_0|\mathbf{x}_{\mathrm{p}(0)}) = p(\mathbf{x}_0)$. The information added to the graph by the CLT approximation can then be written as

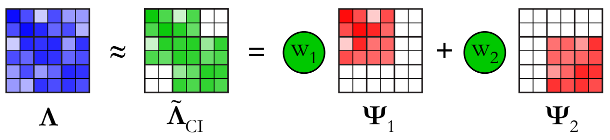
$$\tilde{\Lambda}_{\text{CLT}} = \sum_i \Psi_i, \tag{2.22}$$

where each Ψ_i is the information associated with one of the unary or binary factors in the tree, padded with zeros so that the appropriate dimensions are achieved (Fig. 2.14).

The methods proposed in this chapter all start with the CLT approximation and then use optimization methods to "adjust" the approximation to ensure that it is conservative. Intuitively, this can be thought of as numerically growing the uncertainty of the CLT so that it is conservative, while minimizing the additional KLD from the true distribution. Each method will produce, by construction, an approximation with the same sparsity pattern as the CLT.

2.5.1.2 Covariance Intersection

Figure 2.15 Illustration of Covariance Intersection and Weighted Factors approximate information. The structure of approximate information matrix is shown for the sample graph in Fig. 2.13(d). Note that, even though Covariance Intersection and Weighted Factors have the same structure for the approximate information, they differ in how they ensure that the approximation is conservative.



The CLT approximation is often overconfident in practice. This is due to the fact that the tree structure is not, in general, capable of capturing the full correlation structure of the original distribution. The covariance intersection algorithm, proposed by Julier and Uhlmann [83], can be used to consistently merge measurements with unknown correlation and can be used to weight the CLT factors so that their sum is conservative. Clearly, we should be able to do better than covariance intersection because the true correlation in the original distribution, Λ_t, is known. Covariance intersection, however, does provide an easy-to-compute lower bound on the approximation performance to which we can compare additional methods. Additionally, it provides a strictly-feasible starting point for more complex optimization problems. The approximate target information produced by covariance intersection is defined as a convex combination of the CLT factors,

$$\tilde{\Lambda}_{\text{CI}}(\mathbf{w}) = \sum_i w_i \Psi_i, \tag{2.23}$$

where each w_i scales the information added by each factor (Fig. 2.15). The optimal weights can then be found by solving the convex semidefinite program,

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & f_{KL}(\tilde{\Lambda}_{\text{CI}}(\mathbf{w})) \\ \text{subject to} \quad & \sum_i w_i = 1. \end{aligned} \tag{2.24}$$

Note that it is the fact that the weights must sum to one that ensures the resulting approximation is conservative.

2.5.1.3 Weighted Factors

Because the true distribution is known, we can relax covariance intersection's requirement that weights sum to one. Instead we constrain the weights to be between zero and one, and add the conservative constraint proposed in [173] and [73]. We refer to this formulation as "weighted factors," and its approximate target information is defined as

$$\tilde{\Lambda}_{\text{WF}}(\mathbf{w}) = \sum_i w_i \Psi_i, \tag{2.25}$$

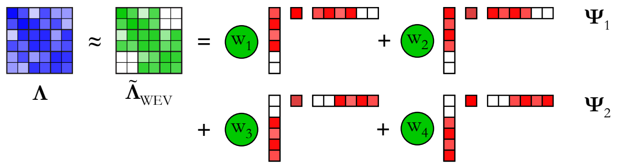
which has the same structure as Covariance Intersection (Fig. 2.15). The optimal weights can then be found by solving

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & f_{KL}(\tilde{\Lambda}_{\text{WF}}(\mathbf{w})) \\ \text{subject to} \quad & 0 \le w_i \le 1, \ \forall i \\ & \Lambda_t \ge \tilde{\Lambda}_{\text{WF}}(\mathbf{w}), \end{aligned} \tag{2.26}$$

which is again a convex semidefinite program.

2.5.1.4 Weighted Eigenvalues

Figure 2.16 Illustration of Weighted Eigenvalues approximate information. The structure of the approximation's information matrix is shown for the sample graph in Fig. 2.13(d).



Instead of weighting each factor by a single value, finer-grained control can be achieved by weighting each factor along its principal axes independently. We refer to this formulation as "weighted eigenvalues." Each factor has an eigen-decomposition given by

$$\Psi_i = \begin{bmatrix} \mathbf{u}_1^i & \cdots & \mathbf{u}_{q_i}^i \end{bmatrix} \begin{bmatrix} \lambda_1^i & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \lambda_{q_i}^i \end{bmatrix} \begin{bmatrix} {\mathbf{u}_1^i}^\top \\ \vdots \\ {\mathbf{u}_{q_i}^i}^\top \end{bmatrix}. \tag{2.27}$$

Using the eigen-decomposition of each factor we can write the approximate target information as

$$\begin{aligned} \tilde{\Lambda}_{\mathrm{WEV}}(\mathbf{w}) &= \sum_i \sum_{j=1}^{q_i} w_j^i \lambda_j^i \mathbf{u}_j^i {\mathbf{u}_j^i}^\top \\ &= \sum_k w_k \lambda_k \mathbf{u}_k \mathbf{u}_k^\top, \end{aligned} \tag{2.28}$$

as illustrated in Fig. 2.16. The optimal weights are found by solving

$$\begin{aligned} \underset{\mathbf{w}}{\text{minimize}} \quad & f_{KL}(\tilde{\Lambda}_{\mathrm{WEV}}(\mathbf{w})) \\ \text{subject to} \quad & 0 \leq w_k \leq 1, \ \forall k \\ & \Lambda_t \geq \tilde{\Lambda}_{\mathrm{WEV}}(\mathbf{w}). \end{aligned} \tag{2.29}$$

Note that the number of optimization variables has increased in comparison to the covariance intersection and weighted factors formulations. As will be demonstrated in §2.6, this results in a significant increase in the computational cost.

2.5.1.5 Implementation Considerations

Each of the proposed semidefinite programs are convex and can be efficiently solved using interior point methods [18, 171, 172]. Interior point methods require that a strictly-feasible starting point be found before optimization, i.e., an initial approximation, where $0 < w_i < 1, \ \forall i$ and $\Lambda_t > \tilde{\Lambda}_t(\mathbf{w})$. Covariance intersection with uniform weights provides an easy-to-compute strictly-feasible starting point, and is used in all experiments.

For low-rank target information, the conservative constraint, $\Lambda_t - \tilde{\Lambda}_t(\mathbf{w})$, is semidefinite and will have at least one zero eigenvalue, and therefore, no strictly-feasible starting point exists. Additionally, it prevents the evaluation of the optimization problem's gradient and Hessian. Instead, the conservative constraint is implemented as

$$\Lambda_t + \epsilon \mathrm{I} \geq \tilde{\Lambda}_t(\mathbf{w}),$$

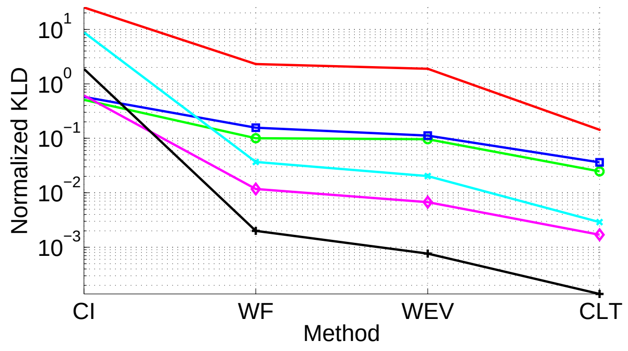
so that a strictly-feasible starting point exists and the gradient and Hessian can be evaluated. Our experimental evaluation indicates that the actual value of ϵ has very little effect on the results. All experiments are performed with $\epsilon = 0.1$, though values $10^{-5} \leq \epsilon \leq 1$ produced nearly equivalent results.

2.6 Experimental Evaluation of Guaranteed Conservative GLC

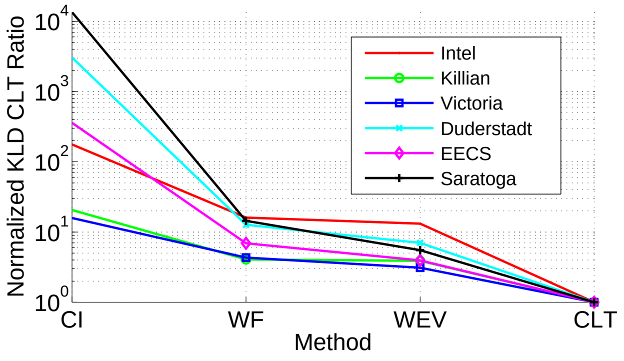
Table 2.3 Experimental Normalized KLD for Conservative GLC

		Covariance Intersection		Weighted Factors		Weighted Eigenvalues		Chow-Liu Tree
Dataset	% Removed	NKLD	CLT Ratio	NKLD	CLT Ratio	NKLD	CLT Ratio	NKLD
Intel Lab	33.3%	25.237	175.85×	2.302	16.04×	1.890	13.16×	0.144
Killian Court	66.7%	0.508	20.58×	0.101	4.07×	0.096	3.90×	0.025
Victoria Park	75.0%	0.574	15.85×	0.157	4.32×	0.112	3.10×	0.036
Duderstadt Center	50.0%	8.788	3,038.06×	0.037	12.70×	0.020	7.00×	0.003
EECS Building	25.0%	0.608	357.15×	0.012	6.90×	0.007	3.95×	0.002
USS Saratoga	33.3%	1.865	13,441.43×	0.002	14.44×	0.001	5.51×	0.000

Figure 2.17 KLD comparison for conservative sparse approximate node removal. The KLD (normalized by the DOF of the distribution after node removal) for the Covariance Intersection, Weighted Factors, Weighted Eigenvalues, and CLT sparse approximate node removal methods is shown in (a). In (b) the ratio of KLD between each method and CLT is plotted.



(a) Normalized KLD



(b) Ratio of Normalized KLD w.r.t. CLT

To evaluate the proposed methods, we test their performance on a variety of SLAM graphs, summarized in Fig. 2.5 and Table 2.1. As before, the proposed algorithms were implemented using iSAM [85–87] as the underlying optimization engine. For each graph, the original full graph is first optimized using iSAM. Then the different node removal algorithms are each used to remove a set of nodes evenly spaced throughout the trajectory. Finally, the graphs are again optimized in iSAM.

For each experiment the true marginal distribution is recovered by obtaining the linearized information matrix from the full graph about the optimization point and performing Schur-complement marginalization. This provides a ground-truth distribution that we can directly compare our conservative distribution against. In order to provide a benchmark, the CLT approximation as proposed in §2.3.4 is also evaluated.

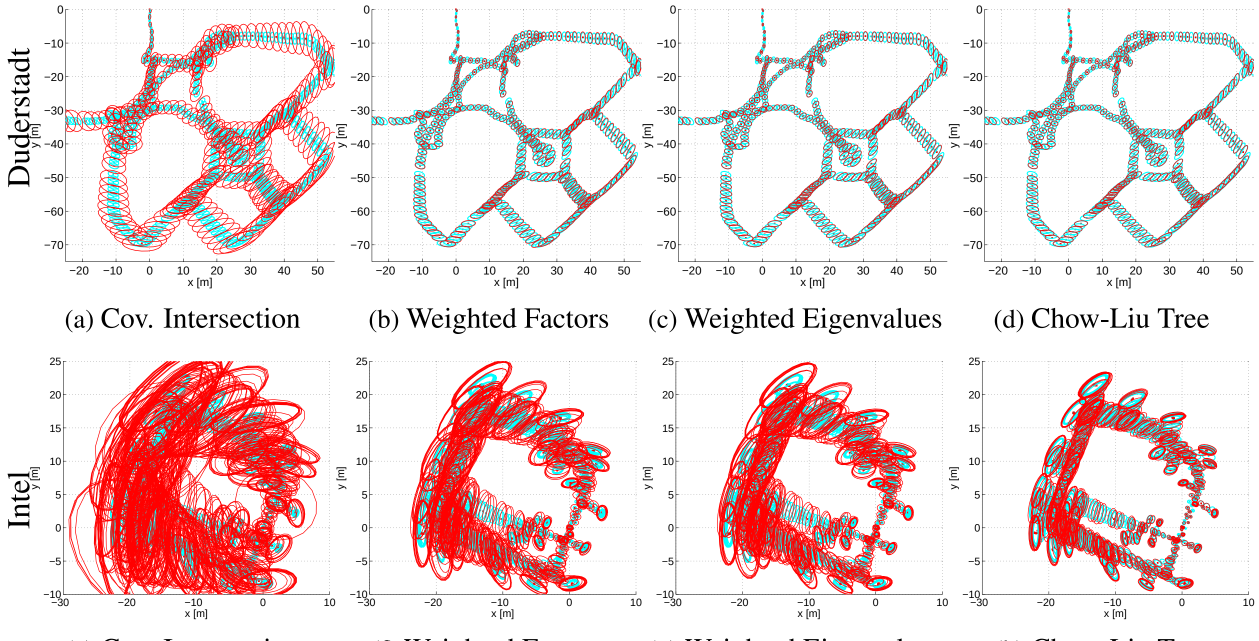
The results for each method, in terms of KLD, are shown in Table 2.3 and Fig. 2.17. The "CLT ratio" columns provide a direct comparison with the CLT, which is not guaranteed to be conservative, but serves as a baseline as it is the minimum KLD distribution among all spanning trees. As one would expect, covariance intersection produces a very high KLD because it is excessively conservative. The weighted factors formulation improves the KLD significantly with respect to covariance intersection, while the weighted eigenvectors formulation improves the KLD further still.

For the Duderstadt, EECS, and Saratoga graphs, the subjective difference in the quality of the estimates is very small, with weighted factors, weighted eigenvalues, and the CLT producing visually indistinguishable results, see Fig. 2.18 top row. For the Intel dataset, and to a lesser extent the Killian and Victoria datasets, there appears to be more room for improvement, with a noticeable difference between the weighted eigenvalues result and the CLT for the Intel graph, see Fig. 2.18 bottom row.

To evaluate the "conservativeness" of the proposed methods, we plot the minimum eigenvalue of the covariance-form consistency constraint for each node marginal, i.e. $\min \lambda(\tilde{\Sigma}_{ii} - \Sigma_{ii})$, depicted in Fig. 2.19. Values below zero indicate overconfidence, with only the CLT producing overconfident results. Covariance intersection, weighted factors, and weighted eigenvalues all produce conservative estimates, with each producing a slightly tighter estimate (closer to zero) than the previous.

Finally, we consider the computational cost of the proposed methods. A plot showing the node removal time as a function of the number of variables in the elimination clique is shown in Fig. 2.20. The average node removal times for covariance intersection, weighted factors, weighted eigenvalues and the CLT were 7, 32, 448, and 5 milliseconds, respectively. Even though covariance intersection and weighted factors were both solving optimization problems over the same number of variables, weighted factors is more expensive. This is due to the fact that the equally-weighted covariance intersection solution, used as the initial point for all optimizations, was often very close

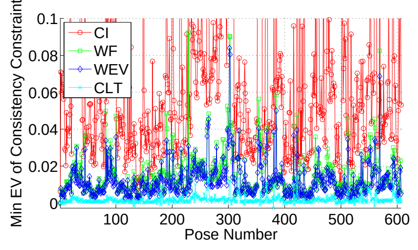
Figure 2.18 Comparison of 3-σ uncertainty ellipses for Conservative GLC. Sample 3-σ uncertainty ellipses are shown for the Duderstadt graph with 50.0% node removal (top row) and the Intel graph with 33.3% node removal (bottom row). True marginal ellipses are shown in cyan, while the marginal ellipses from the approximate distribution are shown in red. For the Duderstadt graph both weighted factors (b) and weighted eigenvalues (c) produce distributions very similar to the CLT (d) while remaining conservative. For the Intel graph both weighted factors (f) and weighted eigenvalues (g) produce similar distributions that are noticeably more conservative than the CLT (h).



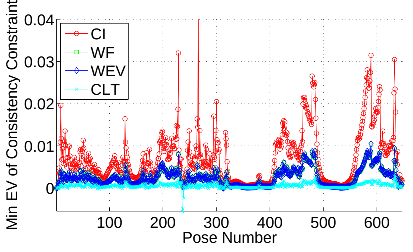
(a) Cov. Intersection (b) Weighted Factors (c) Weighted Eigenvalues (d) Chow-Liu Tree

(e) Cov. Intersection (f) Weighted Factors (g) Weighted Eigenvalues (h) Chow-Liu Tree

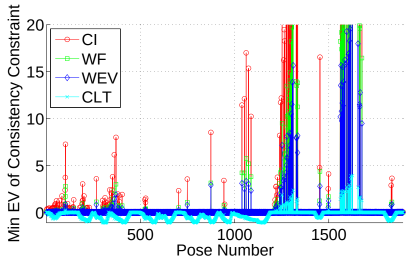
Figure 2.19 Minimum eigenvalue of the consistency constraint after conservative node removal (i.e., $\min \lambda(\tilde{\Sigma}_{ii} - \Sigma_{ii})$). Covariance intersection, weighted factors, and weighted eigenvalues all produce conservative estimates (values greater than zero), with each producing a slightly tighter estimate (closer to zero) than the previous.



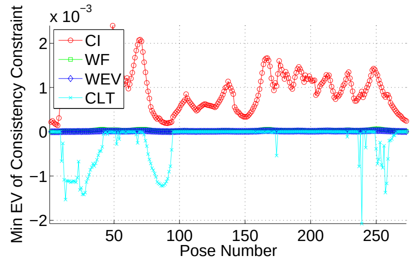
(a) Intel 33.3% Removed



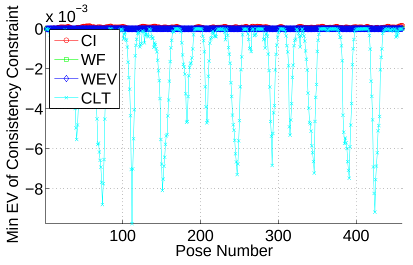
(b) Killian 66.7% Removed



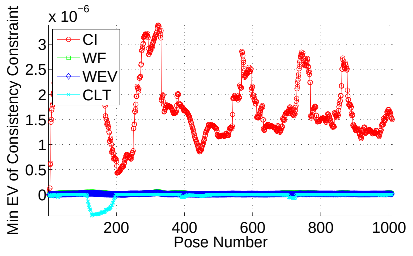
(c) Victoria Park 75.0% Removed



(d) Duderstadt 50.0% Removed



(e) EECS 25.0% Removed



(f) Saratoga 33.3% Removed

to the optimal covariance intersection solution and therefore, covariance intersection converged quickly. Weighted eigenvalues solves a larger optimization problem and therefore is substantially slower. Still, the weighted eigenvalues formulation will often have significantly fewer variables than Vial et al. [173] (which optimizes *all non-zero entries* in the upper triangle of the information matrix) and Huang et al. [73] (which optimizes *every entry* in the upper triangle of the information matrix) for a given information matrix size.

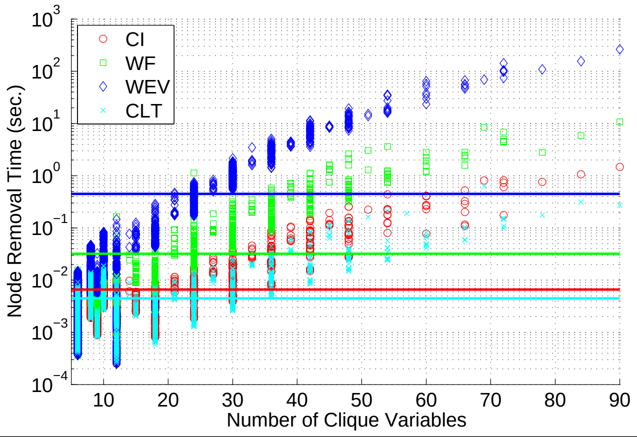
We note that the computational cost of the proposed methods increases quickly with the size of the node removal cliques. However, as experimentally shown in [26], sparse approximate node removal maintains small cliques in real-world SLAM graphs even when removing a very high percentage of nodes. This, in turn, results in essentially constant node removal time regardless of the number of nodes removed and size of the graph beyond the elimination clique.

2.7 Discussion

2.7.1 GLC Node Removal

When considering the application of the GLC node removal, there are a few things to consider:

Figure 2.20 Conservative node removal processing time. Node removal processing time is shown as a function of the number of variables in the elimination clique. Average node removal times (solid lines) for covariance intersection, weighted factors, weighted eigenvalues and the CLT were 7, 32, 448, and 5 milliseconds, respectively.



- When performing GLC, a good linearization point for the relative transforms within the elimination clique must exist. This affects when it is appropriate to remove nodes, especially if performing online node removal. The graph should be optimized as well as possible before node removal. Often it is desirable to remove well-established or "mature" nodes from the graph, instead of nodes that have been recently instantiated and are highly uncertain. Note, however, that this is not a function of node age, but rather whether the graph is sufficiently constrained and optimized to provide a good linearization point.

- Because the target information is often low-rank, we use "pinning" to compute the mutual information when building the CLT and therefore, cannot guarantee that this yields a minimum KLD from the true distribution (though our experimental results show that we achieve a significantly lower KLD than other state-of-the-art methods).

- When removing a set of nodes it is important to note that the order in which they are removed affects the resulting graph connectivity. Experimentally, we found that removing long chains of nodes sequentially sometimes produced large star shaped trees in the graph. To avoid this, sets of nodes were removed in a randomized order in all experiments. The variable elimination ordering problem [92] is well studied for dense node removal. The application and adaptation of existing variable elimination ordering strategies for node removal with sparse connectivity could further improve the performance of GLC-based complexity management schemes.

2.7.2 Guaranteed Conservative Sparse Approximations

- The CLT approximation itself is not guaranteed to be conservative and therefore CLT sparse-approximate GLC node removal does not guarantee a conservative estimate. In fact, our results showed that CLT-based GLC sparse approximation can be either slightly conservative (Fig. 2.7 and Fig. 2.12(a) and (c)), or slightly over-confident (Fig. 2.12(b)). While our proposed GLC method avoids inconsistency pitfalls associated with measurement compounding, and accurately recreates the CLT, it may still be slightly overconfident if the CLT approximation cannot represent all of the true correlation within the clique. We have found experimentally that the CLT performs well on most graphs, and only results in noticeable overconfidence in graphs with large, dense cliques. It is in these situations where the guaranteed-conservative methods proposed in §2.5 are most appropriate.

- All of the conservative methods start with the Chow-Liu tree as their basis. We believe that this is a reasonable starting point as it is the minimum KLD spanning tree. However, there is no guarantee that, after using the optimization based methods to ensure a conservative estimate, the CLT's sparsity pattern remains optimal. Furthermore, other non-tree sparsity patterns may be of interest. This is one strongly appealing aspect of the $\mathcal{L}_1$ regularization method proposed in [73] in that it automatically selects the sparsity pattern.

- There is a trade off between the complexity of the optimization problem and the accuracy of the approximation it is able to achieve. Based upon our experimental results, we feel that the weighted factors formulation provides the best trade off between KLD and computation time, as weighted eigenvalues only performs marginally better while being substantially more computationally expensive.

- In this chapter, the guaranteed-conservative optimization algorithms are presented in the context of sparsifying the dense cliques produced by node marginalization. However, these techniques could be applied to portions of the graph to perform sparsification, without removing nodes.

2.8 Chapter Summary

In this chapter, we presented a factor-based method for node removal in a wide variety of SLAM graphs. This method can be used to alleviate some of the computational challenges in performing inference over long-term graphs by reducing the graph size and density. The proposed method is able to represent either exact marginalization, or a sparse approximation of the true marginalization, in a consistent manner over a heterogeneous collection of constraints. We experimentally evaluated the proposed method over multiple real-world SLAM graphs and showed that it outperformed other state-of-the-art methods in terms of KLD.

CHAPTER III

Learning Visual Feature Descriptors for Dynamic Lighting Conditions

In many robotic applications, especially long-term outdoor deployments, the success or failure of feature-based image registration is largely determined by changes in lighting. In this chapter, we present a method to learn visual feature point descriptors that are more robust to changes in scene lighting than standard hand-designed features. We demonstrate that, by tracking feature points in time-lapse videos, one can easily generate training data that captures how the visual appearances of interest points change with lighting over time. This training data is used to learn feature descriptors that map the image patches associated with feature points to a lower-dimensional feature space where $\mathcal{L}_2$ distance provides good discrimination between matching and non-matching image patches. Results showing that the learned descriptors increase the ability to register images under varying lighting conditions are presented for a challenging indoor-outdoor dataset spanning 27 mapping sessions over a period of 15 months, containing a wide variety of lighting changes. This chapter is based on our work published in [28].

3.1 Introduction

Standard hand-designed visual features such as scale invariant feature transform (SIFT) [104] and speeded up robust features (SURF) [9] detect key-points in an image and then describe the local visual appearance of these key-points as a vector. Image registration can then be performed by matching the key-points between images by comparing the $\mathcal{L}_2$ distance between the descriptors. In order for matching to be successful, the key-point detector and descriptors must be at least partially robust to common image variations such as scale, rotation, viewpoint, and lighting changes. Invariance with respect to scale and rotation are usually accounted for at the feature detection stage, where key-points will be detected at a canonical scale and orientation. The description stage then focuses on representing the appearance of the local region around the key-point such that the descriptor is discriminative while being robust to viewpoint and illumination changes.

In this chapter we focus on increasing the robustness of feature point description to lighting changes. Hand-designed descriptors such as SIFT and SURF have limited lighting invariance—often allowing for affine transformations in image intensity by considering the gradient of intensity, and through other mechanisms such as mean subtraction and normalization. However, in general, the change in appearance caused by lighting affects the image intensity in a complex, nonlinear way.

In many robotic applications, the success or failure of feature-based image registration is largely determined by changes in lighting. This is especially true for medium to long-term outdoor applications, where the scene structure has not changed dramatically, but images separated by even a few hours may be unmatchable due to cyclical changes in lighting. This phenomenon is illustrated in Fig. A.2(a), which shows example imagery from three different locations in our experimental dataset. In this dataset, only a small fraction of the possible matches are successfully registered using standard features, largely because of cyclical changes in lighting.

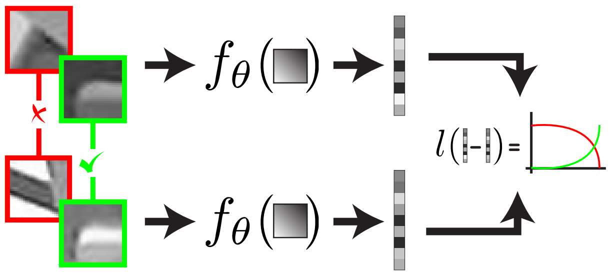
In this work, we seek to learn a feature descriptor that is more robust to changes in local image appearance caused by lighting (Fig. 3.1). To observe how the local appearance of image patches changes under dynamic lighting conditions, we first track key-points and their associated image patches through time-lapse video using a representative training dataset. We then train a feature descriptor using matching and non-matching pairs of image patches sampled from these patch tracks. A contrastive cost function is used so that matching patches are mapped close together (in terms of Euclidean distance in feature space) while separating non-matching patches. The resulting descriptor is more robust to the types of lighting variation observed in the training data.

The remainder of this chapter is outlined as follows: In Section 3.2, we discuss existing work related to the proposed method. The descriptor learning method is described in Section 3.3. Section 3.4 contains details of the training process, including the collection of training data. Experimental results are provided in Section 3.5. Finally, a discussion and concluding remarks are provided in Sections 3.6 and 3.7.

3.2 Related Work

Given the limitations of existing visual feature descriptors, several proposed methods address the difficulties in matching images collected under varying lighting conditions at a systems level.

Figure 3.1 Illustration of visual feature learning method. Pairs of image patches labeled either as matching (green) or non-matching (red) are supplied as input to a feature descriptor function, $f_{\boldsymbol{\theta}}(\,\cdot\,)$, parameterized by $\boldsymbol{\theta}$, that maps the input patch to a feature vector. A contrastive cost function, $l(\,\cdot\,)$, based on the Euclidean distance between the feature vectors, encourages matching feature vectors to be close together in feature space while encouraging non-matching features to be far apart. By learning parameters $\boldsymbol{\theta}$ that minimize this cost function, we produce a mapping to a feature space where Euclidean distance captures the similarity and differences amongst the training pairs. By training with data that includes variation due to changes in lighting, the feature descriptor learns to be robust to lighting variation.



In a mapping and navigation context, both Konolige and Bowman [94] and Churchill and Newman [37] add new example views or visual "experiences" when the current view cannot be registered against previous views. This addresses the problem of changing lighting by capturing several examples of how a location might look under different lighting conditions. Similarly, in Johns and Yang [80, 81], locations are modeled with a collection of features observed at different points in time. These works are mostly orthogonal to the proposed method, and would benefit from features that are more robust to lighting change, because better features reduce the number of samples needed to model a location.

Several recent works have investigated whole image place recognition under changing appearance conditions, including [81, 100, 113, 128]. In Lategahn et al. [100], a set of standard descriptor "building blocks" is defined. Place recognition performance is then optimized by searching the space of possible descriptors constructed from these building blocks. Neubert et al. [128] attempt to predict how a location will look at a different point in time by learning a mapping between appearance codewords. They then perform place recognition between the current image and the predicted image. The formulation, however, focus on changes between two distinct states (e.g., summer and winter) and not continual changes such as those caused by lighting. In Milford et al. [113], whole image place recognition is performed over extreme changes in lighting from day to night by aggressively down-sampling and contrast-normalizing the images before comparison.

In this work, we focus on geometric registration through point correspondence as opposed to whole image place recognition. For some applications, like loop-closure detection in metric mapping, even if one can recognize places under a high degree of lighting variation, it may not be useful if one cannot extract a metric estimate of the motion between the camera views [142]. It is worth noting that the feature descriptors learned using our proposed method could be used in a bag-of-words model [132, 151] for place recognition. However, evaluating if this would improve the robustness with respect to lighting remains future work.

Many methods have been proposed that leverage machine learning to improve the performance of feature descriptors [5, 20, 72, 142, 168, 180, 181]. In Babenko et al. [5], feature matching is cast as a binary classification problem where one attempts to determine if two image patches do or do not match. Boosting is then used with a set of simple hand-designed features to learn a classifier appropriate for a specific domain. In Hua et al. [72], Winder et al. [180], Winder and Brown [181], and Brown et al. [20], the parameters of fixed descriptor pipelines (often a variant of the DAISY descriptor [165]) are optimized to improve descriptor performance. Similarly, in Stavens and Thrun [157], the parameters of standard descriptors, including SIFT, are optimized for specific domains. Ranganathan et al. [142] use the fine vocabulary method of [111] to learn a probability distribution over visual words in an attempt to capture which visual words can be produced by the same scene feature under various lighting conditions. Standard place recognition and feature matching are then reformulated to account for the learned distribution. Both Philbin et al. [139] and Shakhnarovich [149] learn an embedding on top of SIFT features. This is similar to the proposed method except that we learn an embedding directly from the raw pixel input as opposed to on top of a hand-designed feature descriptor. The recent work by Trzcinski et al. [168], which uses boosting to learn a binary descriptor, is most similar to our proposed method in that it learns a descriptor directly from raw pixel data in a supervised setting. However, our proposed method differs in its learning method, descriptor model, and in its focus on robustness to changes in lighting.

To learn an illumination robust feature descriptor we employ a training scheme referred to as a "Siamese" network [19, 34, 66, 117, 159], with the goal of minimizing a contrastive cost function [34, 66, 159] that encourages a nonlinear mapping to a lower-dimensional space where matching features are close together and non-matching features are far apart in Euclidean distance. This goal is often referred to as embedding learning, manifold learning, or distance metric learning.

Siamese networks have been employed in a wide range of applications including signature verification in Bromley et al. [19], face recognition in Chopra et al. [34], and object recognition in Hadsell et al. [66] and Mobahi et al. [117]. An especially compelling result was presented in Taylor et al. [159] where a system was trained that could recognize similar human poses while being highly invariant to other distractors, including changes in lighting.

Beyond Siamese networks, auto-encoder frameworks can also be used to learn nonlinear embeddings, as shown in Hinton and Salakhutdinov [70] and [146]. However, with auto-encoders the goal is to produce a lower-dimensional embedding that can be decoded with minimal reconstruction error, which does not necessarily produce embeddings where Euclidean distance is useful for discrimination [159]. Salakhutdinov and Hinton [145] provide an interesting model that blends a Siamese network with an auto-encoder for regularization.

The contrastive cost function employed here is just one option for learning an embedding. In Goldberger et al. [63], a linear model is optimized in order to minimize a probabilistic version of k-nearest neighbors classification error. A probabilistic loss function based on Kullback-Leibler divergence (KLD) is provided in Hinton and Roweis [69].

3.3 Learning a Feature Descriptor

In this section, we first describe the Siamese network framework that can be used to learn a wide variety of feed-forward descriptor models (Fig. 3.1). We then discuss the specific feature descriptor models considered in this work. As in [5, 20, 72, 180, 181], we focus on the description of the image patch associated with key-points provided by an existing key-point detector[1]. We assume that the detector provides us with a pixel location, a scale, and optionally a canonical orientation to allow for rotation invariance[2]. Given this information we extract an appropriate patch from the image for each key-point. The image patch then becomes the input for the learned descriptor. At this point we assume that we have pairs of patches labeled as either matching or non-matching. We detail how these pairs can be easily generated in §3.4.

3.3.1 Learning with a Siamese Network

First, we define a feature descriptor that maps an image patch $\mathbf{x}$ to a feature vector $\mathbf{y}$ as

$$\mathbf{y} = f_{\boldsymbol{\theta}}(\mathbf{x}),$$

where $\boldsymbol{\theta}$ parameterizes the descriptor. During training we work with pairs of training examples that are known to be matching or non-matching. Let $\mathbf{x}_i$ and $\mathbf{x}_j$ be two training image patches. The current function is then used to describe each patch,

$$\mathbf{y}_i = f_{\boldsymbol{\theta}}(\mathbf{x}_i) \text{ and } \mathbf{y}_j = f_{\boldsymbol{\theta}}(\mathbf{x}_j).$$

[1] We use the SURF detector throughout our experiments.

[2] In our experiments we do not exploit the canonical orientation as we focus on robotic applications where the imagery does not undergo large rotations.

We then consider the squared Euclidean distance in feature space

$$d_{ij}^2 = \|\mathbf{y}_i - \mathbf{y}_j\|_2^2.$$

Using the contrastive cost function from [66],

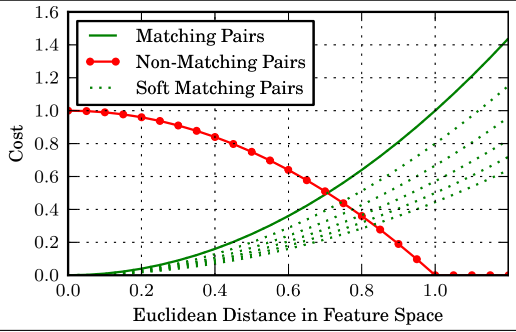
$$l_{\boldsymbol{\theta}}\left(\mathbf{y}_i, \mathbf{y}_j\right) = \begin{cases} s_{ij} d_{ij}^2, & \text{if matching} \\ \max\left(1.0 - d_{ij}^2,\ 0\right), & \text{if non-matching} \end{cases} \tag{3.1}$$

where the similarity score s_{ij} between the matching pairs is set as proposed in [159][3] (since we are training with temporal sequences). We define s_{ij} based on the difference in time between when the two patches were observed,

$$s_{ij} = \frac{1}{1 + \alpha |t_i - t_j|}, \tag{3.2}$$

where t_i and t_j are the observation times of the training patches in hours and α is a scale factor controlling the time scale of the similarity weight. In our experiments, we selected $\alpha = 1/8$ h. Taylor et al. [159] experimentally demonstrated that this “soft” similarity allows the embedding to better capture the temporal similarity in appearance. They also experimentally showed that this soft similarity improved training results. The resulting cost function (3.1) is illustrated in Fig. 3.2.

Figure 3.2 Contrastive cost function. The cost for matching pairs is shown in green and the cost for non-matching pairs in red. The dashed green lines show the matching cost function using similarity weighting (3.2) with $\alpha = 1/8$ h for $|t_i - t_j| = [2, 4, 5, 6, 10]$ h.



If we consider a training set of N training pairs, the learning objective becomes

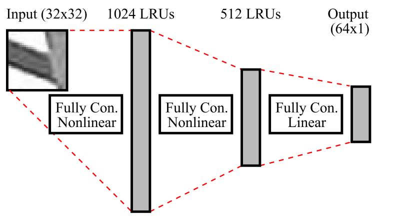
$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \arg\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{n=1}^{N} l_{\boldsymbol{\theta}}\left(\mathbf{y}_i^n, \mathbf{y}_j^n\right). \tag{3.3}$$

[3]Hadsell et al. [66] set $s_{ij} = 1$ to treat all matching pairs evenly.

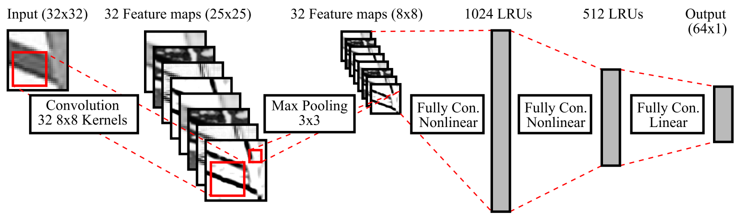
The learning objective is a weighted sum of positive pairwise terms, $l_{\boldsymbol{\theta}}\left(\mathbf{y}_i^n, \mathbf{y}_j^n\right) \geq 0$, therefore, the loss function is also positive, $\mathcal{L}(\boldsymbol{\theta}) \geq 0$. In general, the minimum of the objective will be greater than zero, $\mathcal{L}(\boldsymbol{\theta}) > 0$, for two reasons: First, our feature descriptor, $f_{\boldsymbol{\theta}}(\mathbf{x})$, may not have enough complexity to learn an ideal transform that achieves $\mathcal{L}(\boldsymbol{\theta}) = 0$. Second, if positive and negative pairs share patches, the interaction between the pairwise terms may not allow for all terms to be at their minimum simultaneously. Hadsell et al. [66] discuss an intuitive interpretation of the contrastive cost function where positive pairs are pulled together by springs while negative pairs are pushed apart. The optimum value of the cost function is achieved when this system is at equilibrium. We optimize the objective using stochastic gradient descent, the details of which are described in §3.4.

3.3.2 Feature Descriptor Models

Figure 3.3 Feature descriptor models.



(a) Multi-Layer Perceptron (MLP)



(b) Convolutional Multi-Layer Perceptron (CMLP)

In our experiments we consider two standard model classes for the learned feature descriptor; a multi-layer perceptron (MLP) and a convolutional multi-layer perceptron (CMLP) [101] (Fig. 3.3). The MLP consists of multiple fully-connected layers, each performing a nonlinear transformation on the output of the previous layer. If we denote the input to each hidden layer as $\mathbf{h}_{i-1}$ and the output as $\mathbf{h}_i$ then

$$\mathbf{h}_i = g\left(\mathrm{W}_i \mathbf{h}_{i-1} + \mathbf{b}_i\right),$$

where W_i is a matrix defining a linear transform, $\mathbf{b}_i$ is a bias vector, and $g(\,\cdot\,)$ is a nonlinear activation function applied in an elementwise fashion to its input vector. This layer is parameterized by $\boldsymbol{\theta}_i = [\mathrm{W}_i, \mathbf{b}_i]$, which it contributes to the parameters of the overall model. For the first layer the input will simply be the raw image patch as a vector, $\mathbf{h}_0 = \mathbf{x}$.

The CMLP expands upon the MLP by adding convolutional and pooling layers. The convolutional layers exploit the fact that the statistics of natural images can be considered stationary over the location in the image. Instead of learning the parameters of a function of the whole image, weights are learned for kernels that are convolved with the image to produce a number of feature maps. This greatly reduces the number of parameters in the model without significantly reducing its representational capacity. For a detailed description we refer the reader to [101]. The pooling layers perform a spatial subsampling that reduces the size of the input for subsequent layers and provides invariance to small translational shifts in the input. In the proposed models we use non-overlapping max pooling, which performed slightly better than mean pooling. It is interesting to note that the CMLP structure is very similar to that of many hand-designed feature descriptors [9, 104, 165], which often include a convolution filtering stage (e.g., computing oriented gradients) and a pooling stage (e.g., spatial binning or averaging).

In both the MLP and CMLP we use rectifying nonlinearity, referred to as a linear rectified unit (LRU) [125],

$$s(x) = \max(x, 0),$$

which we found to be quicker to train than hyperbolic tangent, or sigmoid nonlinearities. Additionally, both the MLP and CMLP have a linear output layer.

3.3.3 MNIST Example

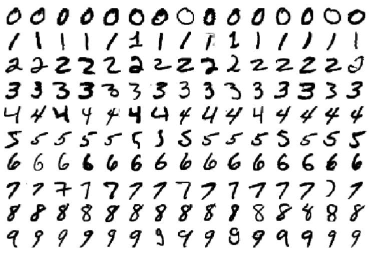
To demonstrate the Siamese network learning method described in §3.3.1, we present results using the MNIST handwritten digit dataset (Fig. 3.4(a)).[4] We randomly selected 50,000 digit pairs from the dataset and for each pair we consider the digits to be matching if they are from the same class (i.e., the same numeral 0 to 9). We then train the MLP descriptor shown in Fig. 3.3(a). The only modification is to reduce the descriptor's output dimension from 64 to 2. This allows us to directly visualize the resulting feature space for a set of previously-unseen test digit images. We see in Fig. 3.4(b) how the contrastive loss function drives matching digits close together in feature space while pushing non-matching digits apart—producing a feature space where the Euclidean distance provides good discrimination between classes.

3.4 Training the Models

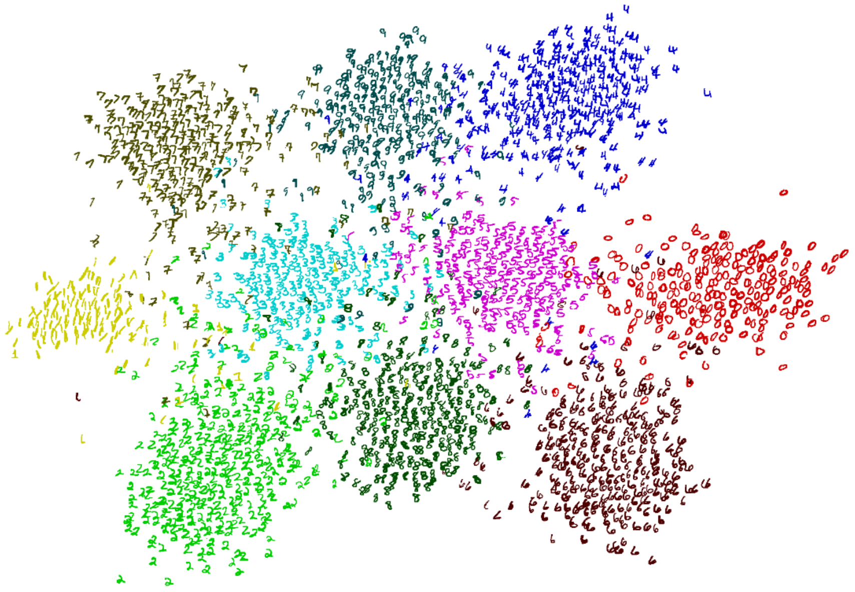
In this section, we describe how we produce training data by tracking interest points in time-lapse videos. We also provide the details of the stochastic gradient descent learning.

[4]The MNIST dataset is publicly available from http://yann.lecun.com/exdb/mnist/index.html

Figure 3.4 Learning method demonstrated on the MNIST dataset



(a) Example MNIST Digits



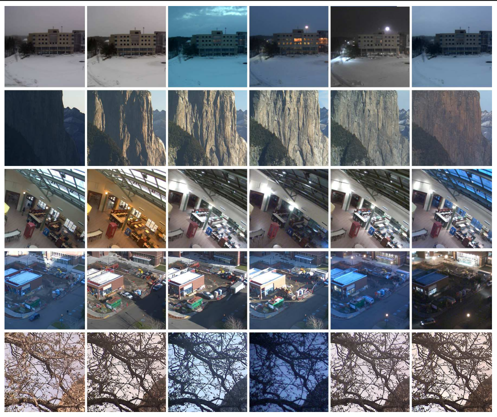
(b) Learned 2D Feature Space

3.4.1 Generating Training Data

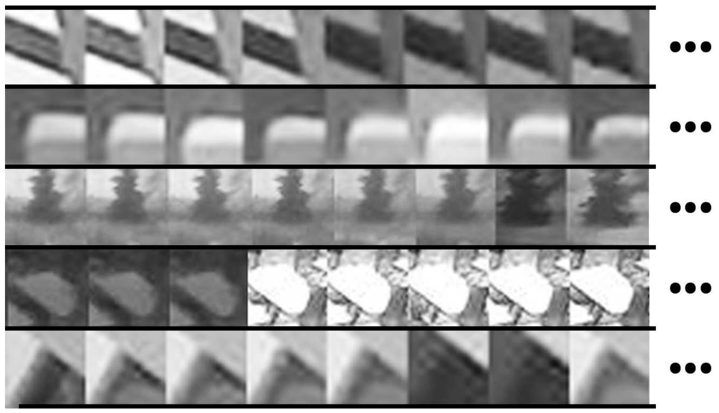
Many methods have been proposed to generate a training set of image patches. In [20, 72, 180, 181], 3D reconstructions are used to establish correspondence between patches in the source images. In [139], image-to-image feature-based matching with outlier rejection is used in order to generate training data, since they seek only to learn an encoding on top of existing feature descriptors. Images with known pose are used in [100]. It is also possible to generate sequences of image patches by tracking interest points in video [157, 184] or by sliding a window through static images [11].

In this work we elect to generate patches by tracking interest points in video. In order to capture the changes in appearance caused as the lighting changes with time, we use time-lapse videos. To generate these videos we downloaded imagery from stationary webcams at a fixed rate. In total, 230 different webcam locations were used, including mostly outdoor scenes, both natural and urban, as well as some indoor scenes. Imagery was downloaded every 20 minutes for 72 hours. Sample imagery from five locations is shown in Fig. 3.5(a). Of the 230 locations, 184 were used to generate training data, 23 for validation and 23 for testing in §3.5.

Figure 3.5 Sample images and patch tracks from the webcam dataset. Sample images from five locations are shown in (a). Note that the images have been sub-sampled so the difference in time between images is larger than the 20 minutes used during patch extraction. Five sample patch tracks are shown in (b). Note that the tracks will be of varying lengths, and only subsections are shown here.



(a) Sample Webcam Images



(b) Sample Patch Tracks

3.4.1.1 Tracking interest points in time-lapse videos

Given a sequence of webcam frames, we track features through time as follows:

- We detect interest points in each incoming image. In our experiments we use the SURF detector, however, any detector that provides location and scale would be acceptable. (Note that we do not use the SURF descriptor for tracking). One could also use the canonical orientation of an interest point detector in order to achieve some degree of rotation invariance; however, we do not as our target application uses a ground robotic platform that does not undergo large rotations.

- For each interest point we attempt to associate it with an existing track. Because the imagery is collected from a static viewpoint we can use several simple criteria, similar to those proposed in [20, 180]. First, the interest point must be within 5 pixels of the most recent observation of the track. Second, the scale of the interest point must be within $\pm 50\%$ of the most recent observation's scale. Third, the difference in time between a new patch and the most recent observation of a track can be no more than 1 hour. These criteria are often adequate to uniquely associate a new patch with an existing track. If there are still multiple candidates, we select the track that minimizes

$$\frac{r + r_t - \|\mathbf{x} - \mathbf{x}_t\|_2}{2\max(r, r_t)},$$

where r and r_t are the radii of the interest point and track, respectively, and $\mathbf{x}$ and $\mathbf{x}_t$ are the locations of the interest point and track, respectively. If we cannot find a valid existing track, a new track is created based upon that patch.

- After processing each image we consider the current tracks. Tracks that have been updated recently are kept for association in future images. Tracks that have not been updated in an hour are no longer updated and are wrapped up and saved.
- Because we wish to emphasize the temporal change in the dataset, we downsample the final tracks by a factor of two, increasing the time between each sample in the track from 20 minutes to 40 minutes.

The results of this process are illustrated in Fig. 3.5(b). Using this processing pipeline on the entire webcam dataset produced approximately 3.1 million feature tracks (2.5 million for training, 0.3 million for validation and 0.3 million for testing) with an average of approximately 5 patches per track.

3.4.1.2 Generating training pairs from tracks

Given a set of patch tracks, it is easy to generate a very large set of matching and non-matching pairs for training. Starting with all feature tracks we randomly sample pairs of tracks without replacement. From a pair of tracks we then randomly select two matching pairs (one from each set) and two non-matching pairs (from between the two sets). This produces an even number of matching and non-matching pairs in the dataset. We repeat this process until all patch tracks have been used at least once. This ensures that each track is used.

Given the combinatorially huge number of possible pairs of tracks, and possible pairs within each track, this process can be repeated multiple times. During training we continuously sample new pairs.

3.4.1.3 Augmenting the training data with viewpoint variation

The webcam dataset does a good job of capturing the changes patches undergo with respect to lighting; however, because the videos are captured from static locations, they do not contain any viewpoint variance. To account for the lack of viewpoint variance we augment the patches extracted from the webcam dataset using the viewpoint variant patches provided in Brown et al. [21]. This dataset provides an additional 0.9 million patch pairs (which we divide evenly between training, testing, and validation). Another option would be to use the existing image patches with synthetic affine warps, which has been shown to produce good results in [137].

3.4.2 Training Descriptor Models

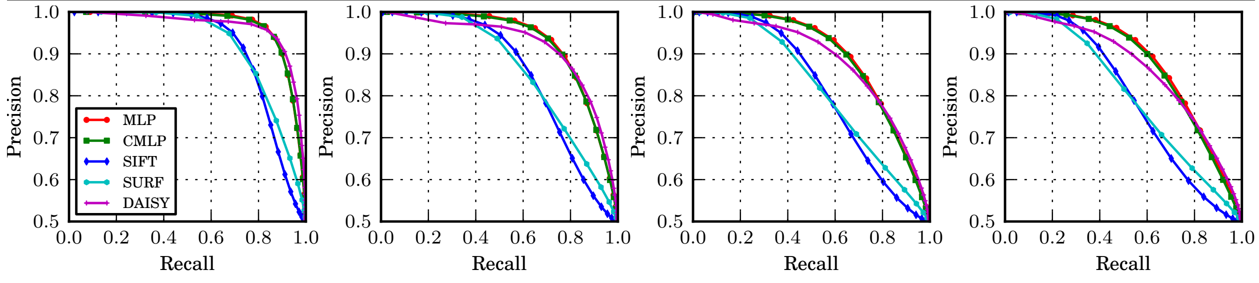
Batch stochastic gradient descent was employed in order to optimize the model parameters in (3.3). We used a batch size of 1000 pairs with a learning rate of $\lambda = 0.1$ and momentum of $\beta = 0.9$, producing an update procedure at step k of

$$\begin{aligned}\mathbf{v}_{k+1} &= \beta \mathbf{v}_k - \lambda \frac{\partial \mathcal{L}_k}{\partial \boldsymbol{\theta}_k} \\ \boldsymbol{\theta}_{k+1} &= \boldsymbol{\theta}_k + \mathbf{v}_{k+1},\end{aligned}$$

where $\frac{\partial \mathcal{L}_k}{\partial \boldsymbol{\theta}_k}$ is the gradient of the objective function (3.3) with respect to the parameters $\boldsymbol{\theta}$ over the kth batch of training data. Training was implemented using Theano [12], which allows for automatic differentiation of the objective function and GPU-based evaluation of the feature descriptor models.

3.5 Experimental Evaluation

Figure 3.6 Precision and recall for learned features. By comparing the precision-recall curves for pairs from the 23 test webcam locations approximately 1 h, 4 h, 8 h, and 12 h apart, we see that the performance of the proposed learned features degrades gracefully as the time between images increases, especially in the region above 90% precision.



(a) 1 h Between Images (b) 4 h Between Images (c) 8 h Between Images (d) 12 h Between Images

We evaluate the proposed feature descriptor on two datasets. The first consists of 23 webcam locations not used during training. This dataset is used to evaluate how temporal changes in lighting affects matching. The second dataset consists of data collected by a ground robot and allows us to compare the descriptor's performance in a challenging real-world environment.

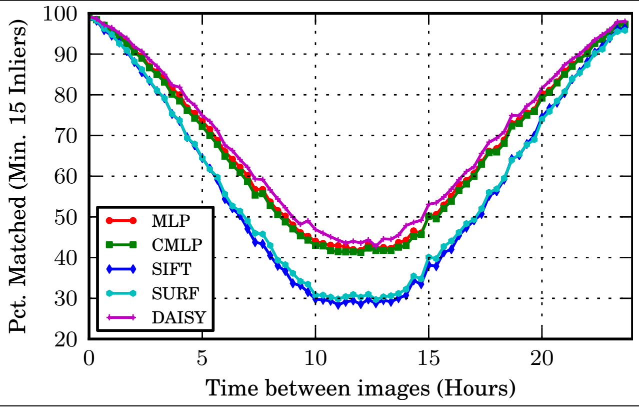
In addition to the proposed descriptors, we compare against SIFT [104], SURF [9], and DAISY [165]. For the DAISY descriptor we use learned parameters provided by [180], specifically the "T1-4-2r8s" version as it has an output dimension of 68 and computation time comparable with our learned descriptors. In comparison, SIFT has a dimension of 128 while SURF and both of the learned descriptors produce 64 dimensional vectors. We also attempted to compare with another learned descriptor, DIRD [100]; however, DIRD was optimized with respect to whole image place recognition and was not effective for point-to-point geometric image registration. In order to focus on the properties of the descriptors, the same key-points (which were detected using the SURF detector) were used for all feature descriptors.

3.5.1 Webcam Dataset

We first explore the performance of the feature descriptors using imagery from 23 webcam locations that were not used during training. Because the webcam data was collected frequently, it allows us to evaluate performance with respect to the time between images.

Using matching and non-matching pairs from this test set, we sweep out the precision and recall curve for a descriptor by classifying points as matching or non-matching with a varying distance threshold. We see in Fig. 3.6 that when the time between image pairs is small, all of the methods perform well, with the learned descriptors and DAISY having the best performance. However, as we increase the time between image pairs, the two learned methods degrade gracefully, maintaining good precision-recall curves in the challenging region around 12 hours. Note that this is especially true in the region above 90% precision where we would like to operate so that matching is not overwhelmed by outliers.

Figure 3.7 Matching results for webcam dataset. Results are averaged from exhaustive pairwise matching at 23 webcam locations.



To evaluate the features in an image registration context we attempt to register all pairs of images at each location that were collected within 24 hours of each other. The ability to match images in this situation is driven primarily by the change in lighting throughout the day. So over the course of 24 h, the ability to register images will start high, and gradually reduce as the time between images increases, hitting a minimum at about 12 h before increasing as time-of-day lighting conditions return to those most similar after 24 h. For indoor locations the patterns might be less distinct, however, many indoor locations still go through similar cycles caused by working hours and light through windows.

When matching features we perform nearest-neighbor matching based on Euclidean distance and only include matches that pass a second-nearest-neighbor test [104] with a threshold of 0.7. Inliers and outliers can be easily determined based on a distance threshold of 10 pixels because images in this dataset were collected from a static viewpoint.

In Fig. 3.7 we consider the percentage of pairs that could be registered with a minimum of 15 inliers (essentially a practical "bare-minimum" to reliably compute an estimate of camera motion). We see that, in the most challenging region, around 12 h between image pairs, the learned feature descriptors (CMLP and MLP) successfully match over 40% of possible pairs. DAISY performed ever-so-slightly better in this region matching just under 45%. SIFT and SURF match significantly fewer pairs, about 30%. Similar patterns were observed with other minimum inlier thresholds, though the percentage of matches decreases significantly as the minimum required number of inliers increases.

Note that Fig. 3.7 is smooth because it averages many different pairs, from different locations, with different starting points throughout the day. With smaller sample sizes the relationship between matching and time between images can vary dramatically, i.e., two images collected 8 hours apart at night might match easily, while two images collected 1 hour apart before and after sunset will not be matched.

3.5.2 North Campus Long-Term Dataset

One caveat of the previous experiment is that the webcam imagery was taken from a static viewpoint. In order to evaluate the feature descriptors in a more realistic setting we consider their performance on imagery collected by a robotic platform. The imagery was collected in 27 sessions over the course of 15 months on University of Michigan's North Campus (Fig. A.1). This dataset contains a wide variety of lighting conditions ranging from early morning to just after dusk. Additionally, this data includes viewpoint variance and additional challenges caused by moving objects, seasonal changes, and even construction projects. Given known robot pose, the dataset is

Figure 3.8 North Campus dataset matching results. The percentage of matching pairs is averaged over 500 locations in the North Campus dataset.

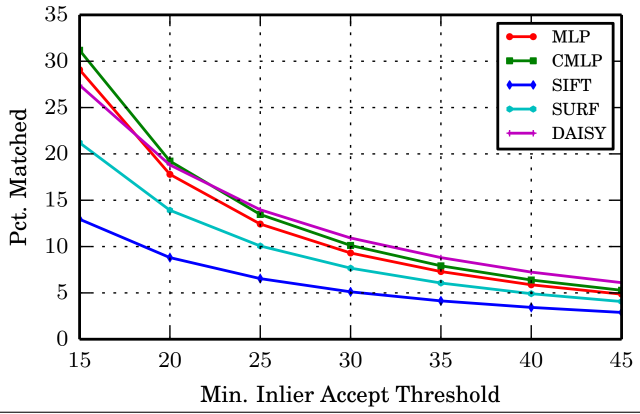
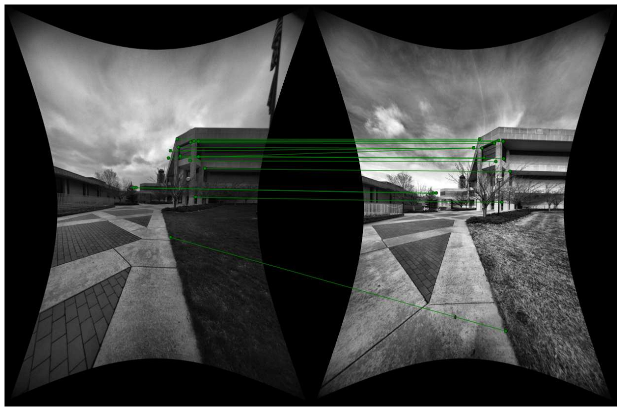


Figure 3.9 Sample image pair registered with learned feature. This image pair was successfully registered using the CMLP descriptor, but not SIFT, SURF, nor DAISY.



(a) Sample Image Registration

(b) Sample Matching Patches

split up into 500 locations with an average of 37 images per location.

At each location we match all pairs of images. As before, when matching features we perform nearest neighbor matching based on Euclidean distance and employ the second-nearest-neighbor test with a threshold of 0.7. Outliers are rejected by fitting an Essential matrix using random sample consensus [68].

In Fig. 3.8 we show the percentage of image pairs successfully matched as a function of the minimum number of inliers. Here, we see that again CMLP, MLP, and DAISY provide the best results, matching around 30% of possible pairs at the lowest threshold. SIFT and SURF are significantly less successful. An example image pair that was successfully registered using the CMLP descriptor, but not SIFT, SURF, nor DAISY, is shown in Fig. 3.9.

3.5.3 Computation Time

Finally, we provide the computation time of the learned features in Table 3.1. The learned descriptors were developed using Theano and therefore can be computed using the CPU or GPU. For SIFT and SURF we evaluated with OpenCV's CPU version and timing information is provided only as a rough comparison—well-optimized GPU versions of both are readily available.

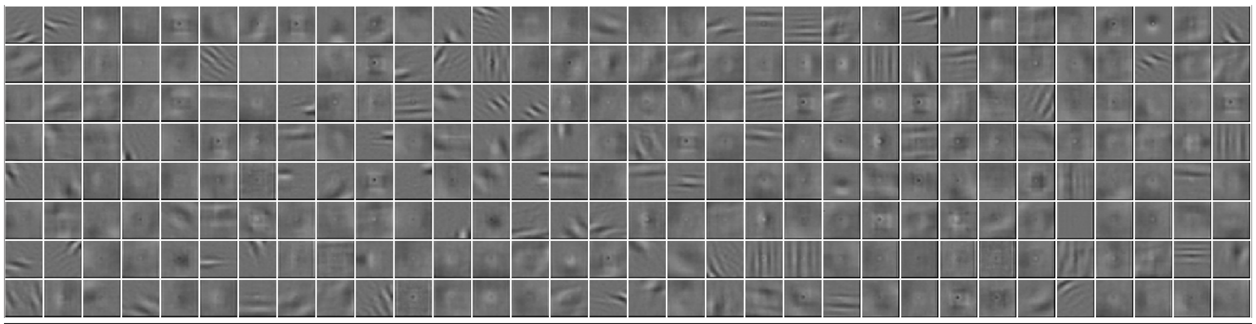
Table 3.1 Mean Feature Extraction Time

	CPU	GPU
MLP	0.68 ms/feature	0.07 ms/feature
CMLP	1.34 ms/feature	0.27 ms/feature
SIFT	0.64 ms/feature	—
SURF	0.20 ms/feature	—
DAISY	0.65 ms/feature	—

3.5.4 Qualitative Analysis

Having seen that the learned feature descriptors provide improved performance over standard hand-designed features we will now try to provide some qualitative insight into what the descriptors are learning. The features learned by the early stages of the MLP and CMLP descriptors are shown in Fig. 3.10 and Fig. 3.11, respectively. In Fig. 3.10, each of the 1024 patches represents the input that most strongly activates the corresponding feature in the first layer of the network. We can see that the features in this first layer are mostly composed of oriented edges and blobs of varying frequency. Features in the second and third layers of the network will be non-linear combinations of these first-level features.

Figure 3.10 MLP first layer features. Each patch represents the input patch that most strongly activates the corresponding feature in the MLP's first fully-connected layer. Note that a random subset of 256 out of the 1024 first-level features are shown.



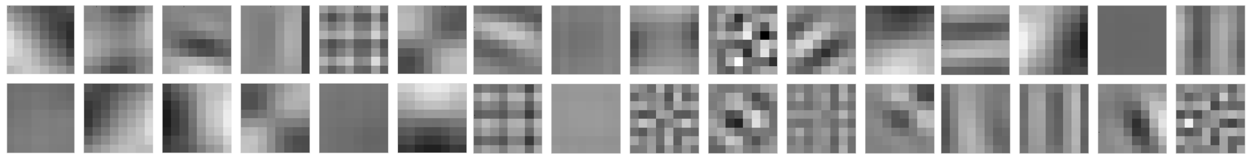
The CMLP descriptor adds an additional convolutional layer to the MLP. The learned convolutional filters are shown in Fig. 3.11(a). Again, we see that the network learns a variety of oriented edge filters with varying scale, orientation, and frequency. The output of the first layer convolutional filters is shown in Fig. 3.11(c) for three sample patches (Fig. 3.11(b)). These convolutional features are fed into the first fully-connected layer with 1024 outputs. In Fig. 3.11(d), we show the features that most strongly activate 10 of the 1024 features in the first fully-connected layer. Here each row represents an input to the fully-connected layer.

To visualize the learned feature space we use t-distributed stochastic neighbor embedding (t-SNE) [170] to reduce the dimensionality of output feature space from 64 to 2. This allows us to plot a representation of the learned feature space where patches from the validation set are arranged spatially according to the location of their associated feature descriptor. The t-SNE visualizations for the MLP and CMLP feature descriptors are shown in Fig. 3.12 and Fig. 3.13, respectively. In both cases, we see that patches with similar appearance are mapped to similar portions of the feature space. We have highlighted regions with similar textures, oriented edges, rectangular blobs, and corners. It is important to note that we are only able to visualize this space using a dimensionality reduction like t-SNE, and because of this, it is difficult to make concrete claims about the original space. Nonetheless, these visualizations hopefully provide some insight into what the descriptors are learning.

3.6 Discussion and Future Work

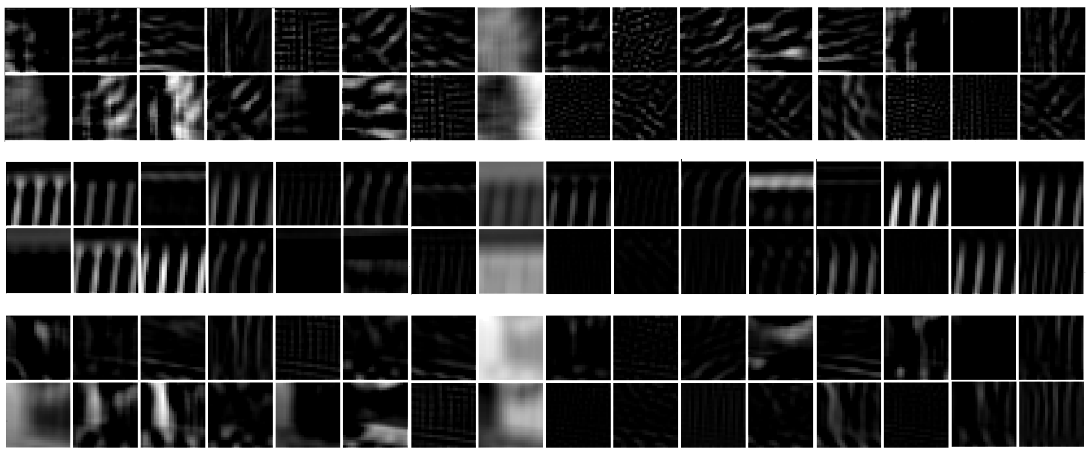
Selecting the model parameters for the feature descriptors presents a large number of design choices. This includes the number of layers, the type and dimension of each layer, the activation function, the type of pooling, etc. We feel that the two models used in this work are reasonable

Figure 3.11 CMLP convolutional and fully-connected features. The learned convolutional filters in the first layer are shown in (a). The output of the convolutional filters is shown in (c) for three sample patches, (b). These convolutional features are fed into the first fully-connected layer with 1024 outputs. In (d), the features that most strongly activate 10 of the 1024 features in the first fully-connected layer are shown—each row represents an input to the fully-connected layer.

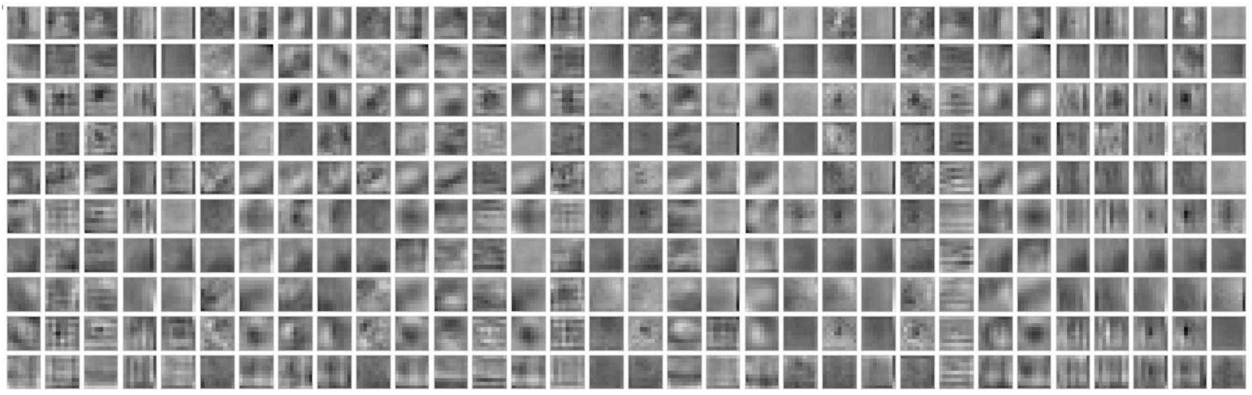


(a) CMLP Convolutional Filters

(b) Patches



(c) Sample Convolutional Features



(d) CMLP Fully-Connected Features

Figure 3.12 MLP feature space visualized with t-SNE.

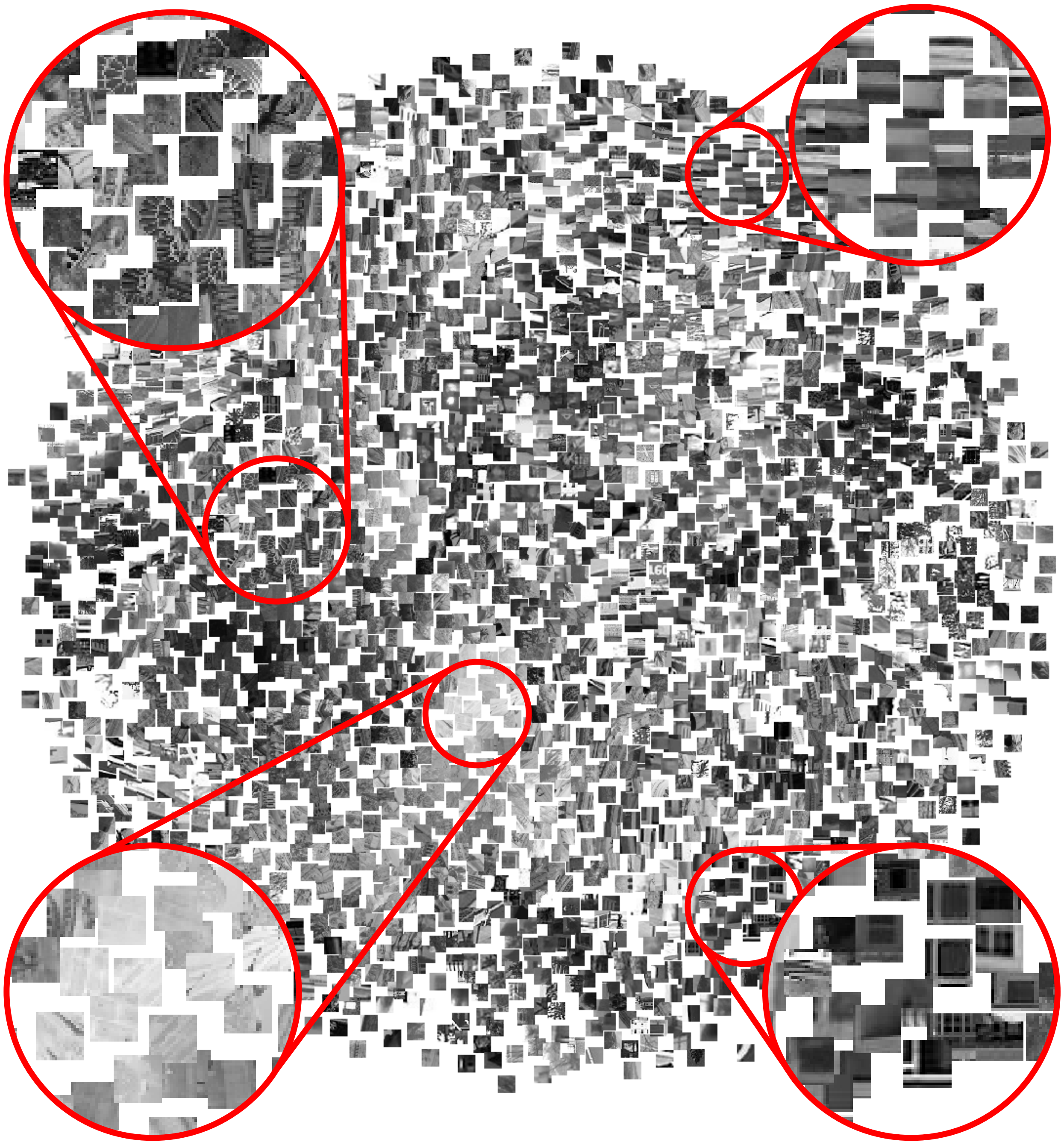
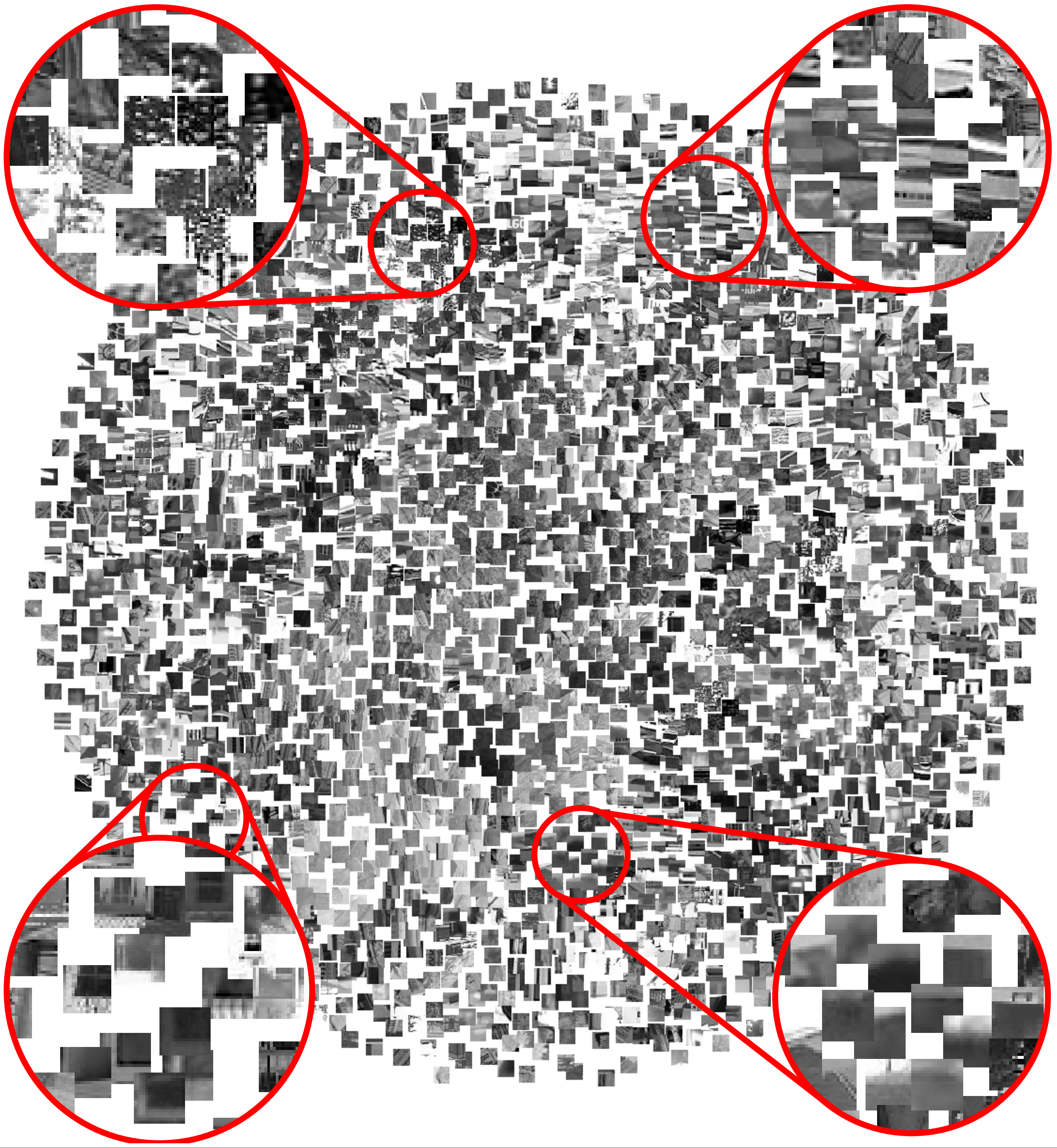


Figure 3.13 CMLP feature space visualized with t-SNE.



and good representatives of two points in the configuration space. However, many of the other model variations considered during development produced very similar results—a more thorough evaluation of the model choices with respect to performance and complexity would be beneficial.

Additionally, we would like to more thoroughly evaluate some of the other algorithm design choices, including the effect of output dimension and the effect of the smooth similarity measure and its time constant.

Beyond lighting invariance, we believe that a similar training scheme could be used in many applications to learn domain specific features. Specifically, we plan to apply the method to underwater imagery in future work.

Finally, it would be beneficial to compare the learned descriptors against additional existing feature descriptors beyond SIFT, SURF, and DAISY.

3.7 Chapter Summary

In this chapter, we have presented a method to learn visual feature point descriptors that are more robust to changes in scene lighting than standard hand-designed features. We demonstrated that, by tracking feature points in time-lapse videos, one can generate training data that captures how the visual appearance of interest points changes with lighting over time. With this training data we learned feature descriptors that map the image patches associated with feature points to a lower-dimensional feature space where Euclidean distance provides good discrimination between matching and non-matching image patches. The learned features provided better image registration performance on a challenging robotic dataset than hand-designed features including SIFT and SURF.

CHAPTER IV

Long-Term SLAM in Dynamic Environments

In this chapter, we leverage the generic linear constraint (GLC) sparse-approximate node removal method developed in Chapter II and the learned visual feature descriptors from Chapter III to develop simultaneous localization and mapping (SLAM) systems capable of long-term operation in dynamic environments. We consider two systems: one based on 3D light detection and ranging (LIDAR), and the other based on omni-directional vision.

Using the LIDAR-based system, we evaluate the use of GLC node removal as a method to control the computational complexity of long-term SLAM. We experimentally demonstrate that GLC provides a principled and flexible tool that enables a wide variety of complexity management schemes. Specifically, we consider two main classes: batch multi-session node removal, in which nodes are removed in a batch operation between mapping sessions, and online node removal, in which nodes are removed as the robot operates. The evaluation of GLC using the LIDAR-based system was originally presented in [26].

Data association is significantly more challenging in the vision-based system. We do not directly observe the 3D structure of the scene and the visual appearance of the environment changes more quickly than the 3D structure. In this chapter, we consider an exemplar-based SLAM system that seeks to address these challenges by maintaining a small set of example views at each location in the map. These example views allow the map to capture how the appearance of a location changes with time. We evaluate the use of the learned feature descriptors proposed in Chapter III, and several exemplar update schemes that use GLC to remove nodes and their associated imagery from the graph in order to select the set of example views that represents each location.

4.1 Introduction

In previous chapters, we presented a method to perform sparse-approximate node removal in SLAM factor graphs (Chapter II) and a method to learn visual feature descriptors that are more robust to changes in lighting (Chapter III). In this chapter, we explore how these tools can be used to improve long-term SLAM in dynamic environments. We consider two SLAM systems: one using 3D LIDAR and another using omni-directional vision.

As discussed in Chapter I and Chapter II, graph-based SLAM [45, 52, 86, 93, 105, 134, 160] has been used to successfully solve many challenging SLAM problems in robotics, yet it becomes computationally intractable in the long-term as new nodes must be continually added to the graph for localization. The computational complexity of the graph is dependent not only on the spatial extent of the environment, but also the *duration* of the exploration (Fig. 4.2(f)).

In this chapter, we use a LIDAR-based SLAM system to experimentally evaluate the performance of GLC when used to control the computational complexity of long-term SLAM by removing spatially redundant nodes. We demonstrate that GLC provides a principled and flexible tool enabling a wide variety of complexity management schemes. Specifically, we consider two main classes: batch multi-session node removal, in which nodes are removed in a batch operation between mapping sessions, and online node removal, in which nodes are removed as the robot operates. Using these schemes, we achieve a small and constant computational complexity with respect to time by maintaining a smaller, more-sparse graph in scenarios where the complexity of standard graph-SLAM grows super-linearly.

As discussed in Chapter I and Chapter III, visual data association is a significant challenge in dynamic environments. In this chapter, we propose and experimentally examine an exemplar-based visual SLAM system designed to address the challenges of long-term SLAM. We evaluate the use of the learned feature descriptors from Chapter III and consider several exemplar update schemes that actively maintain a small set of example views at each location in the map. These example views allow the map to capture how the appearance of a location changes with time. The exemplar updated schemes are implemented using GLC to remove unwanted nodes and their associated imagery from the map. We attempt to balance two competing objectives in the exemplar update schemes: First, we want to remove a sufficient amount of nodes such that the SLAM optimization problem remains computationally tractable. Second, we want to maintain a sufficient number of nodes and their associated imagery so that we capture the changing appearance of the environment and allow for more successful image registrations.

4.1.1 Related Work

4.1.1.1 Controlling the Computational Complexity of Long-Term SLAM

Several methods have been proposed to control the complexity of long-term SLAM including [79, 94, 97, 174]. These works, and others related to SLAM optimization complexity, are discussed in detail in §2.1.1.

4.1.1.2 Accounting for Dynamic Environments in SLAM

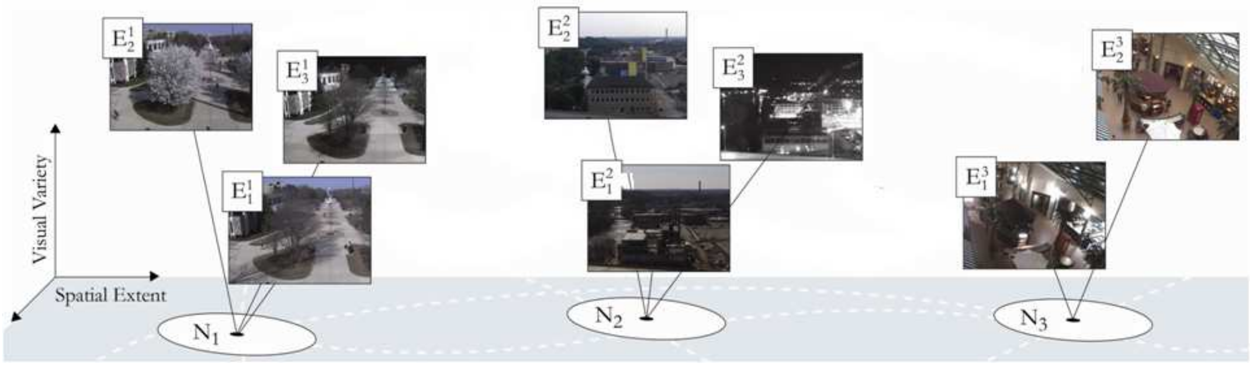
The majority of SLAM solutions, to date, make the assumption that the environment is static. For many robotic applications, this assumption is not extremely detrimental. This is especially true for systems that exploit robust data association methods, notably image and LIDAR registration—these methods are capable of successful measurements despite the presence of short-term dynamic effects, such as partial scene occlusion and moving objects.

Many methods have been proposed that try to filter out the dynamic elements of the environment while maintaining the assumption that the underlying environment is static, for example [22, 57, 67, 177], among others. It has also been shown that it is possible to explicitly identify and track dynamic objects in the environment, either for specific classes of objects, such as people [120], or *a-priori*-unknown dynamic objects [118].

Recently, several works have proposed methods that explicitly model dynamic changes in the map. Biber and Duckett [14, 15] represent the environment by a collection of sample-based maps, each of which incorporates new samples and forgets old samples at a different rate. This rate determines the timescale of each map. During localization, the robot tests all timescales to determine which best agrees with the current observation and then performs measurement updates against that map. The idea of sample-based maps is continued by Dayoub and Duckett [44], where the authors use a short-term versus long-term memory model to update the collection of visual features that represent the appearance of a location. Tipaldi et al. [164] propose a method that uses a "dynamic occupancy grid" based on a hidden Markov model to capture dynamic changes in the environment. In Johns and Yang [80, 81], locations are modeled with a collection of features observed at different points in time to better model changes in appearance. Walcott-Bryant et al. [174] propose a pose SLAM method that uses the graph's planar LIDAR scans to detect when the environment has changed. They remove the poses associated with scans that no longer represent the current state of the environment from the graph.

The works most relevant to our proposed systems can be considered "exemplar-based" methods (Fig. 4.1). Promising examples of these methods elect to represent locations in the map as a

Figure 4.1 Exemplar-based map representation. Each physical location, N_i, is represented by a set of exemplar views, E^i, that capture the change in appearance the location undergoes with time.



collection of views [94], environmental states [156], or view sequences [36], corresponding to how the environment appeared at various observation instances. These exemplar-based representations are capable of representing many different types of temporal variation. These methods provide a promising way forward toward developing SLAM systems for long-term autonomy in dynamic environments, and we therefore elect to use this model for our visual SLAM system. Stachniss and Burgard [156] proposed a method to learn exemplar configurations of an indoor environment from planar LIDAR scan data using fuzzy k-means clustering. They then use these exemplar configurations in a particle-filter based localization framework. Konolige and Bowman [94] present a vision-based method that works within the context of vision-based pose-graph SLAM [95], where the pose-graph is divided into metric neighborhoods of views bounded by physical location and view attitude. Each view is an example of how the neighborhood looked at the time it was collected. They then present a least-recently-used view deletion algorithm, which limits the number of exemplars per neighborhood to a fixed number and encourages a long-term equilibrium with the minimum set of exemplars that explains the visual variation of that neighborhood. Churchill and Newman [36] use sequences of views, termed "experiences," as the basic unit of the temporal map instead of individual views. New experiences are added to the map when the existing experiences are unable to explain the current observations.

The remainder of this section is outlined as follows: We first describe our LIDAR-based SLAM system and propose several complexity management schemes that use GLC node removal in §4.2. The LIDAR-based system is used to experimentally evaluate the effects of the repeated application of GLC node removal in §4.3. In §4.4, we present a vision-based SLAM system that uses GLC node removal to maintain an exemplar-based representation of the environment, and the learned feature descriptors to perform image registration. The visual SLAM system is then evaluated in

§4.5. Finally, §4.6 and §4.7 offer a discussion and concluding remarks.

4.2 Long-Term LIDAR-Based SLAM using GLC Node Removal

We first consider the LIDAR-based SLAM system. This system was designed for the Segway robotic platform that collected the North Campus dataset (Appendix A). This system performs pose SLAM (Fig. 1.1(c)) using the Velodyne HDL-32E LIDAR (Appendix A) as the primary sensing modality.

The Velodyne has 32 lasers mounted on a platform that spins about its vertical axis at 10 Hz to provide a full 360 degree azimuthal field of view. Each pose in the graph is associated with a sparse 3D point cloud around the robot corresponding to one revolution of the Velodyne. We derive relative constraints between poses by registering two scans using Normal Distributions Transform scan matching [106]. This allows one to directly observe the full 3D rigid-body transformation between two poses.

Odometry is estimated with an extended Kalman filter (EKF) that uses a differential-drive process model and integrates measurements from the Segway's wheel encoders, a commodity inertial measurement unit that observes roll and pitch, and a single-axis fiber optic gyro that observes change in heading. The odometry model is described in detail in Appendix A.5. In order to produce relative odometry constraints between poses, the EKF tracks the current pose of the robot and the pose of the last node added to the graph in a delayed-state framework [52]. When we wish to add a new node to the graph, we can compute the relative transform from the last node to the current robot pose. Because the delayed-state EKF tracks the correlation between the current robot pose and the last node added to the graph, we can also compute the uncertainty of the odometry constraint as described by Smith et al. [152]. We then marginalize out the old node from the delayed-state filter, augment the state with the new node, and continue to track the current pose of the robot. In our experiments, new nodes are added to the graph whenever odometry indicates that the robot has moved more than 3 m.

When available, measurements from a consumer-grade GPS are also added as xy prior factors in the SLAM graph. Note that GPS is unavailable for the indoor portions of the robot's trajectory. Additionally, GPS is available but suffers from a restricted view of the sky and possible multi-path effects (Fig. A.3) through large portions of the trajectory near tall buildings.

LIDAR loop-closures are proposed based on the current SLAM estimate. First, a rough set of loop-closure candidates are proposed based on the mean of the state estimate. For each of these candidates, the joint marginal covariance of the current pose and the candidate pose is recovered.

Using this covariance we estimate the probability that a node lies within the basin of convergence of the scan matching algorithm (approximately $6\ \mathrm{m}$). If this probability is sufficiently high ($>$ 25% in our experiments), we consider this node as a valid candidate. We only try to register a maximum of three loop-closures for each new node to limit the computational resources spent on scan matching. If we have more than three valid candidates (very common in areas that have been visited multiple times), we prioritize the candidates based on the expected information gain of a successful measurement as proposed by Ila et al. [75].

To prevent the inclusion of outlier loop-closures and GPS measurements, we gate new measurements based on the Mahalanobis distance between the measurement and the current estimate of the state. We found this to be sufficient and did not use robust-optimization methods [2, 133, 158].

4.2.1 Node Removal Schemes for LIDAR-Based SLAM

Having described our LIDAR-based SLAM system, we now propose four GLC-based graph management schemes: two that are performed as a batch step between each mapping session and two that remove nodes as the robot moves through the environment. In each case, we attempt to produce a graph that has a complexity dictated primarily by spatial extent and not by mapping duration. Therefore, we seek to remove spatially redundant nodes. The definition of spatially redundant nodes varies depending on the motion constraints of the robot and its sensing modalities. We only consider translation in the ground plane as our system is designed for a ground robot with 3D LIDAR that has an unobstructed 360 degree view of the environment. Additionally, we only seek to keep one example view per location as the 3D structure of the environment changes infrequently. Therefore, a node is considered redundant if any other node is within $3\ \mathrm{m}$. This will be relaxed when we consider the vision-based system in §4.4 to allow for multiple views per location.

In the following sections, we will refer to each of the experimental complexity schemes using the abbreviated names in Table 4.1.

Table 4.1 Experimental LIDAR SLAM Complexity Management Schemes

Scheme	Description
Batch-MR	Batch - Keep most recent (Alg. 1)
Batch-ND	Batch - Keep highest node degree (Alg. 2)
Online-RPG	Online - Emulate reduced pose graph (Alg. 3)
Online-MR	Online - Keep most recent (Alg. 4)

Algorithm 1 Batch Node Removal: Keep most recent

```
1: Given nodes in graph, $nodes = \{n_0 \dots n_m\}$
2: $nodes = \texttt{sort\_by\_time\_descending}(nodes)$
3: $keep = \{n_0\}$
4: for all $n_i$ in $nodes$ do
5:     if $\texttt{is\_not\_spatially\_redundant}(n_i, keep)$ then
6:         $keep = keep \cup n_i$
7:     end if
8: end for
9: $GLC\_remove(nodes \setminus keep)$
```

4.2.2 Batch Multi-Session Node Removal

We first consider a multi-session scenario where the robot repeatedly performs SLAM in discrete sessions. Under these conditions, node removal can be performed between sessions as a batch operation. In oversampled regions, we seek to keep the most recently added nodes as this allows the map to adapt to the changing environment; for example, there were several locations undergoing construction during our data collection. As we pass through these regions multiple times, old nodes that are spatially redundant should be removed, while new nodes capturing the changing structure should be kept. Essentially, the map can march forward in time, replacing old observations with new ones. In practice, this is performed by sorting the nodes according to their instantiation time, and then looping through the nodes in order, keeping each node that is "sufficiently far" from all other nodes currently being kept. This is detailed in Algorithm 1.

This strategy, however, only considers when a node was added. Depending on the application, additional information about each node may be available and may lead to different criteria. As an example, we consider that some nodes in the graph may be more useful for registration than others. Nodes that have been successfully registered against many times may be good candidates to keep in the graph; first, because they occur in locations that the robot repeatedly visits and second, because the observation is such that registration is repeatedly successful. Therefore, if we first sort by node degree (the number of edges connected to a node) and then remove nodes as before, we have a strategy that encourages the retention of useful nodes, as detailed in Algorithm 2. Note that many nodes will have the same node degree and instantiation time is used as a secondary criteria in the case of a tie.

Algorithm 2 Batch Node Removal: Keep highest node degree

```
1: Given nodes in graph, $nodes = \{n_0 \ldots n_m\}$
2: $nodes =$ sort_by_node_degree_descending($nodes$)
3: $keep = \{n_0\}$
4: for all $n_i$ in $nodes$ do
5:     if is_not_spatially_redundant($n_i, keep$) then
6:         $keep = keep \cup n_i$
7:     end if
8: end for
9: $GLC\_remove(nodes \setminus keep)$
```

Algorithm 3 Online Node Removal: Emulate reduced pose graph

```
1: Given previous nodes in graph, $nodes = \{n_0 \ldots n_m\}$
2: Given recent node $n_{m+1}$
3: $neighbors =$ get_redundant_neighbors($n_{m+1}, nodes$)
4: if $neighbors \neq \varnothing$ then
5:     $GLC\_remove(n_{m+1})$
6: end if
```

4.2.3 Online Node Removal

The first online node removal scheme we consider emulates the complexity reduction scheme proposed in [79], but uses GLC instead of measurement composition. In this method, referred to as the "Reduced Pose Graph," a new node is not added when the current pose is spatially redundant. Instead, the measurements from the current pose are used to add constraints between existing nodes. Conceptually, this can be thought of as temporarily adding the current node to the graph, adding its measurements and then marginalizing out the current node. This practice will add constraints between the existing nodes without permanently adding a new redundant node. We can, therefore, easily emulate this strategy by performing GLC node removal to remove recently-added redundant nodes, as detailed in Algorithm 3. Given a recently added node we look for its spatially redundant neighbors (i.e., nodes that are sufficiently close to the new node so as to make it redundant). If any spatially redundant neighbors are found, the recently-added node is removed.

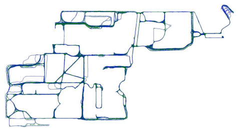
From a data association perspective, the Reduced Pose Graph formulation may not be ideal as it keeps the first sensory sample of a given location and avoids adding all subsequent observations. We therefore consider an online scheme that does exactly the opposite; instead of removing the recent node, we remove the neighbors that are made redundant by the recent node, as detailed in Algorithm 4. This is essentially an online version of Algorithm 1.

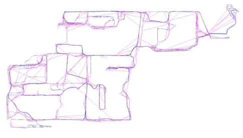
Algorithm 4 Online Node Removal: Keep most recent

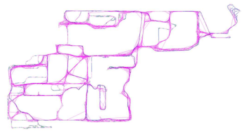
```
1: Given previous nodes in graph, $nodes = \{n_0 \dots n_m\}$
2: Given recent node $n_{m+1}$
3: $neighbors = \texttt{get\_redundant\_neighbors}(n_{m+1}, nodes)$
4: $GLC\_remove(neighbors)$
```

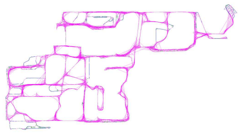
4.3 Experimental Evaluation of Long-Term LIDAR-Based SLAM

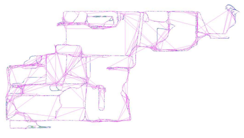
Figure 4.2 Graph comparison for LIDAR-based SLAM node removal schemes. The resulting graphs are shown after 27 mapping sessions using the proposed complexity management schemes (see Table 4.1). Links include odometry (blue), 3D LIDAR scan matching (green) and generic linear constraints (magenta). The full graph without node removal is shown as *Full*. The top row shows a top down view. The bottom row shows an oblique view scaled by time in the z-axis; each layer along the z-axis represents a mapping session.

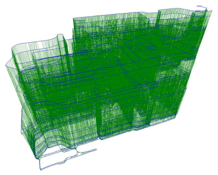

(a) *Full*
Top View

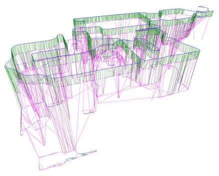

(b) *Batch-MR*
Top View

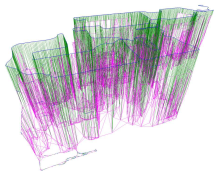

(c) *Batch-ND*
Top View

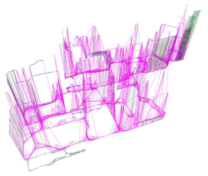

(d) *Online-RPG*
Top View

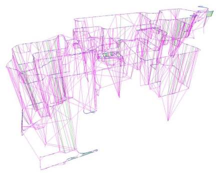

(e) *Online-MR*
Top View


(f) *Full*
Time Scaled


(g) *Batch-MR*
Time Scaled


(h) *Batch-ND*
Time Scaled


(i) *Online-RPG*
Time Scaled

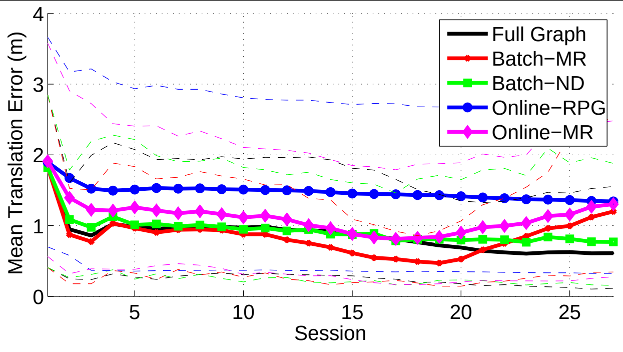

(j) *Online-MR*
Time Scaled

In order to validate the proposed LIDAR SLAM system and its associated GLC complexity management schemes, we use the North Campus dataset (Appendix A). A ground-truth graph was created from all trajectories without node removal and with the addition of constraints from a highly accurate RTK GPS system. GLC was implemented using iSAM [85, 86] as the underlying optimization engine. The code is available for download within the iSAM repository [87].

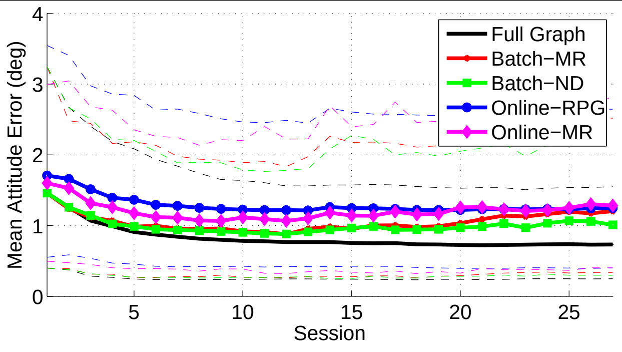
The graphs for each proposed method at the end of the last full run are shown in Fig. 4.2. In the bottom row, by scaling the z-axis according to time, we can clearly see the effects of the different node removal schemes. Using *Batch-MR*, we see that the most recent session is well-connected to the previous session with some sparse connectivity to older nodes in the graph. *Batch-ND* produces similar results but with more connectivity to previous nodes, which have been kept due to a high node degree. *Online-MR* has also removed the bulk of the nodes from previous sessions, additionally removing those from the penultimate session. In contrast, *Online-RPG* has kept its earliest observations of each location and removed newer nodes adding connectivity between older nodes.

4.3.1 Error with Respect to Ground-Truth

Figure 4.3 Error in GLC reduced graphs for LIDAR-based SLAM. Mean errors for translation ($\sqrt{\delta_x^2 + \delta_y^2 + \delta_z^2}$) and attitude ($\sqrt{\delta_r^2 + \delta_p^2 + \delta_h^2}$) are computed with respect to RTK-based ground-truth at the end of each mapping session for batch and online node removal methods. 5% and 95% percentile bounds are denoted with dashed lines. Errors for the node removal methods are compared against the errors for the full graph from which no nodes were removed (black).



(a) Translation Error



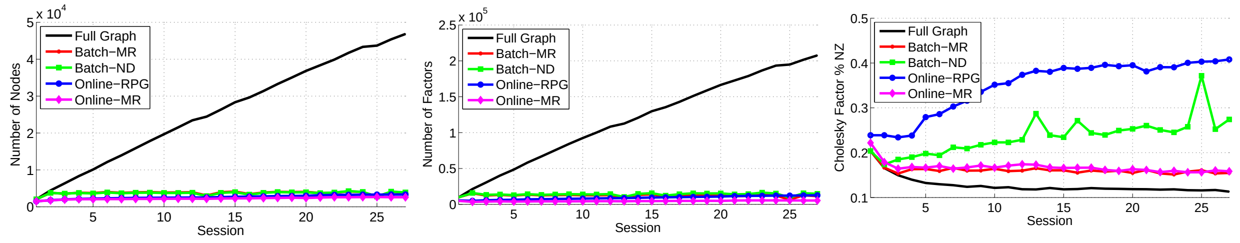
(b) Attitude Error

First, we consider the performance of each complexity management scheme in terms of translation and attitude error from the ground-truth. We include a full graph that was built without node removal as a baseline (Fig. 4.3). Note that this full graph does not include the RTK GPS and therefore is not the same as the ground-truth graph. We see all methods produce estimates with error similar to the full graph, with the online methods having slightly higher error than the batch methods in general. It is worth noting that by the end of each session, before batch sparsification, the batch methods will have almost double the number of nodes as the online methods, potentially allowing for more informative loop-closures (Fig. 4.4(a)). The *Online-RPG* method produces the highest error. This is most likely due to the fact that data association becomes more difficult as the environment changes with time, as described in §4.2.3.

4.3.2 Computational Complexity

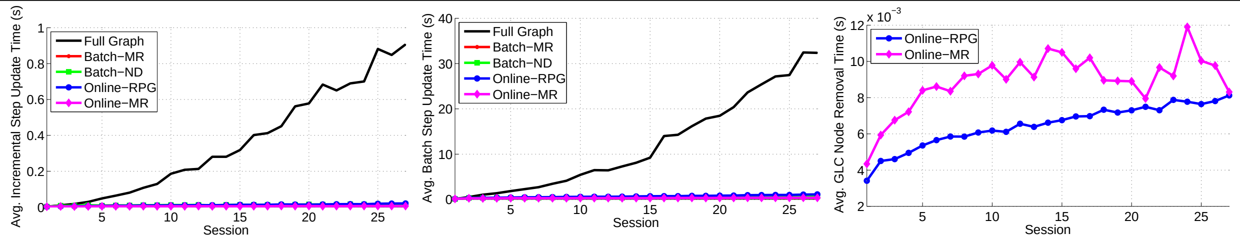
Though the graphs produced with GLC node removal have a similar or slightly higher error than the full graph, they are vastly less computationally complex. We see in Fig. 4.4 that all schemes limit the number of nodes and factors to be essentially constant as only small additions

Figure 4.4 Graph complexity for LIDAR-based SLAM. Note that for batch methods the complexity statistics are recorded at the end of each session immediately before node removal.



(a) Number of Nodes (b) Number of Factors (c) Graph Sparsity

Figure 4.5 Graph optimization time for LIDAR-based SLAM. The mean CPU time for incremental and batch iSAM optimization update steps, and for GLC node removal, is plotted in seconds.



(a) Incremental Updates (b) Batch Updates (c) GLC Node Removal

to the spatial extent of the map are made after the first session, with no method exceeding 4,000 nodes or 15,000 factors. In comparison, the full graph grows linearly ending with over 46,000 nodes and 200,000 factors. We also see that the sparsity of the measurement Cholesky Factor, R, is nearly constant, with *Online-RPG* growing the fastest as new connectivity is added between old nodes when newer nodes are removed. Note, however, that even for *Online-RPG* the maximum fill in is 0.4%, which is still quite sparse.

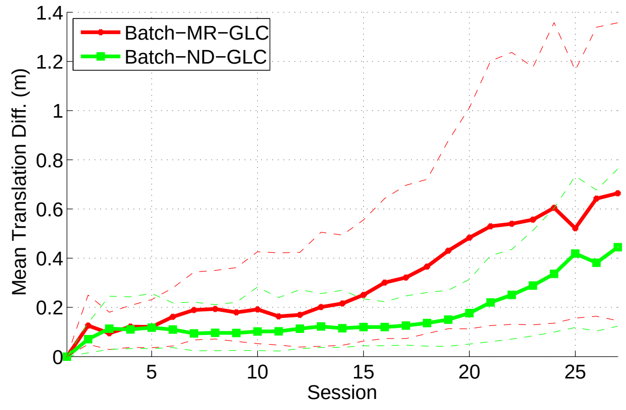
As new nodes and factors are added to the graph, iSAM performs two different types of updates: an incremental update, where the solution is updated without relinearization, and a batch update, where the solution is repeatedly relinearized and solved until convergence. In our experiments, the batch optimization update was called every 50 incremental updates. In Fig. 4.5, we see that in the full graph, the computation time for incremental and batch update steps grows super-linearly, while the proposed methods remain roughly constant (Fig. 4.5(a) and 4.5(b)). The time to remove a node using GLC is also relatively constant on the order of $10\ \mathrm{ms}$ (Fig. 4.5(c))—though slightly higher in the case of *Online-RPG* due to the higher connectivity density.

It is important to note that not all methods perform the same number of incremental and batch update steps. iSAM requires a batch optimization step after node removal and it is desirable to be as close to the optimal as possible before creating new GLC constraints. Therefore, in the *Online-RPG* and *Online-MR* schemes, we wait until 10 nodes have been flagged for removal, and then perform a batch-relinearization optimization step immediately before and after removing them. This results in the batch optimization step being called more often for the *Online-RPG* and *Online-MR* schemes. The total processing time for the $34.9\ \mathrm{h}$ of logged data, including graph optimization, node removal, data association and scan matching, took $58.7\ \mathrm{h}$ for the full graph. When using the proposed complexity management schemes, total processing times were reduced to $6.1\ \mathrm{h}$ for *Batch-MR*, $6.3\ \mathrm{h}$ for *Batch-ND*, $7.9\ \mathrm{h}$ for *Online-RPG*, and $6.8\ \mathrm{h}$ for *Online-MR*, which is at least 4.4 times faster than real-time and 7.4 times faster then the full-graph optimization.

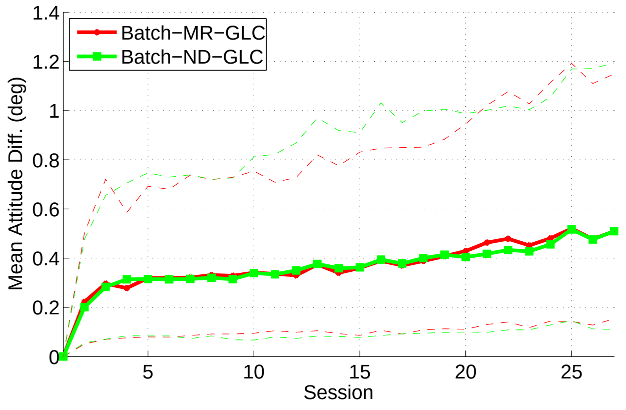
4.3.3 Distribution Comparison

In the previous experiment, each complexity management scheme elects to remove a different set of nodes, and therefore, the robot will make different data association decisions, resulting in fundamentally different graphs. In order to isolate the effects of GLC, we wish to directly compare the distribution produced by repeatedly applying sparse-approximate GLC node removal to a full distribution derived using the exact same measurements, from which no nodes have been removed. This can be done for the batch methods by accumulating the measurements from each session into one large graph. The results of this comparison are shown in Fig. 4.6. Here, we see that repeatedly applying sparse approximate GLC node removal will produce a difference in the estimates from the full graph, though the difference remains low, both in terms of mean (Fig. 4.6(a) and 4.6(b)) and Kullback-Leibler divergence (KLD) (Fig. 4.6(c)). In Fig. 4.6(d), we consider the eigenvalues of the difference between the marginal covariances of the GLC-derived and the full distribution, $\mathrm{eig}(\Sigma_{ii}^{\mathrm{GLC}} - \Sigma_{ii}^{\mathrm{FULL}})$. In the ideal case the eigenvalues of this difference will be zero, indicating perfect agreement between GLC and the true marginalization. Values larger than zero indicate conservative estimates while those less than zero indicate over-confidence. We can see that the reduced graph is generally conservative (positive values). However, the minimum of this range is slightly negative, indicating that, in some dimensions, the reduced graph is not perfectly conservative. Guaranteeing a conservative approximation, while still producing a low KLD, is possible using the techniques proposed in §2.5. However, we do not consider this small overconfidence significant enough to warrant the use of these techniques for this dataset, and therefore, elect to use the Chow-Liu tree (CLT) sparse approximation.

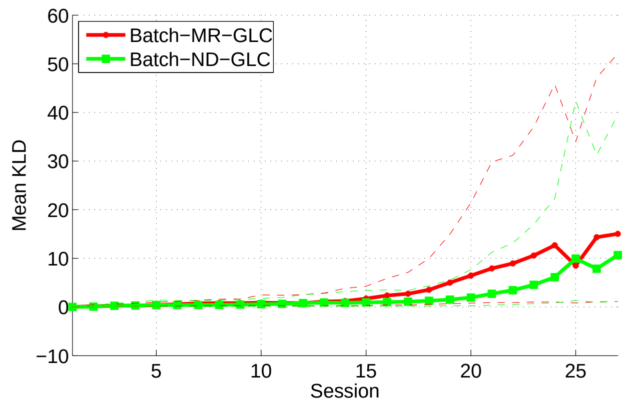
Figure 4.6 LIDAR-based SLAM distribution comparison. The estimated distributions using batch methods are compared with the estimated distributions using the same measurements but without node removal. Translation error (a) is defined as $\sqrt{\delta_x^2 + \delta_y^2 + \delta_z^2}$ and attitude error (b) as $\sqrt{\delta_r^2 + \delta_p^2 + \delta_h^2}$. The average KLD between resulting marginal covariances for each node are shown in (c). By looking at the range of the eigenvalues of the difference between the covariances of the GLC-derived marginals and full graph's marginals, $\mathrm{eig}(\Sigma_{ii}^{\mathrm{GLC}} - \Sigma_{ii}^{\mathrm{FULL}})$, (d) we can see that the reduced graph is mostly conservative (positive values). 5% and 95% percentile bounds are denoted with dashed lines.



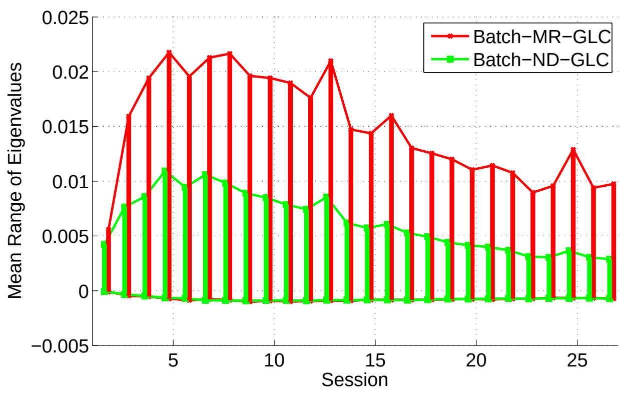
(a) Translation Error



(b) Attitude Error



(c) KL Divergence



(d) $\mathrm{eig}(\Sigma_{ii}^{\mathrm{GLC}} - \Sigma_{ii}^{\mathrm{FULL}})$

4.4 Long-Term Visual SLAM in Dynamic Environments

We now consider the vision-based SLAM system. Even though it is possible to perform landmark SLAM (Fig. 1.1(a)) with features derived from imagery, we elect to perform pose SLAM (Fig. 1.1(c)) to avoid having to estimate the millions of 3D feature points that would exist in an environment the size of the North Campus dataset.

On the Segway platform, imagery is collected using a Ladybug3 omni-directional camera system (Appendix A). This system uses six cameras to collect imagery in a hemisphere around the vehicle. Each pose in the graph is associated with five monocular images from the cameras that image in a horizontal ring around the robot. We ignore the upward facing camera because it usually captures the sky and is often overexposed. Given two poses and their associated images, we derive relative constraints between the poses by registering pairs of monocular images.

To generate visual loop-closures, we start with a set of candidate poses proposed based on the current SLAM estimate as in §4.2. Each candidate pose has five images that may match with the five images associated with the current pose. Features are extracted from each image using the convolutional multi-layer perceptron (CMLP) learned feature descriptor proposed in Chapter III. We then seek to establish feature correspondence between the two sets of images. One could directly search for correspondences between the two sets of images by lumping all features together and simply searching for the best matches between the two aggregated feature sets. Unfortunately, aggregating all the features together can make matching significantly more difficult. To avoid ambiguous matches, it is common to use the second-nearest-neighbor test [104], which ensures that a pair of features match only if the match is significantly better than any other possible match. In a very large set of features, it is more likely that a feature will be close to more than one other feature and will not be matched due to failing the second-nearest-neighbor test. However, if we can reduce the set of possible matches, we can increase our confidence in weaker matches [23, 54]. To do so, we use the heading from the current SLAM estimate to predict which pairs of images overlap and then limit the correspondence search between pairs of images. In our experiments, we only attempt to match images with approximately 60% overlap or better. A similar approach was presented by Pandey et al. [138].

Given the feature correspondences between two images, we use the standard two-view registration techniques outlined in §1.1.2.2 to produce constraints between the two poses. Note that the constraints derived from monocular imagery cannot observe the scale of the relative transform and only constrain five of the six degree of freedoms (DOFs). Observations of scale are provided by the graph's odometry backbone and GPS when it is available.

We note that many SLAM systems elect to use a place recognition method to propose loop-closure (see §1.2.5). Even though our experiments do not use place recognition, there is nothing in the system that prevents the use of place recognition as an alternative method to propose loop-closures.

Odometry and GPS constraints are computed and integrated exactly as in the LIDAR SLAM system in §4.2. Again, new nodes are added to the graph whenever the odometry indicates the robot has moved more than 3 m.

To prevent the inclusion of outlier loop-closures and GPS measurements, we gate new measurements based on the Mahalanobis distance between the measurement and the current estimate of the state. Even with this gating, we found it beneficial to apply the Dynamic Covariance Scaling M-estimator proposed by Agarwal et al. [2] to the 5-DOF constraints produced by image registration as proposed in [135].

4.4.1 Exemplar Update Scheme

Like the node removal schemes for the LIDAR SLAM system, the goal of each visual exemplar update scheme is to remove spatially redundant poses so that the size of the graph does not grow unbounded with time. However, because of the additional data association challenges specific to vision (§1.1.2.3), we relax the constraint that only one node be allowed per spatial neighborhood. To model how a place might vary in appearance with time we allow up to max_sr_nodes spatially redundant nodes. With the proposed exemplar update schemes we seek to maintain sufficient variety of example views while removing nodes from the graph so visual registration continues to be successful. For each scheme, we start with the current nodes and two user-defined limits: the maximum number of nodes allowed in the graph, max_nodes, and the maximum number of spatially redundant nodes, max_sr_nodes. By setting max_nodes, one can control the overall computational complexity of the graph. If the spatial extent of the environment is known, as it is in our experiments, one can simply set a value. Otherwise one could set max_nodes based on the area covered by the robot. Setting max_sr_nodes to a value greater than one, allows for multiple example views of the same location. In our experiments max_sr_nodes was set to 3.

It is interesting to consider the relationship between the maximum number of nodes per neighborhood, max_sr_nodes, and a total maximum number of nodes in the graph, max_nodes. If we do not enforce max_nodes, then every neighborhood will fill until max_sr_nodes is reached. However, when the max_nodes constraint is active, some neighborhoods will have fewer than max_sr_nodes nodes. It is even possible that no nodes will be kept in areas of the environment where data association is consistently poor.

Algorithm 5 Batch Node Removal: Keep most recent

```
1: Given nodes in graph, $nodes = \{n_0 \ldots n_m\}$, $max\_nodes$, and $max\_sr\_nodes$
2: $nodes =$ sort_by_time_descending($nodes$)
3: $keep = \{n_0\}$
4: for all $n_i$ in $nodes$ do
5:     if spatially_redundant_cnt($n_i, keep$) $< max\_sr\_nodes$ and len($keep$) $< max\_nodes$ then
6:         $keep = keep \cup n_i$
7:     end if
8: end for
9: $GLC\_remove(nodes \setminus keep)$
```

We elect to use batch removal between sessions as it produced the best results in the LIDAR SLAM system. As in §4.2, nodes within 3 m of each other are considered to be spatially redundant. Note that we do not consider heading because omni-directional imagery is used. We will refer to each of the experimental complexity schemes using the abbreviated names in Table 4.2.

Table 4.2 Experimental Visual SLAM Exemplar Update Schemes

Scheme	Description
Batch-MR	Batch - Keep most recent (Alg. 5)
Batch-CND	Batch - Keep camera node degree (Alg. 6)
Batch-CMR	Batch - Keep camera most recent (Alg. 7)

The first scheme is the *Batch-MR* method proposed in §4.2, which keeps the most recent nodes in oversampled regions, allowing the map to adapt to the changing environment. For use in the vision system, we relax the constraint that there be no spatially redundant nodes and allow up to max_sr_nodes nodes per location while enforcing that the total number of nodes in the graph is less than max_nodes. The updated *Batch-MR* scheme is detailed in Algorithm 5.

The *Batch-MR* scheme does not consider how useful the imagery at a node may be for data association. Some regions of the trajectory may have imagery that is consistently difficult to register. Nodes in these regions should be given low priority when selecting the nodes to keep. In the North Campus dataset, this happens frequently when the robot passes through open parking lots or areas with trees on both sides of the path. To account for this, we track each time a node is successfully used to produce a measurement. We can then sort the nodes based on the number of times they have been used in a measurement. In the case of a tie, we favor nodes that have been used in a measurement more recently to move the example views forward through time. Once the nodes are sorted, we greedily select nodes to keep until the max_sr_nodes and max_nodes constraints prevent including any more. This method is referred to as *Batch-CND* and is detailed in Algorithm 6.

Algorithm 6 Batch Node Removal: Keep camera node degree

```
1: Given nodes in graph, $nodes = \{n_0 \ldots n_m\}$, $max\_nodes$, and $max\_sr\_nodes$
2: $nodes =$ sort_by_num_cam_meas_descending($nodes$)
3: $keep = \{n_0\}$
4: for all $n_i$ in $nodes$ do
5:     if spatially_redundant_cnt($n_i, keep$) $< max\_sr\_nodes$ and len($keep$) $< max\_nodes$ then
6:         $keep = keep \cup n_i$
7:     end if
8: end for
9: $GLC\_remove(nodes \setminus keep)$
```

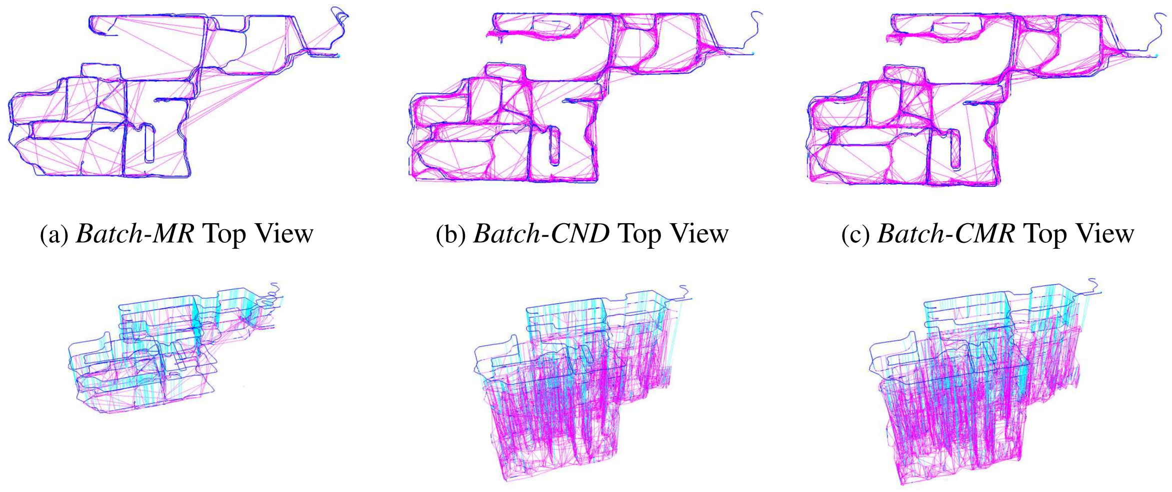
Algorithm 7 Batch Node Removal: Keep camera most recent

```
1: Given nodes in graph, $nodes = \{n_0 \ldots n_m\}$, $max\_nodes$, and $max\_sr\_nodes$
2: $nodes =$ sort_by_cam_meas_time_descending($nodes$)
3: $keep = \{n_0\}$
4: for all $n_i$ in $nodes$ do
5:     if spatially_redundant_cnt($n_i, keep$) $< max\_sr\_nodes$ and len($keep$) $< max\_nodes$ then
6:         $keep = keep \cup n_i$
7:     end if
8: end for
9: $GLC\_remove(nodes \setminus keep)$
```

This scheme is analogous to the *Batch-ND* scheme proposed for LIDAR SLAM, however, instead of just considering node degree (which could be caused by odometry, scan matching, GPS, and GLC factors), we focus specifically on the number of camera factors.

One potential pitfall of the *Batch-CND* scheme is when the environment drastically changes (e.g., due to construction or the changing of seasons) it may take a long time for well established nodes that have produced many successful measurements to be replaced—even though they are no longer valid. One option to prevent this lag is to prioritize nodes that have recently been used to produce a camera constraint. We implement this in a scheme referred to as *Batch-CMR*. In this scheme, nodes are first sorted by the time of their most-recent camera measurement. To break ties, which occur because all nodes that match a new node have the same most-recent measurement time, we fall back on the number of camera measurements to retain nodes that have been repeatedly useful. The *Batch-CMR* scheme is detailed in Algorithm 7.

Figure 4.7 Graph comparison for vision-based SLAM node removal schemes. The resulting graphs are shown after 27 mapping sessions using the proposed complexity management schemes (see Table 4.2). Links include odometry (blue), 5-DOF camera constraints (cyan) and GLCs (magenta). The top row shows a top down view. The bottom row shows an oblique view scaled by time in the z-axis; each layer along the z-axis represents a mapping session.



(a) *Batch-MR* Top View (b) *Batch-CND* Top View (c) *Batch-CMR* Top View

(d) *Batch-MR* Time Scaled (e) *Batch-CND* Time Scaled (f) *Batch-CMR* Time Scaled

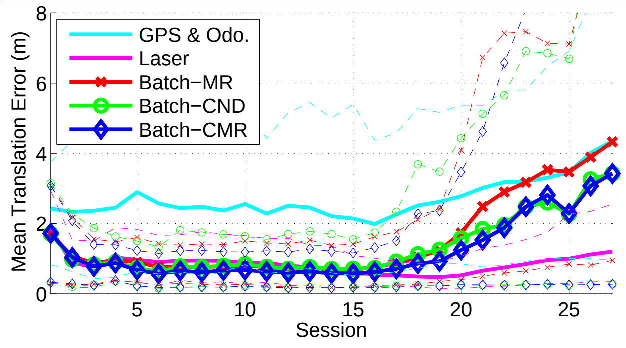
4.5 Experimental Evaluation of GLC Long-Term Visual SLAM

We first compare the proposed exemplar update schemes using the CMLP learned features. We limit the total number of nodes in the graph to $max\_nodes = 4000$ and limit the number of nodes in each neighborhood to $max\_sr\_nodes = 3$. From the evaluation of the LIDAR SLAM system, we know that a graph with 4000 nodes is computationally feasible for our system. Additionally, it takes just over 2000 nodes to sample the entire space of trajectories with a single node for each $3\ \mathrm{m}$ neighborhood. Therefore, the $max\_nodes = 4000$ constraint will limit the average number of nodes per neighborhood to approximately 2, though the distribution will vary depending on the update scheme.

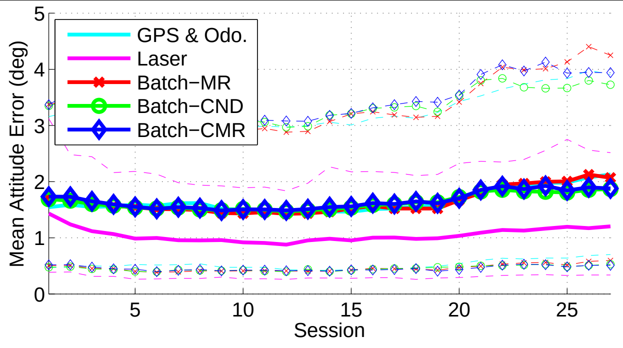
Fig. 4.7 shows the graphs for each method at the end of the last session. We see that the *Batch-MR* scheme (Fig. 4.7(a) and (d)) only has nodes from the previous two to three sessions, having removed all older nodes. Both the *Batch-CND* (Fig. 4.7(b) and (e)) and *Batch-CMR* (Fig. 4.7(c) and (f)) schemes maintain some older nodes that have proven sufficiently useful for visual registration.

We compare the error from ground-truth at the end of each session for the node removal schemes in Fig. 4.8(a) and (b). In these plots, we also show the results for a graph using only odometry and GPS constraints (to highlight the contribution of the visual constraints) and the

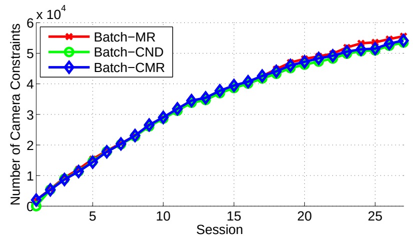
Figure 4.8 Comparison of exemplar update schemes for vision-based SLAM. In (a) and (b), we compare the mean error for translation ($\sqrt{\delta_x^2 + \delta_y^2 + \delta_z^2}$) and attitude ($\sqrt{\delta_r^2 + \delta_p^2 + \delta_h^2}$) with respect to RTK-based ground-truth at the end of each mapping session for our proposed node removal methods (5% and 95% percentile bounds are denoted with dashed lines). As a comparison, errors for a SLAM solution that ignores the visual data and only includes GPS and odometry constraints, and for the *Batch-MR* LIDAR SLAM solution are also provided. In (c), we show the number of successful image registrations that have been integrated into the graph at the end of each session. In (d) and (e), we show the average incremental and batch update time for each scheme.



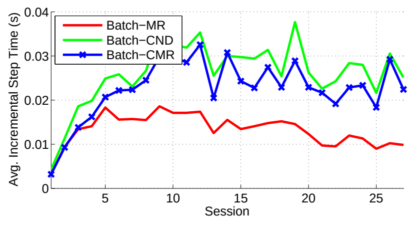
(a) Translation Error



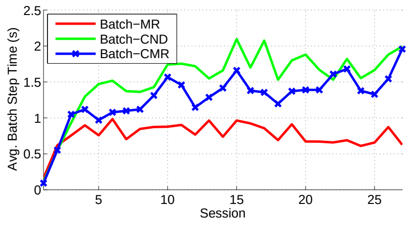
(b) Attitude Error



(c) Camera Factors



(d) Incremental Update Time

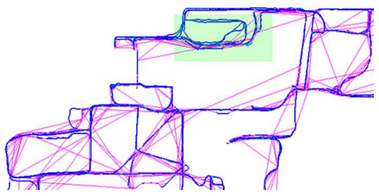


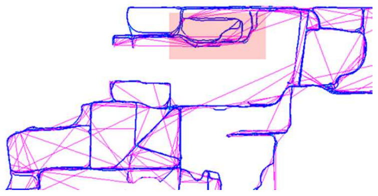
(e) Batch Update Time

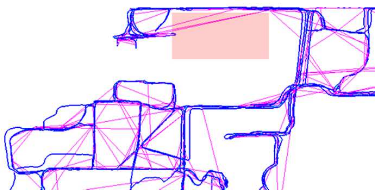
LIDAR *Batch-MR* result (as a performance benchmark). We see that all of the methods have a very low error, comparable with that of the LIDAR systems for the first 16 sessions. After this point the visual methods begin to accrue more error, with the *Batch-CND* and *Batch-CMR* slightly outperforming *Batch-MR*.

Investigating the graphs around the 16th session more closely, we see that errors increase when a large construction project restricted the trajectories on the north side of campus to a long, disconnected, feature-poor path. These changes are illustrated in Fig. 4.9 using the *Batch-MR* graph, which quickly removes nodes from areas that have not been visited recently. Prior to Session 16, the area highlighted in green in Fig. 4.9(a) was accessible and many of the sessions passed through this area. This area was closed from Session 18 onward due to a large construction project, as highlighted in red in Fig. 4.9(b). All subsequent sessions were restricted to the path along the north of the construction site (Fig. 4.9(c)). This path follows a road lined with trees and lawn. Views of more interesting structure were blocked by a tall, screened construction fence (Fig. 4.9(d)). After returning from this long and poorly-constrained path, several trajectories accepted poor loop-closures and failed to tightly integrate back into the graph. This causes an increase in error for these trajectories and all subsequent trajectories as new images register to them. This challenging feature-poor area of the environment affected all the visual methods—highlighting that challenges still remain for vision-based SLAM front ends.

Figure 4.9 Change in the route caused by construction. The change in route is shown with the *Batch-MR* graph, which quickly removes nodes from areas that have not been visited recently. Prior to Session 16, the area highlighted in green was accessible, and many of the sessions passed through this area. This area was closed from Session 18 onward, as highlighted in red. All subsequent sessions were restricted to the path along the north of the construction site. Sample imagery from this path is shown in (d).


(a) Session 16


(b) Session 18


(c) Session 22

(d) Sample Imagery

In Fig. 4.8(c) we show the number of successful camera constraints that have been integrated into the graph at the end of each session. We see that each of the methods produces roughly the same number of successful image constraints. In Fig. 4.8(d) and (e) we show the average incremental and batch update times. For all methods, these times level out as we reach the $max\_nodes = 4000$ limit. The *Batch-CND* and *Batch-CMR* schemes are noticeably more computationally expensive as they retain more of the original camera factors that have not been folded into the reduced set of GLC factors. The total processing time for the $34.9\ \mathrm{h}$ of logged data, including graph optimization, node removal, feature extraction, and image registration, took around $24\ \mathrm{h}$ for each of the methods, which is about 1.5 times faster than real-time. Note that the bulk of this time, around $16\ \mathrm{h}$, was spent on feature extraction.

In Figures 4.10, 4.11, 4.12, and 4.13 we show example imagery from four different neighborhoods. Each row represents the example views at the end of a session, up to $max\_sr\_nodes = 3$. In Fig. 4.10, we see a feature-rich indoor scene that establishes a relatively stable set of example views. Fig. 4.11 shows an outdoor scene where the example views cycle through various lighting conditions. We see variation in example views capturing changes in foliage in Fig. 4.12 and snow in Fig. 4.13.

We experimentally evaluate the effect of maintaining multiple example views per location by comparing the *Batch-CND* scheme with three example views per neighborhood and with one example view per neighborhood. The results of this comparison in terms of error and number of camera factors is shown in Fig. 4.14. Surprisingly, the algorithm finds no benefit in maintaining more than one example view, even though the visual inspection of the example views indicates that they are capturing some of the appearance variation of the neighborhoods. There are several reasons why this may be the case: First, we only propose a maximum of 20 camera factors for every new node to limit the computation cost of data association. With more example views per neighborhood, we may need to consider significantly more potential factors. Second, potential camera factors are prioritized by their expected information gain [75]. This is a good choice for the underlying optimization problem, but it does not consider the example view representation of a neighborhood. Here, place recognition, or a method specifically designed for exemplar-based maps, such as our preliminary work in [24], may be more appropriate.

Finally, we consider the effect of the learned feature descriptors on the SLAM system. To do so we use the *Batch-CND* scheme with the learned CMLP and multi-layer perceptron (MLP) features proposed in Chapter III and the commonly-used scale invariant feature transform (SIFT) feature. We see in Fig. 4.15(c) that the CMLP successfully registers significantly more links than MLP and SIFT, corroborating the results from Chapter III. However, simply making more camera constraints did not directly translate into a reduction in error, with all features having similar errors

Figure 4.10 Sample *Batch-CND* exemplars inside the CSE building. Each row represents the exemplars (up to three) for this location at the end of a session. 14 of the 27 sessions are shown.

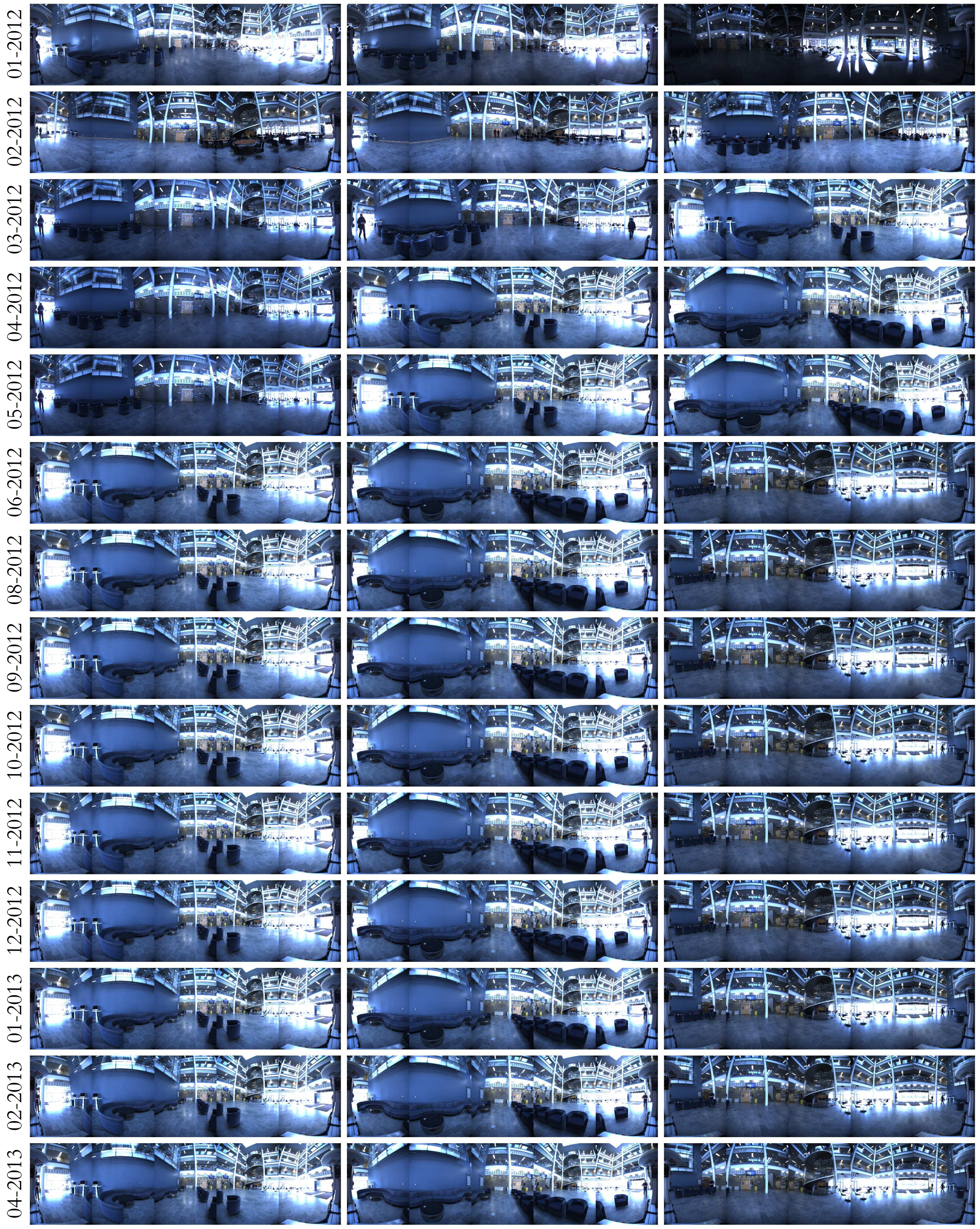


Figure 4.11 Sample *Batch-CND* exemplars outside the CSE building. Each row represents the exemplars (up to three) for this location at the end of a session. 14 of the 27 sessions are shown.

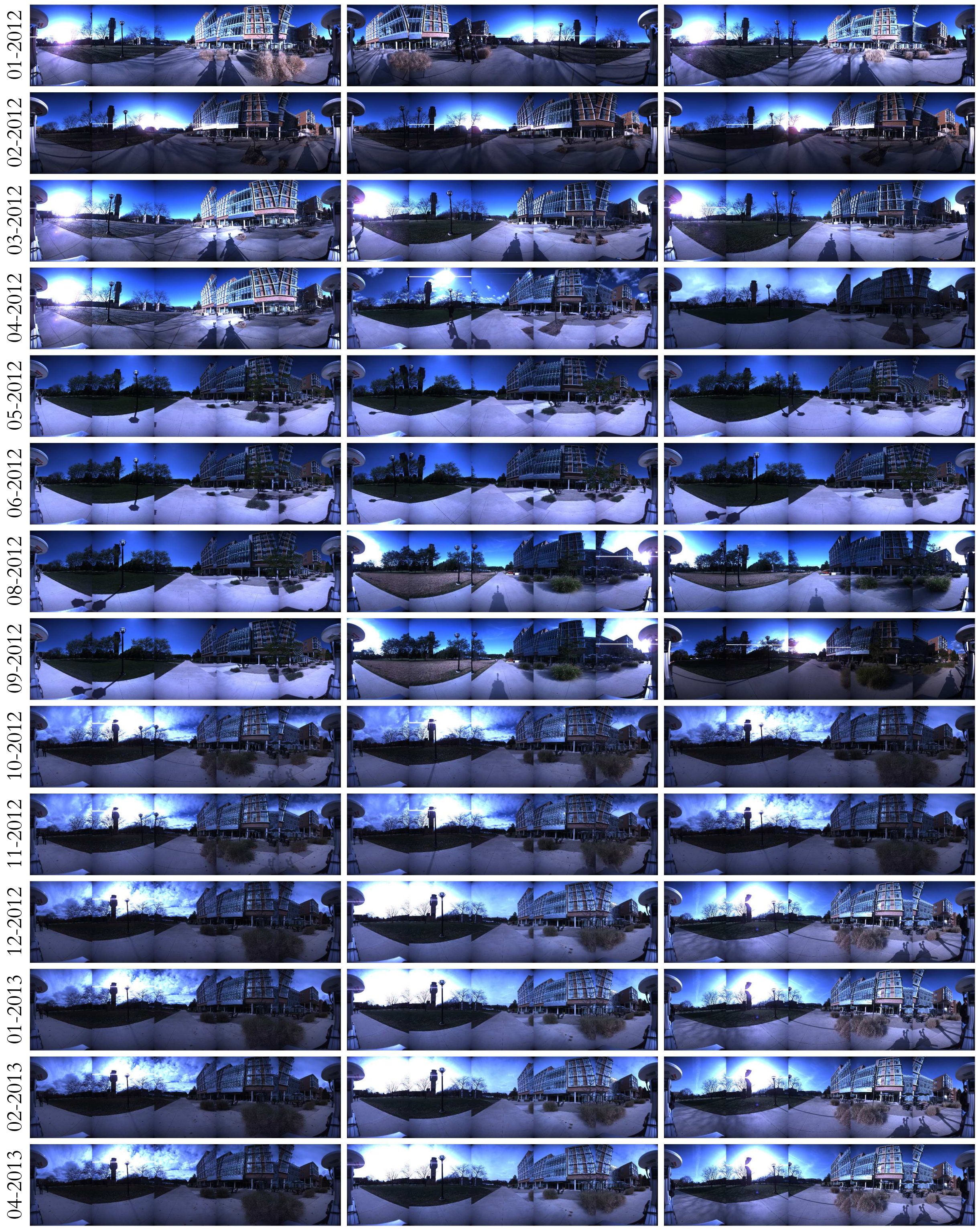


Figure 4.12 Sample *Batch-CND* exemplars outside the EECS building. Each row represents the exemplars (up to three) for this location at the end of a session. 14 of the 27 sessions are shown.

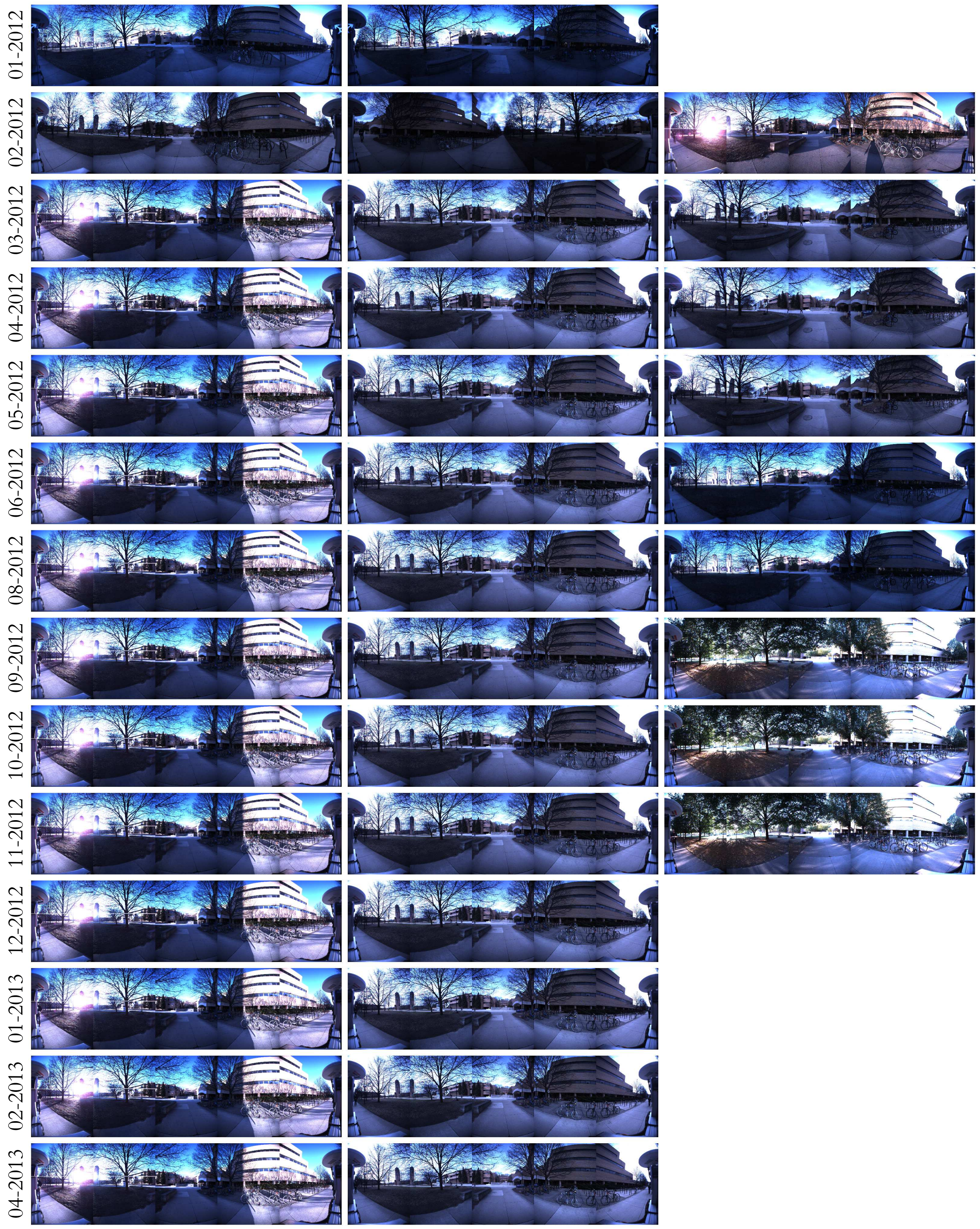


Figure 4.13 Sample *Batch-CND* exemplars outside the Duderstadt center. Each row represents the exemplars (up to three) for this location at the end of a session. 14 of the 27 sessions are shown.

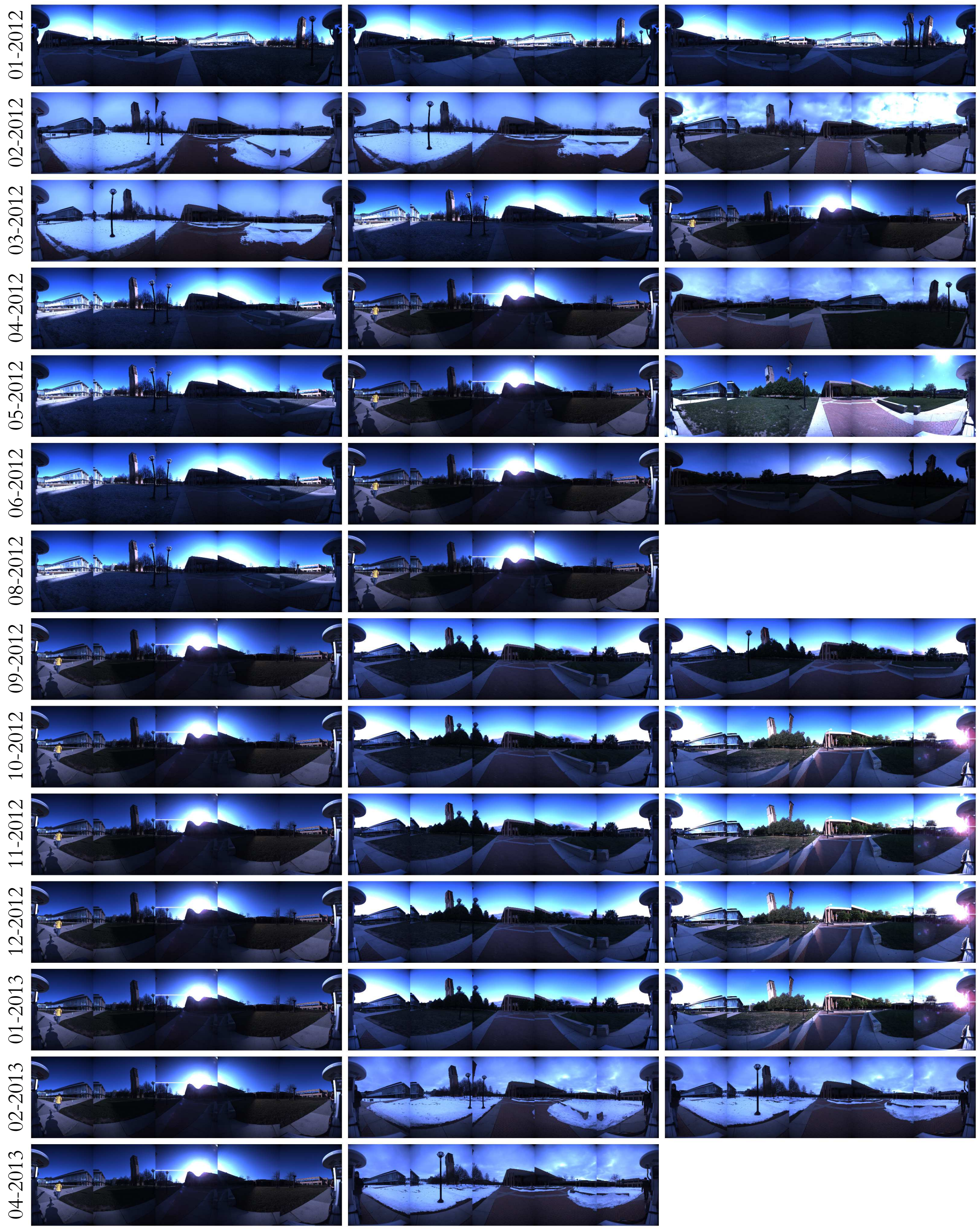
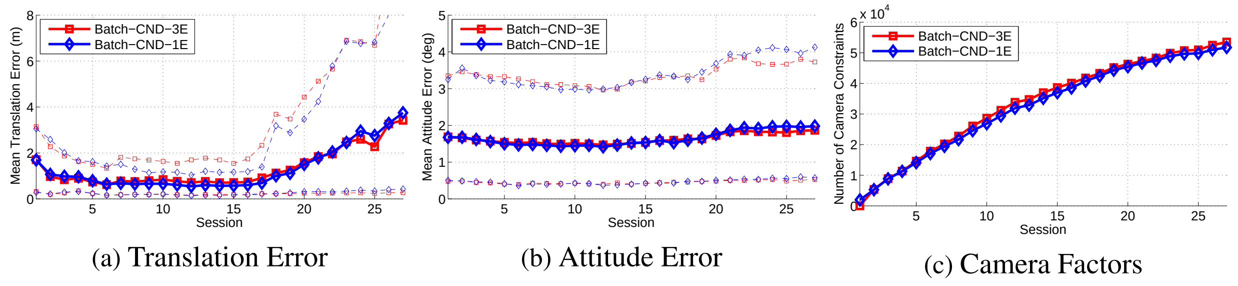
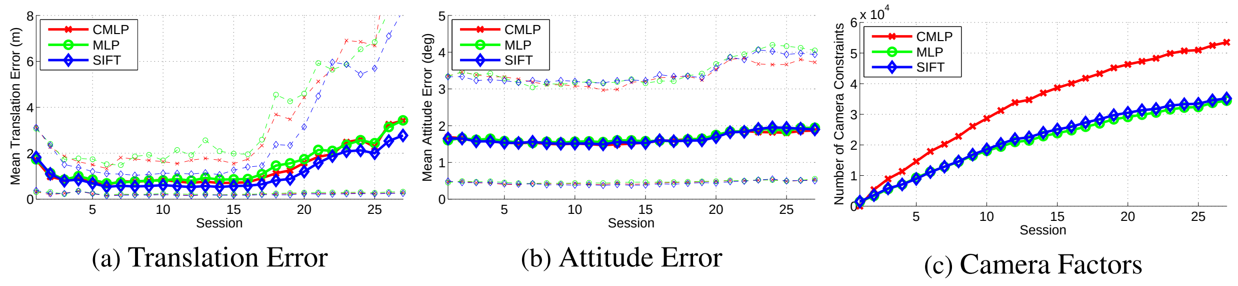


Figure 4.14 Effect of multiple exemplars for vision-based SLAM. In (a) and (b), we compare the mean error for translation ($\sqrt{\delta_x^2 + \delta_y^2 + \delta_z^2}$) and attitude ($\sqrt{\delta_r^2 + \delta_p^2 + \delta_h^2}$) with respect to RTK-based ground-truth at the end of each mapping session with a maximum of one or three example views per neighborhood. 5% and 95% percentile bounds are denoted with dashed lines. In (c), we show the number of successful image registrations that have been integrated into the graph at the end of each session.



(a) Translation Error (b) Attitude Error (c) Camera Factors

Figure 4.15 Comparison of visual feature descriptors for vision-based SLAM. In (a) and (b), we compare the mean error for translation ($\sqrt{\delta_x^2 + \delta_y^2 + \delta_z^2}$) and attitude ($\sqrt{\delta_r^2 + \delta_p^2 + \delta_h^2}$) with respect to RTK-based ground-truth at the end of each mapping session with the two learned features proposed in Chapter III and the standard SIFT feature. 5% and 95% percentile bounds are denoted with dashed lines. In (c), we show the number of successful image registrations that have been integrated into the graph at the end of each session.



(a) Translation Error (b) Attitude Error (c) Camera Factors

(Fig. 4.15(a) and (a)). We suspect that, because the Segway robot has very good odometry, yielding a strong odometry backbone, the number of links does not strongly affect the accuracy. A small number of good loop-closures spread evenly throughout the trajectory may be sufficient to correct the odometry. This may not be true on other robotic platforms with weaker odometry.

4.6 Discussion

4.6.1 Using GLC Node Removal to Control Long-Term SLAM Computational Complexity

Having compared four different complexity management schemes based on GLC node removal, we can highlight some things to consider when designing new schemes:

- Removing larger sets of nodes less often produces better results than removing small sets of nodes more often. Note that there is not a binary difference between online and batch node removal, it is just a matter of how long nodes are left in the graph before removal.

- Even though the GLC constraints are reparameterized in terms of relative transforms, they still commit to a relative linearization point. Therefore, it is desirable that the relative transforms be as close as possible to the optimal solution before node removal. The graph should be optimized as well as possible before node removal.

- When removing a set of nodes, it is important to note that the order in which they are removed affects the resulting graph connectivity. Experimentally, we found that removing long chains of nodes sequentially sometimes produced large star shaped trees in the graph, which slowed subsequent node removal. To avoid this, sets of nodes were removed in a randomized order in all experiments. The variable elimination ordering problem [92] is well-studied for dense node removal. The application and adaptation of existing variable elimination ordering strategies for node removal with sparse connectivity could further improve the performance of GLC-based complexity management schemes. Toward this, we have compared the randomized ordering used in this thesis with a greedy minimum-degree ordering. Preliminary results indicate that the minimum-degree ordering can significantly reduce the time it takes to remove nodes from the graph. However, it does not seem to impact the quality of the resulting sparse approximation.

4.6.2 Long-Term Visual SLAM

Compared to the LIDAR SLAM system, there are several areas where the exemplar-based visual SLAM system proposed in this chapter could be improved:

- The increase in error after session 16 (§4.5) indicates that there is still significant room for improvement in visual data association in unstructured environments.

- Even though the exemplar update schemes qualitatively appear to capture interesting changes in the visual appearance of the neighborhoods, this did not result in improved visual data association (§4.5). It would be interesting to further explore how the variance in appearance captured by the example views could be better exploited.

4.7 Chapter Summary

In this chapter, we demonstrated that GLC provides a principled and flexible tool that enables a variety of complexity management schemes where pairwise measurement composition would normally be used. We proposed and evaluated four complexity management schemes based on GLC node removal. Each method is shown to successfully solve a large-scale long-term SLAM problem while greatly reducing the associated computational complexity. Additionally, we proposed and evaluated a visual, exemplar-based SLAM using learned visual feature descriptors and GLC node removal.

CHAPTER V

Conclusion

Extending the capabilities of simultaneous localization and mapping (SLAM) systems operating for long-term periods of time in dynamic environment requires addressing two core problems: First, the computational complexity of the SLAM optimization problem must not grow unbounded with time. Many state-of-the-art graph SLAM formulations require that nodes be continuously added to the graph for the robot to stay localized and preserve problem sparsity. Second, the SLAM front-end must continue to function as the environment changes with time. While certain short-term and small-scale dynamic changes can be considered as noise within the SLAM front-end, truly long-term SLAM requires a front-end that explicitly accounts for dynamic changes in the environment. This is especially true for vision-based SLAM where even the change in lighting between morning and evening may be enough to break state-of-the-art systems. In this thesis, we have produced the following contributions that seek to address these challenges:

- We have collected a challenging dataset on the University of Michigan's North Campus appropriate for the evaluation of long-term SLAM systems (Appendix A). Using our Segway robotic platform, we collected imagery and light detection and ranging (LIDAR) data from January 2012 to April 2013. In addition to allowing us to throughly evaluate the proposed algorithms throughout this thesis, we plan to release this dataset to the community.

- In Chapter II, we proposed a factor-based method for node removal in graph SLAM that addresses the shortcomings of measurement composition. The proposed method, which we refer to as generic linear constraint (GLC), is able to produce a new set of constraints over the marginalization clique that can represent either the true marginalization, or a sparse approximation of the true marginalization. We experimentally demonstrate that GLC can be used to provide an accurate approximation of marginalization when removing a large number of nodes from a SLAM graph. We also present several methods that ensure a conservative approximation of the true marginalization.

- In Chapter III, we presented a method to learn visual feature point descriptors that are more robust to changes in scene lighting than standard hand-designed features. We demonstrated that, by tracking feature points in time-lapse videos, one can easily generate training data that captures how the visual appearance of interest points changes with lighting over time. This training data was used to learn feature descriptors that map the image patches associated with feature points to a lower-dimensional feature space where $\mathcal{L}_2$ distance provides good discrimination between matching and non-matching image patches. We showed that our learned feature descriptors outperformed standard hand-designed features on imagery from the North Campus dataset.

- In Chapter IV, we proposed LIDAR- and vision-based SLAM systems capable of long-term operation in dynamic environments. These systems leveraged the proposed GLC node removal to control the computational complexity of the graph over time, and to actively preserve a set of example views for each location. Using the LIDAR-based system, we experimentally demonstrated that GLC node removal can be used to control the computational complexity of long-term SLAM. We proposed an exemplar-based SLAM system that seeks to address the challenges of long-term visual SLAM. We evaluated the use of the learned feature descriptors from Chapter III, and several exemplar update schemes that actively seek to maintain a small set of example views at each location in the map that capture how the appearance of a location changes with time.

5.1 Future Directions

The methods and results developed in this thesis motivate several areas of future research.

5.1.1 Improving GLC Node Removal

Considering the GLC node removal method presented in Chapter II, there are several potential avenues for future work:

- We have considered two classes of GLC node removal, fully-dense and a sparse tree approximation. These classes represent the opposite ends of a spectrum of connectivity structures.

Mazuran et al. [108] have shown that the addition of even a few additional edges can substantially reduce the Kullback-Leibler divergence (KLD) of the approximation while still maintaining sufficient sparsity. Extending GLC to allow for a variety of sparse approximation structures is a promising avenue for improving its performance.

- GLC provides the best performance when nodes are well-constrained and near their true optimal solution before they are removed. Committing to a linearization point, even if it is strictly relative, can be detrimental if the linearization point is far from the optimal. It would be beneficial to predict, based upon the graph, when nodes are sufficiently well-constrained and can be removed accurately.

- When considering the application of sparse-approximate GLC, one must currently decide if the Chow-Liu tree (CLT) or a guaranteed-conservative variant is most appropriate. We have found experimentally that, in some cases, the guaranteed-conservative methods perform very well and should be used in place of the CLT. In other cases, there is a significant increase in the KLD when using the conservative methods, and one must choose, based on their application requirements, if a low KLD is more important than the guarantee of a conservative estimate. We currently have no method to predict when the guaranteed conservative methods will perform well and rely on experimental evaluation before making a decision. A better understanding of the conditions where the conservative methods perform well would be useful in the practical application of GLC.

- As mentioned in Chapter II, when removing a set of nodes it is important to note that the order in which they are removed affects the resulting graph connectivity. To avoid removing long chains of nodes sequentially, we instead removed sets of nodes in a randomized order in all experiments. The variable elimination ordering problem [92] is well-studied for dense node removal. The application and adaptation of existing variable elimination ordering strategies for node removal with sparse connectivity could further improve the performance of GLC-based complexity management schemes. Toward this, we have compared the randomized ordering used in this thesis with a greedy minimum-degree ordering. Preliminary results indicate that the minimum-degree ordering can significantly reduce the time it takes to remove nodes from the graph. However, it does not seem to impact the quality of the resulting sparse approximation.

5.1.2 Long-Term Visual Data Association

Long-term visual data association still faces many challenges if we hope to achieve results that are truly competitive with specialized sensors like LIDAR. Vision-based SLAM systems can struggle with natural, unstructured environments as described in §4.5. Additionally, there will always be a trade-off between the discriminative power of local features and their robustness to changes in visual appearance. We can try to improve local descriptors so that they are robust to some changes, as in Chapter III, while still being sufficiently discriminative. However, it seems unlikely that in very challenging scenarios the local appearance around key-points in an image will be sufficiently consistent or discriminative to establish correspondence and perform geometric registration. Recent works in whole-image place recognition, such as [38, 113, 126, 132, 151], have shown very promising results on datasets with large changes in visual appearances. However, if geometric image registration is desired, whole-image place recognition is not adequate by itself. Methods that allow for both robust place recognition and robust geometric image registration using a mid-level or hierarchical representation would be a very interesting area of future research. Additionally, leveraging a more semantic understanding of the environment could lead to improved image registration in dynamic environments.

5.1.3 Systems for Long-Term Mapping and Navigation

In most, if not all, long-term mapping and navigation applications, it will not make sense to perform SLAM indefinitely. Adding new measurements to the graph provides diminishing returns, and at some point adding further measurements will not result in a significantly better map. This motivates a system that performs SLAM for a period of time and then switches to localization. Designing such a system raises many questions: How can we determine when further measurements are no longer important? How do we account for the fact that the graph may be well constrained in some areas but not others? How can we detect that a portion of a map has changed and "rebuild" that area? It would be interesting to address these questions with systems that smoothly alternate between localization and SLAM to build, utilize, and update a map as needed.

APPENDIX

APPENDIX A

North Campus Long-Term Dataset

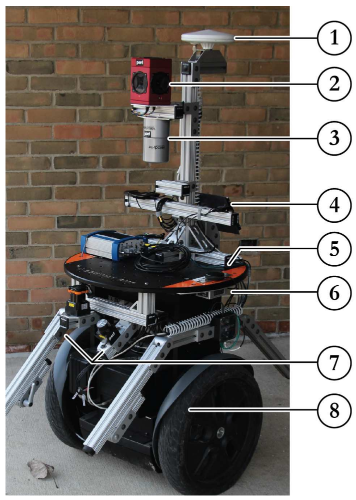
In order to validate the algorithms proposed in this thesis, we have collected a challenging long-term dataset on the University of Michigan's North Campus. This dataset will be used for evaluation throughout this thesis. The North Campus long-term dataset consists of data collected by a Segway robotic platform, Fig. A.1(a), approximately biweekly, between January 8th, 2012 and April 5th, 2013, on the University of Michigan's North Campus. The Segway is outfitted with a Ladybug3 omni-directional camera, a Velodyne HDL-32 3D LIDAR, two Hokuyo planar LIDARs, an inertial measurement unit (IMU), a single-axis fiber optic gyro (FOG), consumer grade GPS, and a RTK GPS for ground-truth.

A.1 Data Collection

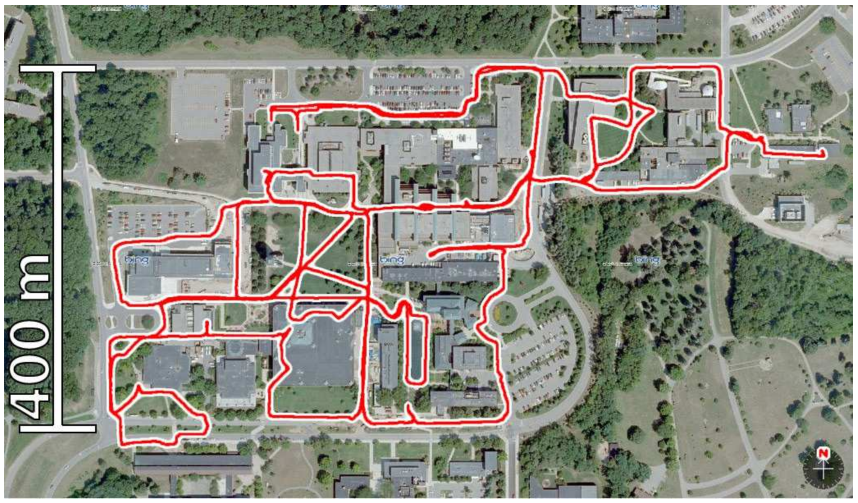
The dataset contains 34.9 hours of logs covering 147.4 km of robot trajectory. The dataset was collected in 27 discrete mapping sessions, Fig. A.1(b). Each session covers roughly the entire mapped area, however, the path for each session is varied. We also varied the time of day for each session—from early morning to just after dusk. Each session contains data from indoor and outdoor environments. The dataset contains many dynamic elements, including pedestrians, bicyclists, and vehicle traffic. Because we repeatedly traverse the same environment the dataset also captures longer-term dynamics, including moving furniture, weather and lighting conditions, seasonal changes, and two large construction projects. Sample imagery and lidar data are shown in Fig. A.2(a) and Fig. A.2(b).

A.2 Sensors

Figure A.1 The North Campus long-term dataset. The Segway robotic platform used for experimental data collection is outfitted with an RTK GPS (1), omni-directional camera (2), 3D LIDAR (3), IMU (4), consumer-grade GPS (5), 1-axis FOG (6), 2D LIDARs (7), and CPU (8) (a). A sample trajectory from one session of data collection, overlaid on satellite imagery is shown in (b).

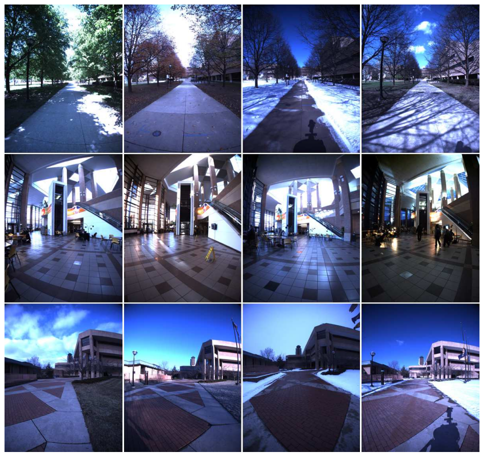


(a) Segway Robot

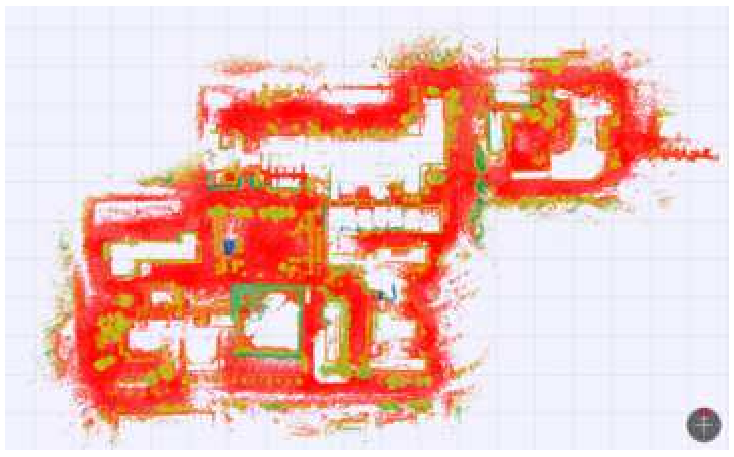


(b) Sample Trajectory

Figure A.2 Sample images and point cloud from North Campus dataset. Sample images from the dataset (only forward camera shown) are shown in (a). A sample LIDAR point cloud from single session, colored by height above ground is shown in (b).



(a) Sample Images



(b) Sample Point Cloud

The sensors collected in the dataset include:

(a) *Velodyne HDL-32E LIDAR*: The HDL-32E has 32 lasers mounted on a platform that spins about its vertical axis to provide a full 360 degree azimuthal field of view. The range of the sensor is 100 meters. We captured our dataset with the laser spinning at 10 Hz.

(b) *Pointgrey Ladybug3 omni-directional camera*: The Pointgrey Ladybug3 (LB3) is a high resolution omni-directional camera system. It has six 2-Megapixel (1600x1200) cameras, with five CCDs positioned in a horizontal ring and one positioned vertically, that enable the system to collect video from more than 80% of the full sphere. We collected our dataset at full resolution (i.e. 1600x1200) and 5 fps in a JPG compressed format.

(c) *Hokuyo UTM-30LX LIDAR*: The UTM-30 is a single beam LIDAR with a 30 meter range and a 270 degree field of view. The UTM-30 is mounted horizontally on the front of the Segway platform.

(d) *Hokuyo URG-04LX LIDAR*: The URG-04 is a single beam LIDAR with a 4 meter range and a 240 degree field of view. The URG-04 is mounted in a "push-broom" configuration to sweep out the ground plane in front of the vehicle.

(e) *Microstrain 3DM-GX3-45 IMU*: The GX3 contains 3-axis accelerometers, gyroscopes, and magnetometers, and an integrated GPS receiver. Its internal signal processor provides filtered 3D position, velocity, and attitude at 100 Hz.

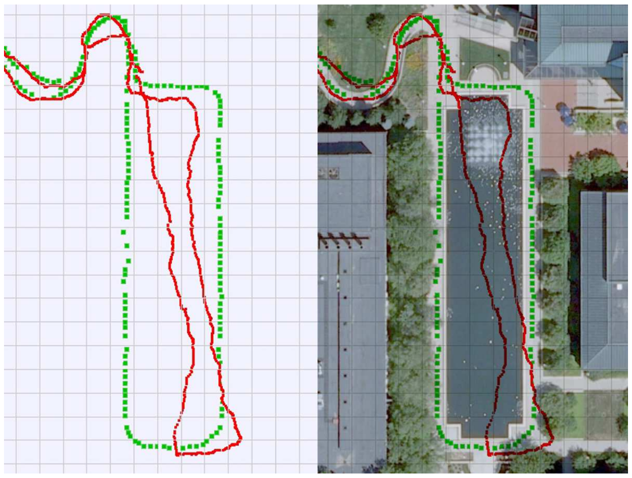
(f) *KVH DSP-5000 single-axis FOG*: The KVH fiber optic gyro provides highly accurate rotation measurements around a single axis. On the Segway platform it is used to measure yaw.

(g) *Garmin 18x 5Hz*: The 18x provides consumer grade GPS at 5 Hz (Fig. A.3).

(h) *NovAtel DL-4 plus RTK GPS*: The DL-4 GPS receiver provides highly accurate, Real-Time Kinematic (RTK) corrected GPS at 1 Hz. A NovAtel RTK base station was installed on campus to provide corrections. Outdoors, this provides highly accurate position information to ground-truth the robot trajectory (Fig. A.3).

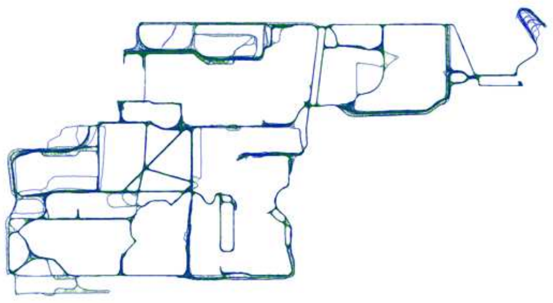
A.3 Ground-Truth

Figure A.**3** Comparison of GPS performance between RTK GPS (green) and consumer grade GPS (red) on North Campus. The true trajectory follows the RTK solution around the fountain. This performance is typical for many of the outdoor portions of the dataset as the consumer GPS is more affected by the close proximity of campus buildings

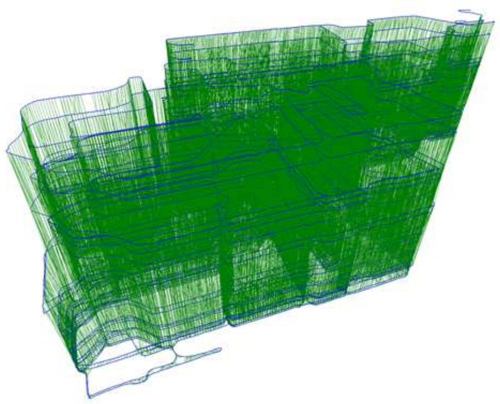


We have preprocessed a large SLAM solution, Fig. A.4, with all sessions using laser scan matching and RTK GPS (Fig. A.3) to provide ground truth robot pose. To compute ground truth poses between nodes in the graph we interpolate based on the odometry.

Figure A.**4** North Campus dataset ground truth. The ground truth SLAM graph comprised of all sessions is shown in (a). Links include odometry (blue) and 3D lidar scan matching (green). An oblique view scaled by time in the z-axis is shown in (b). Each layer along the z-axis represents a mapping session.



(a) Top View



(b) Time Scaled

A.4 Coordinate Frame Conventions

In this section we describe the coordinate frame conventions used in the North Campus dataset. Following the conventions described by Eustice [51] we define the 6-DOF pose of frame j with respect to frame i as

$$\mathbf{x}_{ij} = \left[{}^{i}\mathbf{t}_{ij}{}^{\top}, \boldsymbol{\Theta}_{ij}^{\top}\right]^{\top} = [x_{ij}, y_{ij}, z_{ij}, \phi_{ij}, \theta_{ij}, \psi_{ij}]^{\top}.$$

Here, ${}^{i}\mathbf{t}_{ij}$ is a translation 3-vector from i to j as expressed in frame i, and $\boldsymbol{\Theta}_{ij}$ is a 3-vector of Euler angles with ϕ representing roll about the x axis, θ as pitch about y, and ψ as yaw about z. To produce the 3×3 orthonormal rotation matrix that rotates frame j into frame i, the Euler angles are applied in $\mathrm{rotz}(\psi) \rightarrow \mathrm{roty}(\theta) \rightarrow \mathrm{rotx}(\phi)$ order yielding

$$\begin{aligned}
{}^{i}_{j}\mathrm{R} &= \mathrm{rotxyz}(\boldsymbol{\Theta}_{ij}) \\
&= \mathrm{rotz}(\psi_{ij})^{\top}\,\mathrm{roty}(\theta_{ij})^{\top}\,\mathrm{rotx}(\phi_{ij})^{\top} \\
&= \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}^{\top} \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}^{\top} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix}^{\top} \\
&= \begin{bmatrix} \cos\psi\cos\theta & -\sin\psi\cos\phi + \cos\psi\sin\theta\sin\phi & \sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi \\ \sin\psi\cos\theta & \cos\psi\cos\phi + \sin\psi\sin\theta\sin\phi & -\cos\psi\sin\phi + \sin\psi\sin\theta\cos\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix}.
\end{aligned}$$

The 4×4 homogeneous coordinate transformation matrix from frame j to frame i defined by $\mathbf{x}_{ij}$ is then defined as

$$ {}^{i}_{j}\mathrm{H} = \begin{bmatrix} {}^{i}_{j}\mathrm{R} & {}^{i}\mathbf{t}_{ij} \\ \mathbf{0} & 1 \end{bmatrix}.$$

This 6-degree of freedom (DOF) convention is used for representing robot pose in the dataset and the rigid-body transformations between the vehicle and sensor coordinate frames.

Table **A.1** GPS Linearization Constants

Latitude Origin	lat_0	$42.293215°$
Longitude Origin	lon_0	$-83.709662°$
Altitude Origin	alt_0	260 m
Earth Equatorial Radius	r_e	$6,378,135$ m
Earth Polar Radius	r_p	$6,356,750$ m

Robot poses are represented in a local coordinate frame aligned with the cardinal directions, with x pointing north, y east, and z down. The origin of this coordinate frame is fixed in GPS coordinates as described in Table A.1. Converting between the local frame and GPS coordinates is done by linearizing around this origin. To define the transformation from GPS coordinates to the local frame, we first compute an approximation of the earth's radius in the north-south direction, r_{ns}, and in the east-west direction, r_{ew}, at the origin of the linearization,

$$\begin{aligned} r_{ns} &= \frac{(r_e r_p)^2}{\left((r_e \cos lat_0)^2 + (r_p \sin lat_0)^2\right)^{\frac{3}{2}}} \\ r_{ew} &= \frac{r_e^2}{\sqrt{(r_e \cos lat_0)^2 + (r_p \sin lat_0)^2}} \end{aligned},$$

where r_e and r_e are the equatorial and polar radii of the earth defined in Table A.1. Using the local radii we can convert from GPS coordinates to the local frame using

$$\begin{aligned} x &= \sin(lat - lat_0)\, r_{ns} \\ y &= \sin(lon - lon_0)\, r_{ew} \cos lat_0. \\ z &= alt_0 - alt \end{aligned}$$

Conversely, we can convert from the local frame to GPS coordinates using

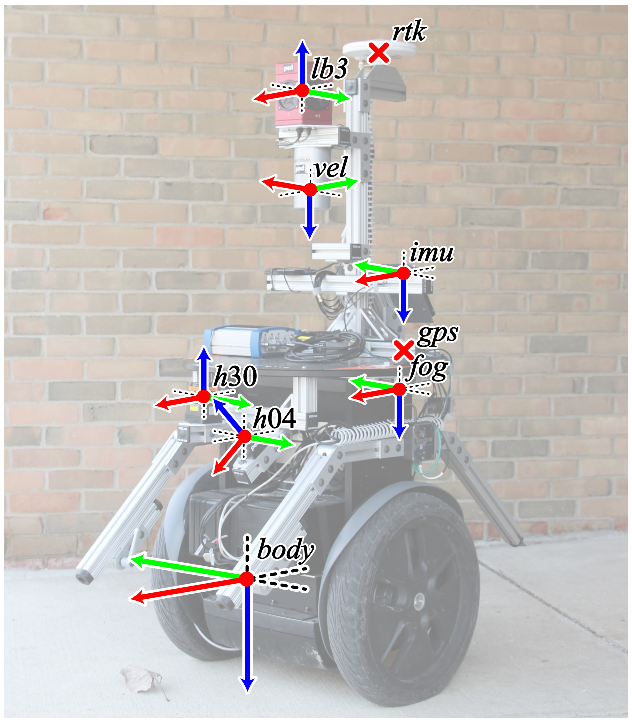
$$\begin{aligned} lat &= \arcsin\left(\frac{x}{r_{ns}}\right) + lat_0 \\ lon &= \arcsin\left(\frac{y}{r_{ew} \cos lat_0}\right) + lon_0. \\ alt &= alt_0 - z \end{aligned}$$

Table A.**2** Sensor Coordinate Frames

Sensor	Transform	x m	y m	z m	ϕ°	θ°	ψ°
Velodyne LIDAR	$x_{body,vel}$	0	0	-0.9	0	0	-90
Ladybug3 Base	$x_{body,lb3}$	0.03	0	-1.1	180	0	0
Microstrain IMU	$x_{body,imu}$	-0.11	-0.18	-0.71	0	0	0
KVH FOG	$x_{body,fog}$	0	-0.25	-0.49	0	0	0
Garmin GPS	$x_{body,gps}$	0	-0.25	-0.51	N/A	N/A	N/A
Novatel RTK GPS	$x_{body,rtk}$	-0.24	0	-1.24	N/A	N/A	N/A
Hokuyo UTM30-LX LIDAR	$x_{body,h30}$	0.28	0	-0.44	180	0	0
Hokuyo URG04-LX LIDAR	$x_{body,h04}$	0.31	0	-0.38	180	-40	0

The robot's body frame is centered on the axle between the Segway's wheels with x pointing forward, y to the right, and z down. Each sensor's frame of reference is defined with respect to

Figure A.5 Illustration of the Segway's sensor frames. For each frame, the x, y, and z axes are colored red, green and blue, respectivly.



this body frame. An illustration of the body and sensor frames is provided in Fig. A.5. The 6-DOF transformations for each sensor, relative to the body frame, are given in Table A.2.

A.5 Odometry Model

Odometry is estimated with an extended Kalman filter (EKF) that uses a differential-drive process model to integrate measurements from the Segway's wheel encoders and a single-axis FOG that observes change in yaw. Measurement updates are derived from a commodity IMU that observes roll, pitch, and body-frame angular rates.

We define the Segway's state at time t as,

$$\mathbf{x}_t = [x, y, \phi, \theta, \psi, p, q, r]^\top ,$$

where $[x, y]^\top$ represent the robot's translational position in a local frame, $[\phi, \theta, \psi]^\top$ are the Euler angles representing orientation, and $[p, q, r]^\top$ are the body-frame angular rates. We do not estimate the robot's altitude, z, in the local frame because change in z is not observable using the Segway's odometry sensors. Altitude is estimated in our ground-truth by fusing RTK GPS and LIDAR scan matching with the odometry.

The Segway process model predicts the translation and yaw of the robot using a differential drive model, and the roll and pitch using a constant velocity model. For a given time step, the process model takes as input

$$\mathbf{u}_t = [v_r, v_l, \delta_\psi]^\top ,$$

where v_r and v_l represent the speeds of the left and right wheels and δ_ψ denotes the change in yaw measured by the single-axis FOG. Given the two wheel speeds, we can compute the speed of the vehicle at the center of the wheelbase as

$$v_c = \frac{1}{2}(v_r + v_l). \tag{A.1}$$

The relationship between the body-frame angular rates and the roll and pitch rates can be derived, as described in [51, § A.3], by considering the inverse relationship where the Euler angle rotation sequence $\mathrm{rotz}(\psi) \rightarrow \mathrm{roty}(\theta) \rightarrow \mathrm{rotx}(\phi)$ is used to map Euler rates to body rates as

$$\begin{aligned}\begin{bmatrix} p \\ q \\ r \end{bmatrix} &= \begin{bmatrix} \dot\phi \\ 0 \\ 0 \end{bmatrix} + \mathrm{rotx}(\phi) \begin{bmatrix} 0 \\ \dot\theta \\ 0 \end{bmatrix} + \mathrm{rotx}(\phi)\,\mathrm{roty}(\theta) \begin{bmatrix} 0 \\ 0 \\ \dot\psi \end{bmatrix} \\ &= \underbrace{\begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix}}_{\mathcal{J}^{-1}} \begin{bmatrix} \dot\phi \\ \dot\theta \\ \dot\psi \end{bmatrix}.\end{aligned}$$

Thus, the mapping from body-frame rates to the roll and pitch rates is given by

$$\mathcal{J} = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \sin\phi\cos\theta \\ 0 & -\sin\phi & \cos\phi\cos\theta \end{bmatrix}^{-1} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix}.$$

The Segway's process model updates the roll and pitch of the vehicle using a constant velocity model. We compute the required angular velocities for roll and pitch as

$$\begin{bmatrix} \dot{\phi} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \end{bmatrix} \begin{bmatrix} p \\ q \\ r \end{bmatrix}. \tag{A.2}$$

Using (A.1) and (A.2) the process model is then defined as

$$\hat{\mathbf{x}}_{t+\delta_t} = f(\mathbf{x}_t, \mathbf{u}_t) + \boldsymbol{\omega}_t = \begin{bmatrix} x + v_c\cos(\theta)\delta_t \\ y + v_c\sin(\theta)\delta_t \\ \phi + \dot{\phi}\delta_t \\ \theta + \dot{\theta}\delta_t \\ \psi + \delta_\psi \\ p \\ q \\ r \end{bmatrix} + \boldsymbol{\omega}_t, \tag{A.3}$$

where $\boldsymbol{\omega}_t \sim \mathcal{N}\big(\mathbf{0}, \mathrm{Q}\big)$ and δ_t is the duration of the time step. The process model noise is comprised of two terms: one capturing the uncertainties associated with the control vector, and another capturing uncertainties in the constant velocity terms. The process model noise is defined as

$$\mathrm{Q} = \frac{\partial f}{\partial \mathbf{u}_t}\,\mathrm{diag}([\sigma_{v_r}^2, \sigma_{v_l}^2, \sigma_{\delta_\phi}^2])\frac{\partial f}{\partial \mathbf{u}_t}^\top + \mathrm{diag}([0, 0, \sigma_\phi^2, \sigma_\theta^2, 0, \sigma_p^2, \sigma_q^2, \sigma_r^2]).$$

Measurement updates are derived from the Microstrain IMU, which observes the platform's roll, pitch and body-frame angular rates. This leads to linear observation models

$$\hat{\mathbf{z}}_{\phi\,\theta} = h_{\phi\,\theta}(\mathbf{x}_t) = \begin{bmatrix} \phi \\ \theta \end{bmatrix} + \boldsymbol{\nu}_{\phi\,\theta} \qquad \boldsymbol{\nu}_{\phi\,\theta} \sim \mathcal{N}\big(\mathbf{0}, \mathrm{diag}([\sigma_\phi^2, \sigma_\theta^2])\big)$$

$$\hat{\mathbf{z}}_{pqr} = h_{pqr}(\mathbf{x}_t) = \begin{bmatrix} p \\ q \\ r \end{bmatrix} + \boldsymbol{\nu}_{pqr} \qquad \boldsymbol{\nu}_{pqr} \sim \mathcal{N}\big(\mathbf{0}, \mathrm{diag}([\sigma_p^2, \sigma_q^2, \sigma_r^2])\big).$$

In order to produce the relative odometry factors between poses that are used in our SLAM systems, the EKF tracks the current pose of the robot and the pose of the last node added to the graph in a delayed-state framework [52]. Letting $\mathbf{x}_{li}$ denote the pose associated with the last node added to the SLAM graph, and $\mathbf{x}_{lj}$ denote the current robot pose, the EKF estimates the distribution

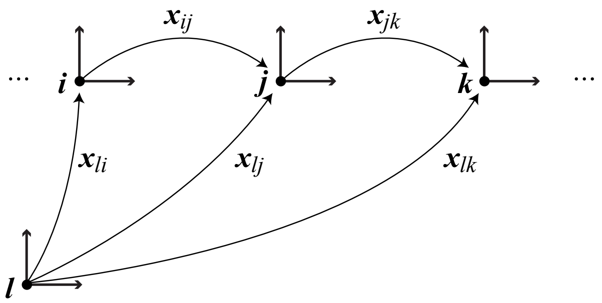
$$p(\mathbf{x}_{li}, \mathbf{x}_{lj}) \sim \mathcal{N}\left(\left[\begin{smallmatrix} \boldsymbol{\mu}_{li} \\ \boldsymbol{\mu}_{lj} \end{smallmatrix}\right], \left[\begin{smallmatrix} \Sigma_{li,li} & \Sigma_{li,lj} \\ \Sigma_{lj,li} & \Sigma_{lj,lj} \end{smallmatrix}\right]\right).$$

When we wish to add a new node associated with the pose $\mathbf{x}_{lj}$ to the graph, we can compute the relative transform from the last node to the current robot pose, using the "tail-to-tail" function described by Smith et al. [152]. Because the delayed-state EKF tracks the correlation between the current robot pose and the last node added to the graph, we can also compute a first-order approximation of the uncertainty of the odometry factor. This yields the relative factor

$$p(\mathbf{x}_{ij}) \sim \mathcal{N}\left(\ominus \mathbf{x}_{li} \oplus \mathbf{x}_{lj},\ {}_{\ominus}\mathrm{J}_{\oplus} \left[\begin{smallmatrix} \Sigma_{li,li} & \Sigma_{li,lj} \\ \Sigma_{lj,li} & \Sigma_{lj,lj} \end{smallmatrix}\right] {}_{\ominus}\mathrm{J}_{\oplus}^{\top}\right),$$

where ${}_{\ominus}\mathrm{J}_{\oplus}$ is the Jacobian of the tail-to-tail function. We then marginalize the old pose, $\mathbf{x}_{li}$, from the delayed-state filter and augment the state with a new vector of variables to track the current pose of the robot, $\mathbf{x}_{lk}$. This process is illustrated in Fig. A.6.

Figure A.6 Generating odometry factors using a delayed-state EKF. The delayed-state EKF estimates the pose of the last node added to the graph and the current pose of the robot. The joint distribution of these two poses is used to produce the relative odometry factor.



BIBLIOGRAPHY

BIBLIOGRAPHY

[1] M. Achtelik, M. Achtelik, Y. Brunet, M. Chli, S. Chatzichristofis, J.-D. Decotignie, K.-M. Doth, F. Fraundorfer, L. Kneip, D. Gurdan, L. Heng, E. Kosmatopoulos, L. Doitsidis, G. H. Lee, S. Lynen, A. Martinelli, L. Meier, M. Pollefeys, D. Piguet, A. Renzaglia, D. Scaramuzza, R. Siegwart, J. Stumpf, P. Tanskanen, C. Troiani, and S. Weiss. SFly: Swarm of micro flying robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2649–2650, Vilamoura, Portugal, Oct. 2012. (p. 9)

[2] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard. Robust map optimization using dynamic covariance scaling. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 62–69, Karlsruhe, Germany, May 2013. (p. 13, 82, 91)

[3] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. Building rome in a day. *Communications of the ACM*, 54(10):105–112, 2011. (p. 9)

[4] S. Anderson and T. D. Barfoot. Towards relative continuous-time SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1033–1040, Karlsruhe, Germany, May 2013. (p. 12)

[5] B. Babenko, P. Dollar, and S. Belongie. Task specific local region matching. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1–8, Rio de Janeiro, Brazil, Oct. 2007. (p. 14, 59, 60)

[6] A. Bacha, C. Bauman, R. Faruque, M. Fleming, C. Terwelp, C. Reinholtz, D. Hong, A. Wicks, T. Alberi, D. Anderson, S. Cacciola, P. Currier, A. Dalton, J. Farmer, J. Hurdus, S. Kimmel, P. King, A. Taylor, D. V. Covern, and M. Webste. Odin: Team VictorTango's entry in the DARPA Urban Challenge. *Journal of Field Robotics*, 25:467–492, 2008. (p. 9)

[7] T. Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (SLAM): Part II. *IEEE Robotics and Automation Magazine*, 13(3):108–117, Sept. 2006. (p. 1)

[8] T. Bailey, J. Nieto, J. Guivant, M. Stevens, and E. Nebot. Consistency of the EKF–SLAM algorithm. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3562–3568, Beijing, China, Oct. 2006. (p. 10)

[9] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008. (p. 7, 14, 56, 63, 68)

[10] P. Beeson, J. Modayil, and B. Kuipers. Factoring the mapping problem: Mobile robot map-building in the hybrid spatial semantic hierarchy. *International Journal of Robotics Research*, 29(4):428–459, Apr. 2010. (p. 12)

[11] J. Bergstra and Y. Bengio. Slow, decorrelated features for pretraining complex cell-like networks. In *Proceedings of the Advances in Neural Information Processing Systems Conference*, pages 99–107, Vancover, Canada, Dec. 2009. (p. 64)

[12] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: A CPU and GPU math compiler in python. In *Proceedings of the Python for Scientific Computing Conference*, Austin, TX, 2010. (p. 67)

[13] P. J. Besl and H. D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, Feb. 1992. (p. 6, 15)

[14] P. Biber and T. Duckett. Dynamic maps for long-term operation of mobile service robots. In *Proceedings of the Robotics: Science & Systems Conference*, pages 17–24, 2005. (p. 14, 79)

[15] P. Biber and T. Duckett. Experimental analysis of sample-based maps for long-term SLAM. *International Journal of Robotics Research*, 28(1):20–33, 2009. (p. 14, 79)

[16] J. Bohren, T. Foote, J. Keller, A. Kushleyev, D. Lee, A. Stewart, P. Vernaza, J. Derenick, J. Spletzer, and B. Satterfield. Little Ben: The Ben Franklin Racing Team's entry in the 2007 DARPA Urban Challenge. *Journal of Field Robotics*, 25(9):598–614, 2008. (p. 9)

[17] M. Bosse, P. Newman, J. Leonard, and S. Teller. An Atlas framework for scalable mapping. *International Journal of Robotics Research*, 23:1113–1139, Dec. 2004. (p. 10)

[18] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. (p. 48)

[19] J. Bromley, I. Guyon, Y. LeCun, E. Sackinger, and R. Shah. Signature verification using a 'Siamese' time delay neural network. *International Journal of Pattern Recognition and Artifical Intelligence*, 7(4):669–688, 1993. (p. 59)

[20] M. Brown, G. Hua, and S. Winder. Discriminative learning of local image descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(1):43–57, 2010. (p. 14, 59, 60, 64, 65)

[21] M. Brown, S. Winder, and G. Hua. Learning local image descriptors data. `http://www.cs.ubc.ca/~mbrown/patchdata/patchdata.html`, 2011. (p. 66)

[22] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun. Experiences with an interactive museum tour-guide robot. *Artificial Intelligence*, 114:3–55, 2003. (p. 14, 79)

[23] N. Carlevaris-Bianco and R. M. Eustice. Multi-view registration for feature-poor underwater imagery. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 423–430, Shanghai, China, May 2011. (p. 90)

[24] N. Carlevaris-Bianco and R. M. Eustice. Learning temporal co-observability relationships for lifelong robotic mapping. In *IROS Workshop on Lifelong Learning for Mobile Robotics Applications*, Vilamoura, Portugal, October 2012. (p. 15, 97)

[25] N. Carlevaris-Bianco and R. M. Eustice. Generic factor-based node marginalization and edge sparsification for pose-graph SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 5728–5735, Karlsruhe, Germany, May 2013. (p. 18, 22, 40, 42)

[26] N. Carlevaris-Bianco and R. M. Eustice. Long-term simultaneous localization and mapping with generic linear constraint node removal. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1034–1041, Nov. 2013. (p. 22, 42, 52, 77)

[27] N. Carlevaris-Bianco and R. M. Eustice. Conservative edge sparsification for graph SLAM node removal. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 854–860, Hong Kong, China, June 2014. (p. 18)

[28] N. Carlevaris-Bianco and R. M. Eustice. Learning visual feature descriptors for dynamic lighting conditions. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2769–2776, Chicago, IL, Sept. 2014. (p. 56)

[29] N. Carlevaris-Bianco, A. Mohan, J. R. McBride, and R. M. Eustice. Visual localization in fused image and laser range data. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4378–4385, San Francisco, CA, Sept. 2011. (p. 9)

[30] N. Carlevaris-Bianco, M. Kaess, and R. M. Eustice. Generic node removal for factor-graph SLAM. *IEEE Transactions on Robotics*, 30(6):1371–1385, 2014. (p. 18)

[31] J. Castellanos, J. Neira, and J. Tardós. Limits to the consistency of EKF-based SLAM. In *IFAC Symp. Intell. Auton. Veh.*, Lisbon, Portugal, July 2004. (p. 10)

[32] S. M. Chaves, A. Kim, and R. M. Eustice. Opportunistic sampling-based planning for active visual SLAM. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3073–3080, Chicago, IL, USA, September 2014. (p. 12)

[33] Y.-H. Choi, T.-K. Lee, and S.-Y. Oh. A line feature based SLAM with low grade range sensors using geometric constraints and active exploration for mobile robot. *Autonomous Robots*, 24(1):13–27, 2008. (p. 9)

[34] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verication. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 539–546, San Diego, CA, June 2005. (p. 59)

[35] C. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14:462–467, 1968. (p. 32)

[36] W. Churchill and P. Newman. Practice makes perfect? managing and leveraging visual experiences for lifelong navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4525–4532, Saint Paul, MN, May 2012. (p. 14, 80)

[37] W. Churchill and P. Newman. Experience-based navigation for long-term localisation. *International Journal of Robotics Research*, 32(14):1645–1661, 2013. (p. 15, 58)

[38] M. Cummins and P. Newman. FAB-MAP: Probabilistic localization and mapping in the space of appearance. *International Journal of Robotics Research*, 27(6):647–665, June 2008. (p. 12, 15, 108)

[39] A. Cunningham, M. Paluri, and F. Dellaert. DDF-SAM: Fully distributed SLAM using constrained factor graphs. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3025 – 3030, Taipei, Taiwan, October 2010. (p. 9)

[40] A. Cunningham, V. Indelman, and F. Dellaert. DDF-SAM 2.0: Consistent distributed smoothing and mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 5200–5207, Karlsruhe, Germany, May 2013. (p. 23)

[41] A. Davison. Active search for real-time vision. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 66–73, Beijing, China, Oct. 2005. (p. 32)

[42] A. Davison and N. Kita. Sequential localisation and map-building in computer vision and robotics. In *SMILE Workshop Proceedings of the European Conference on Computer Vision*, Dublin, Ireland, 2000. (p. 10)

[43] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6): 1–16, June 2007. (p. 10)

[44] F. Dayoub and T. Duckett. An adaptive appearance-based map for long-term topological localization of mobile robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3364–3369, Nice, France, Sept. 2008. (p. 14, 79)

[45] F. Dellaert and M. Kaess. Square root SAM: Simultaneous localization and mapping via square root information smoothing. *International Journal of Robotics Research*, 25(12): 1181–1203, 2006. (p. 4, 11, 19, 20, 26, 28, 32, 78)

[46] F. Dellaert, J. Carlson, V. Ila, K. Ni, and C. E. Thorpe. Subgraph-preconditioned conjugate gradients for large scale SLAM. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2566–2571, Taipei, Taiwan, Oct. 2010. (p. 11)

[47] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: Part I. *IEEE Robotics and Automation Magazine*, 13(2):99–110, June 2006. (p. 1)

[48] E. Eade, P. Fong, and M. E. Munich. Monocular graph SLAM with complexity reduction. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3017–3024, Taipei, Taiwan, Oct. 2010. (p. 13, 16, 19, 21, 22, 25)

[49] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *IEEE Computer*, 22(6):46–57, June 1989. (p. 12)

[50] R. Eustice, M. Walter, and J. Leonard. Sparse extended information filters: Insights into sparsification. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3281–3288, Edmonton, Alberta, Canada, Aug. 2005. (p. 10, 21, 23, 27)

[51] R. M. Eustice. *Large-area visually augmented navigation for autonomous underwater vehicles*. PhD thesis, Massachusetts Institute of Technology / Woods Hole Oceaonographic Institution Joint Program, June 2005. (p. 114, 117)

[52] R. M. Eustice, H. Singh, and J. J. Leonard. Exactly sparse delayed-state filters for view-based SLAM. *IEEE Transactions on Robotics*, 22(6):1100–1114, 2006. (p. 10, 19, 26, 78, 81, 118)

[53] R. M. Eustice, H. Singh, J. J. Leonard, and M. R. Walter. Visually mapping the RMS Titanic: Conservative covariance estimates for SLAM information filters. *International Journal of Robotics Research*, 25(12):1223–1242, 2006. (p. 9)

[54] R. M. Eustice, O. Pizarro, and H. Singh. Visually augmented navigation for autonomous underwater vehicles. *IEEE Journal of Oceanic Engineering*, 33(2):103–122, Apr. 2008. (p. 90)

[55] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with application to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, June 1981. (p. 7)

[56] J. Folkesson and H. Christensen. Graphical SLAM—a self-correcting map. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 383–390, New Orleans, LA, USA, April 2004. (p. 13, 22, 24)

[57] D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journal of Artificial Intelligence Research*, 11:391–427, Nov. 1999. (p. 14, 79)

[58] J.-M. Frahm, P. Fite-Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik, , and M. Pollefeys. Building rome on a cloudless day. In *Proceedings of the European Conference on Computer Vision*, pages 368–381, Hersonissos, Greece, Sept. 2010. (p. 9)

[59] U. Frese. Treemap: An O(Log N) algorithm for simultaneous localization and mapping. In C. Freksa, editor, *Spatial Cognition IV*. Springer Verlag, 2004. (p. 22)

[60] U. Frese. Efficient 6-DOF SLAM with treemap as a generic backend. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4814—4819, Rome, Italy, Apr. 2007. (p. 22)

[61] P. Furgale and T. D. Barfoot. Visual teach and repeat for long-range rover autonomy. *Journal of Field Robotics*, 27(5):534–560, 2010. (p. 9)

[62] P. Furgale, T. D. Barfoot, and G. Sibley. Continuous-time batch estimation using temporal basis functions. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2088–2095, Saint Paul, MN, USA, May 2012. (p. 12)

[63] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In *Proceedings of the Advances in Neural Information Processing Systems Conference*, pages 513–520, Whistler, Canada, Dec. 2004. (p. 60)

[64] G. Grisetti, D. Lodi Rizzini, C. Stachniss, E. Olson, and W. Burgard. Online constraint network optimization for efficient maximum likelihood map learning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1880–1885, Pasadena, CA, May 2008. (p. 11)

[65] G. Grisetti, R. Kummerle, C. Stachniss, U. Frese, and C. Hertzberg. Hierarchical optimization on manifolds for online 2D and 3D mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 273–278, Anchorage, AK, USA, May 2010. (p. 11)

[66] R. Hadsell, S. Chopra, and Y. LeCun. Dimensionality reduction by learning an invariant mapping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1735–1742, New York, NY, June 2006. (p. 59, 61, 62)

[67] D. Hähnel, R. Triebel, W. Burgard, and S. Thrun. Map building with mobile robots in dynamic environments. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1557–1563, Taipei, Taiwan, Sept. 2003. (p. 14, 79)

[68] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004. (p. 7, 14, 71)

[69] G. Hinton and S. Roweis. Stochastic neighbor embedding. In *Proceedings of the Advances in Neural Information Processing Systems Conference*, pages 1–8, Vancover, Canada, Dec. 2002. (p. 60)

[70] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313:504–507, 2006. (p. 59)

[71] F. S. Hover, R. M. Eustice, A. Kim, B. Englot, H. Johannsson, M. Kaess, and J. J. Leonard. Advanced perception, navigation and planning for autonomous in-water ship hull inspection. *International Journal of Robotics Research*, 31(12):1445–1464, 2012. (p. 9, 36)

[72] G. Hua, M. Brown, and S. Winder. Discriminant embedding for local image descriptors. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1–8, Rio de Janeiro, Brazil, Oct. 2007. (p. 14, 59, 60, 64)

[73] G. Huang, M. Kaess, and J. J. Leonard. Consistent sparsification for graph optimization. In *Proceedings of the European Conference on Mobile Robotics*, pages 150–157, Barcelona, Spain, Sept. 2013. (p. 13, 22, 23, 24, 43, 44, 47, 52, 54)

[74] P. J. Huber. *Robust statistics*. Springer, 2011. (p. 13)

[75] V. Ila, J. M. Porta, and J. Andrade-Cetto. Information-based compact pose SLAM. *IEEE Transactions on Robotics*, 26(1):78–93, 2010. (p. 13, 20, 82, 97)

[76] W. Jeong and K. M. Lee. CV-SLAM: A new ceiling vision-based SLAM technique. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3195–3200, Edmonton, Canada, Aug. 2005. (p. 9)

[77] Y.-D. Jian and F. Dellaert. iSPCG: Incremental subgraph-preconditioned conjugate gradient method for online SLAM with many loop-closures. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2647–2653, Chicago, IL, USA, Sept. 2014. (p. 11)

[78] H. Johannsson, M. Kaess, M. Fallon, and J. J. Leonard. Temporally scalable visual SLAM using a reduced pose graph. In *RSS Workshop on Long-term Operation of Autonomous Robotic Systems in Changing Environments*, Sydney, Australia, July 2012. (p. 13)

[79] H. Johannsson, M. Kaess, M. Fallon, and J. J. Leonard. Temporally scalable visual SLAM using a reduced pose graph. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 54–61, Karlsruhe, Germany, May 2013. (p. 16, 19, 20, 21, 22, 79, 84)

[80] E. Johns and G.-Z. Yang. Feature co-occurrence maps: Appearance-based localisation throughout the day. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3212–3218, Karlsruhe, Germany, May 2013. (p. 15, 58, 79)

[81] E. Johns and G.-Z. Yang. Generative methods for long-term place recognition in dynamic scenes. *International Journal of Computer Vision*, 106(3):297–314, 2013. (p. 15, 58, 79)

[82] M. Johnson-Roberson, O. Pizarro, S. B. Williams, and I. Mahon. Generation and visualization of large-scale three-dimensional reconstructions from underwater robotic surveys. *Journal of Field Robotics*, 27(1):21–51, 2010. (p. 9)

[83] S. J. Julier and J. K. Uhlmann. A non-divergent estimation algorithm in the presence of unknown correlations. In *Proceedings of the American Control Conference*, pages 2369–2373, Albuquerque, NM, USA, June 1997. (p. 46)

[84] S. J. Julier and J. K. Uhlmann. A counter example to the theory of simultaneous localization and map building. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4238–4243, Seoul, South Korea, May 2001. (p. 10)

[85] M. Kaess and F. Dellaert. Covariance recovery from a square root information matrix for data association. *Robotics and Autonomous Systems*, 57:1198–1210, 2009. (p. 36, 49, 85)

[86] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Transactions on Robotics*, 24(6):1365–1378, 2008. (p. 11, 19, 20, 24, 28, 32, 36, 78, 85)

[87] M. Kaess, H. Johannsson, D. Rosen, N. Carlevaris-Bianco, and J. Leonard. Open source implementation of iSAM. `http://people.csail.mit.edu/kaess/isam`, 2010. (p. 24, 30, 36, 49, 85)

[88] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping using the Bayes tree. *International Journal of Robotics Research*, 31(2):216–235, Feb. 2011. (p. 11)

[89] A. Kim and R. M. Eustice. Real-time visual SLAM for autonomous underwater hull inspection using visual saliency. *IEEE Transactions on Robotics*, 29(3):719–733, 2013. (p. 9)

[90] A. Kim and R. M. Eustice. Active visual SLAM for robotic area coverage: Theory and experiment. *International Journal of Robotics Research*, 2014. In Press. (p. 12)

[91] B. Kim, M. Kaess, L. Fletcher, J. Leonard, A. Bachrach, N. Roy, and S. Teller. Multiple relative pose graphs for robust cooperative mapping. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3185–3192, Anchorage, AK, USA, May 2010. (p. 11)

[92] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009. (p. 53, 103, 107)

[93] K. Konolige and M. Agrawal. FrameSLAM: From bundle adjustment to real-time visual mapping. *IEEE Transactions on Robotics*, 24(5):1066–1077, 2008. (p. 19, 78)

[94] K. Konolige and J. Bowman. Towards lifelong visual maps. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1156–1163, St. Louis, MO, USA, Oct. 2009. (p. 13, 14, 15, 16, 19, 21, 22, 58, 79, 80)

[95] K. Konolige, J. Bowman, J. Chen, P. Mihelich, M. Calonder, V. Lepetit, and P. Fua. View-based maps. *International Journal of Robotics Research*, 29(8):941–957, 2010. (p. 80)

[96] K. Konolige, G. Grisetti, R. Kummerle, W. Burgard, B. Limketkai, and R. Vincent. Efficient sparse pose adjustment for 2D mapping. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1156–1163, St. Louis, MO, Oct. 2010. (p. 11)

[97] H. Kretzschmar and C. Stachniss. Information-theoretic compression of pose graphs for laser-based SLAM. *International Journal of Robotics Research*, 31:1219–1230, 2012. (p. 13, 16, 19, 21, 22, 25, 26, 32, 36, 79)

[98] B. Kuipers. The spatial semantic hierarchy. *Artificial Intelligence*, 119:191–233, May 2000. (p. 12)

[99] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3607–3613, Shanghai, China, May 2011. (p. 11, 24)

[100] H. Lategahn, J. Beck, B. Kitt, and C. Stiller. How to learn an illumination robust image feature for place recognition. In *IEEE Intelligent Vehicles Symposium*, pages 285–291, Gold Coast, Australia, June 2013. (p. 15, 58, 64, 68)

[101] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Nov. 1998. (p. 62, 63)

[102] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman, O. Koch, Y. Kuwata, D. Moore, E. Olson, S. Peters, J. Teo, R. Truax, M. Walter, D. Barrett, A. Epstein, K. Maheloni, K. Moyer, T. Jones, R. Buckley, M. Antone, R. Galejs, S. Krishnamurthy, and J. Williams. A perception-driven autonomous urban vehicle. *Journal of Field Robotics*, 25(10):727–774, Oct. 2008. (p. 9)

[103] J. Levinson and S. Thrun. Robust vehicle localization in urban environments using probabilistic maps. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4372–4378, May 2010. (p. 9)

[104] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004. (p. 7, 14, 56, 63, 68, 69, 90)

[105] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4:333–349, 1997. (p. 11, 19, 78)

[106] M. Magnusson. *The Three-Dimensional Normal-Distributions Transform – an Efficient Representation for Registration, Surface Analysis, and Loop Detection*. PhD thesis, Örebro University, 2009. Örebro Studies in Technology 36. (p. 15, 81)

[107] J. Markoff. Google cars drive themselves, in traffic. *The New York Times*, 10:A1, Oct. 2010. (p. 9)

[108] M. Mazuran, T. G. Diego, S. Luciano, and W. Burgard. Nonlinear graph sparsification for SLAM. In *Proceedings of the Robotics: Science & Systems Conference*, pages 1–8, Berkeley, CA, USA, July 2014. (p. 13, 22, 107)

[109] J. R. McBride, J. C. Ivan, D. S. Rhode, J. D. Rupp, M. Y. Rupp, J. D. Higgins, D. D. Turner, and R. M. Eustice. A perspective on emerging automotive safety applications, derived from lessons learned through participation in the DARPA grand challenges. *Journal of Field Robotics*, 25(10):808–840, Oct. 2008. (p. 9)

[110] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid. RSLAM: A system for large-scale mapping in constant-time using stereo. *International Journal of Computer Vision*, 94 (2):198–214, 2011. (p. 11)

[111] A. Mikulik, M. Perdoch, O. Chum, and J. Matas. Learning a fine vocabulary. In *Proceedings of the European Conference on Computer Vision*, pages 1–14, Hersonissos, Greece, Sept. 2010. (p. 59)

[112] M. Milford and G. Wyeth. SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1643–1649, Saint Paul, MN, USA, May 2012. (p. 15)

[113] M. Milford, E. Vig, W. Scheirer, and D. Cox. Towards condition-invariant, top-down visual place recognition. In *Australasian Conference on Robotics and Automation*, pages 1–10, Sydney, Australia, Dec. 2013. (p. 15, 58, 108)

[114] M. J. Milford and G. F. Wyeth. Mapping a suburb with a single camera using a biologically inspired SLAM system. *IEEE Transactions on Robotics*, 24(5):1038–1053, 2008. (p. 12)

[115] I. Miller, M. Campbell, D. Huttenlocher, F.-R. Kline, A. Nathan, S. Lupashin, J. Catlin, B. Schimpf, P. Moran, N. Zych, E. Garcia, M. Kurdziel, and H. Fujishima. Team Cornell's Skynet: Robust perception and planning in an urban environment. *Journal of Field Robotics*, 25(8):493–527, 2008. (p. 9)

[116] T. P. Minka. Inferring a Gaussian distribution. Technical report, MIT Media Lab, 2001. (p. 34, 35)

[117] H. Mobahi, R. Collobert, and J. Weston. Deep learning from temporal coherence in video. In *Proceedings of the International Conference on Machine Learning*, pages 737–744, Montreal, Canada, June 2009. (p. 59)

[118] J. Modayil and B. Kuipers. The initial development of object knowledge by a learning robot. *Robotics and Autonomous Systems*, 56:879–890, Nov. 2008. (p. 14, 79)

[119] 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 Artifical Intelligence*, pages 593–598, Edmonton, Canada, July 2002. (p. 12)

[120] M. Montemerlo, S. Thrun, and W. Whittake. Conditional particle filters for simultaneous mobile robot localization and people-tracking. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 695–701, Washington, D.C., May 2002. (p. 14, 79)

[121] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proceedings of the International Joint Conference on Artifical Intelligence*, pages 1151–1156, Acapulco, Mexico, Aug. 2003. (p. 12)

[122] H. Moravec and A. Elfes. High resolution maps from wide angle sonar. In *Proceedings of the IEEE International Conference on Robotics and Automation*, volume 2, pages 116–121, Mar. 1985. (p. 12)

[123] A. Murarka, J. Modayil, and B. Kuipers. Building local safety maps for a wheelchair robot using vision and lasers. In *The 3rd Canadian Conference on Computer and Robot Vision*, pages 25–31, Quebec, Canada, June 2006. (p. 9)

[124] A. Murarka, S. Gulati, P. Beeson, and B. Kuipers. Towards a safe, low-cost, intelligent wheelchair. In *Workshop on Planning, Perception and Navigation for Intelligent Vehicles*, pages 42–50, St. Louis, MO, Oct. 2009. (p. 9)

[125] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the International Conference on Machine Learning*, pages 807–814, Haifa, Israel, June 2010. (p. 63)

[126] T. Naseer, L. Spinello, W. Burgard, and C. Stachniss. Robust visual robot localization across seasons using network flows. In *Proceedings of the AAAI National Conference on Artifical Intelligence*, pages 1–7, Quebec, Canada, July 2014. (p. 15, 108)

[127] J. Neira and J. Tardos. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation*, 17(6):890–897, 2001. (p. 6, 37)

[128] P. Neubert, N. Sunderhauf, and P. Protzel. Appearance change prediction for long-term navigation across seasons. In *Proceedings of the European Conference on Mobile Robotics*, pages 198–203, Barcelona, Spain, Sept. 2013. (p. 15, 58)

[129] K. Ni and F. Dellaert. Multi-level submap based SLAM using nested dissection. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2558–2565, Taipei, Taiwan, Oct. 2010. (p. 11)

[130] K. Ni, D. Steedly, and F. Dellaert. Tectonic SAM: Exact; out-of-core; submap-based SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1678–1685, Rome, Italy, Apr. 2007. (p. 11)

[131] K. Ni, D. Steedly, and F. Dellaert. Out-of-core bundle adjustment for large-scale 3D reconstruction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1–8, Rio de Janeiro, Brazil, Oct. 2007. (p. 9)

[132] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 2161–2168, 2006. (p. 15, 59, 108)

[133] E. Olson and P. Agarwal. Inference on networks of mixtures for robust robot mapping. *International Journal of Robotics Research*, 32(7):826–840, July 2013. (p. 13, 82)

[134] E. Olson, J. Leonard, and S. Teller. Fast iterative alignment of pose graphs with poor initial estimates. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2262–2269, Orlando, FL, USA, May 2006. (p. 11, 19, 20, 78)

[135] P. Ozog and R. M. Eustice. Toward long-term, automated ship hull inspection with visual SLAM, explicit surface optimization, and generic graph-sparsication. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3832–3839, Hong Kong, China, June 2014. (p. 30, 91)

[136] P. Ozog, N. Carlevaris-Bianco, A. Kim, and R. M. Eustice. Long-term mapping techniques for ship hull inspection and surveillance using an autonomous underwater vehicle. *Journal of Field Robotics*, 2015. Submitted, Under Review. (p. 9)

[137] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua. Fast keypoint recognition using random ferns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3):448–461, 2010. (p. 67)

[138] G. Pandey, J. R. McBride, S. Savarese, and R. M. Eustice. Visually bootstrapped generalized ICP. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2660–2667, Shanghai, China, May 2011. (p. 90)

[139] J. Philbin, M. Isard, J. Sivic, and A. Zisserman. Descriptor learning for efficient retrieval. In *Proceedings of the European Conference on Computer Vision*, pages 677–691, Hersonissos, Greece, Sept. 2010. (p. 59, 64)

[140] L. Polok, V. Ila, M. Solony, P. Smrz, and P. Zemcik. Incremental block cholesky factorization for nonlinear least squares in robotics. In *Proceedings of the Robotics: Science & Systems Conference*, pages 1–8, Berlin, Germany, June 2013. (p. 11)

[141] E. Prassler, J. Scholz, and P. Fiorini. A robotics wheelchair for crowded public environment. *IEEE Robotics and Automation Magazine*, 8(1):38–45, 2002. (p. 9)

[142] A. Ranganathan, S. Matsumoto, and D. Ilstrup. Towards illumination invariance for visual localization. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3791–3798, Karlsruhe, Germany, May 2013. (p. 14, 15, 58, 59)

[143] C. R. Rao and S. K. Mitra. *Generalized Inverse of Matrices and its Applications*. John Wiley & Sons, 1971. (p. 29, 34, 45)

[144] P. Ridao, M. Carreras, D. Ribas, and R. Garcia. Visual inspection of hydroelectric dams using an autonomous underwater vehicle. *Journal of Field Robotics*, 27(6):759–778, 2010. (p. 9)

[145] R. Salakhutdinov and G. Hinton. Learning a nonlinear embedding by preserving class neighbourhood structure. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 412–419, San Juan, Puerto Rico, Mar. 2007. (p. 60)

[146] R. Salakhutdinov and G. Hinton. Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969–978, 2009. (p. 59)

[147] D. Scaramuzza, M. C. Achtelik, L. Doitsidis, F. Fraundorfer, E. Kosmatopoulos, A. Martinelli, M. W. Achtelik, M. Chli, S. Chatzichristofis, L. Kneip, D. Gurdan, L. Heng, G. H. Lee, S. Lynen, L. Meier, M. Pollefeys, A. Renzaglia, R. Siegwart, J. C. Stumpf, P. Tanskanen, C. Troiani, and S. Weiss. Vision-controlled micro flying robots: From system design to autonomous navigation and mapping in GPS-denied environments. *IEEE Robotics and Automation Magazine*, 21(3):26–40, 2014. (p. 9)

[148] A. V. Segal, D. Haehnel, and S. Thrun. Generalized-ICP. In *Proceedings of the Robotics: Science & Systems Conference*, Seattle, WA, June 2009. (p. 6, 15)

[149] G. Shakhnarovich. *Learning Task-Specific Similarity*. PhD thesis, Massachusetts Institute of Technology, Sept. 2005. (p. 59)

[150] G. Sibley, C. Mei, I. Reid, and P. Newman. Vast-scale outdoor navigation using adaptive relative bundle adjustment. *International Journal of Robotics Research*, 29(8):958–980, 2010. (p. 11)

[151] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2, pages 1470–1477, Nice, France, Oct. 2003. (p. 15, 59, 108)

[152] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In I. Cox and G. Wilfong, editors, *Autonomous Robot Vehicles*, pages 167–193. Springer-Verlag, 1990. (p. 10, 21, 24, 26, 81, 119)

[153] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3D. *ACM Transactions on Graphics*, 25(3):835–846, 2006. (p. 9)

[154] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *International Journal of Computer Vision*, 80(2):189–210, 2008. (p. 9)

[155] S. S. Srinivasa, D. Ferguson, C. J. Helfrich, D. Berenson, A. Collet, R. Diankov, G. Gallagher, G. Hollinger, J. Kuffner, and M. V. Weghe. HERB: A home exploring robotic butler. *Autonomous Robots*, 28(1):5–20, 2010. (p. 9)

[156] C. Stachniss and W. Burgard. Mobile robot mapping and localization in non-static environments. In *Proceedings of the AAAI National Conference on Artifical Intelligence*, pages 1324–1329, Pittsburgh, PA, July 2005. (p. 14, 80)

[157] D. Stavens and S. Thrun. Unsupervised learning of invariant features using video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1649–1656, San Francisco, CA, June 2010. (p. 59, 64)

[158] N. Sunderhauf and P. Protzel. Switchable constraints for robust pose graph SLAM. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1879–1884, Vilamoura, Portugal, Oct. 2012. (p. 13, 82)

[159] G. W. Taylor, I. Spiro, C. Bregler, and R. Fergus. Learning invariance through imitation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2729–2736, Providence, RI, June 2011. (p. 59, 60, 61)

[160] S. Thrun and M. Montemerlo. The graph SLAM algorithm with applications to large-scale mapping of urban structures. *International Journal of Robotics Research*, 25(5-6):403–429, 2006. (p. 19, 78)

[161] S. Thrun, Y. Liu, D. Koller, A. Ng, Z. Ghahramani, and H. Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *International Journal of Robotics Research*, 23(7/8):693–716, 2004. (p. 10, 13, 21, 23, 26, 27)

[162] S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. MIT Press, Cambridge, MA, Sept. 2005. (p. 23, 43)

[163] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney. Stanley: The robot that won the DARPA Grand Challenge. *Journal of Field Robotics*, 23(9):661–692, 2006. (p. 9)

[164] G. D. Tipaldi, D. Meyer-Delius, M. Beinhofer, and W. Burgard. Lifelong localization and dynamic map estimation in changing environments. In *RSS Workshop on Robots in Clutter*, 2012. (p. 14, 79)

[165] E. Tola, V. Lepetit, and P. Fua. DAISY: An efficient dense descriptor applied to wide-baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5): 815–830, 2010. (p. 14, 59, 63, 68)

[166] L. Toohey, O. Pizarro, and S. B. Williams. Multi-vehicle localisation with additive compressed factor graphs. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4584–4590, Chicago, IL USA, Sept. 2014. (p. 23)

[167] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment – a modern synthesis. In W. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, LNCS, pages 298–375. Springer-Verlag, 2000. (p. 7, 9)

[168] T. Trzcinski, M. Christoudias, P. Fua, and V. Lepetit. Boosting binary keypoint descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2874–2881, Portland, OR, USA, June 2013. (p. 14, 59)

[169] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. M. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, B. Salesky, Y. Seo, S. Singh, J. Snider, A. Stentz, W. Whittaker, Z. Wolkowicki, J. Ziglar, H. Bae, T. Brown, D. Demitrish, B. Litkouhi, J. Nickolaou, V. Sadekar, W. Zhang, J. Struble, M. Taylor, M. Darms, and D. Ferguson. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25:425–466, 2008. (p. 9)

[170] L. van der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008. (p. 72)

[171] L. Vandenberghe and S. Boyd. Semidefinite programming. *SIAM Review*, 38(1):49–95, 1996. (p. 48)

[172] L. Vandenberghe, S. Boyd, and S.-P. Wu. Determinant maximization with linear matrix inequality constraints. *SIAM Journal on Matrix Analysis and Applications*, 19(2):499–533, 1998. (p. 48)

[173] J. Vial, H. Durrant-Whyte, and T. Bailey. Conservative sparsification for efficient and consistent approximate estimation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 886–893, San Francisco, CA, USA, Sept. 2011. (p. 21, 23, 24, 43, 44, 47, 52)

[174] A. Walcott-Bryant, M. Kaess, H. Johannsson, and J. J. Leonard. Dynamic pose graph SLAM: Long-term mapping in low dynamic environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1871–1878, Vilamoura, Portugal, Oct. 2012. (p. 13, 16, 19, 21, 22, 79)

[175] J. M. Walls and R. M. Eustice. An origin state method for communication constrained cooperative localization with robustness to packet loss. *International Journal of Robotics Research*, 33:9, 2014. (p. 23)

[176] 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, 2007. (p. 10, 21, 22, 23)

[177] C.-C. Wang, C. Thorpe, and A. Suppe. LADAR-based detection and tracking of moving objects from a ground vehicle at high speeds. In *IEEE Intelligent Vehicles Symp.*, pages 416–421, June 2003. (p. 14, 79)

[178] Y. Wang, R. Xiong, Q. Li, and S. Huang. Kullback-leibler divergence based graph pruning in robotic feature mapping. In *Proceedings of the European Conference on Mobile Robotics*, pages 32–37, Barcelona, Spain, Sept. 2013. (p. 13, 16, 19, 21, 22)

[179] S. B. Williams, O. Pizarro, J. M. Webster, R. J. Beaman, I. Mahon, M. Johnson-Roberson, and T. C. L. Bridge. Autonomous underwater vehicle-assisted surveying of drowned reefs on the shelf edge of the Great Barrier Reef, Australia. *Journal of Field Robotics*, 27(5): 675–697, 2010. (p. 9)

[180] S. Winder, G. Hua, and M. Brown. Picking the best DAISY. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 178–185, Miami, FL, June 2009. (p. 14, 59, 60, 64, 65, 68)

[181] S. A. J. Winder and M. Brown. Learning local image descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, Minneapolis, MN, 2007. (p. 14, 59, 60, 64)

[182] R. W. Wolcott and R. M. Eustice. Visual localization within LIDAR maps for automated urban driving. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 176–183, Chicago, IL, USA, Sept. 2014. (p. 9)

[183] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller, E. Kaus, R. G. Herrtwich, C. Rabe, D. Pfeiffer, F. Lindner, F. Stein, F. Erbs, M. Enzweiler, C. Knoppel, J. Hipp, M. Haueis, M. Trepte, C. Brenk, A. Tamke, M. Ghanaat, M. Braun, A. Joos, H. Fritz, H. Mock, M. Hein, and E. Zeeb. Making Bertha drive—an autonomous journey on a historic route. *IEEE Intelligent Transportation Systems Magazine*, 6(2):8–20, 2014. (p. 9)

[184] W. Y. Zou, S. Zhu, A. Y. Ng, and K. Yu. Deep learning of invariant features via simulated fixations in video. In *Proceedings of the Advances in Neural Information Processing Systems Conference*, pages 3212–3220, Lake Tahoe, NV, Dec. 2012. (p. 64)