Belief-space Planning for Active Visual SLAM in Underwater Environments

by

Stephen M. Chaves

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Mechanical Engineering)
in the University of Michigan
2016

Doctoral Committee:

Associate Professor Ryan M. Eustice, Chair
Assistant Research Professor Michael Kaess, Carnegie Mellon University
Professor Satinder Singh (Baveja)
Assistant Professor Ram Vasudevan

Acknowledgments

This acknowledgments section should probably be the longest chapter in this document. There are so many people who deserve recognition and thanks for their hard work and sacrifice in making all of this possible.

First, I am very grateful to my advisor, Ryan. Somehow during my initial search for graduate schools, I stumbled into a chance meeting with Ryan and was launched into a very dynamic and fulfilling doctoral journey. Thank you, Ryan, for your thoughtful guidance and quiet leadership, and for all the amazing sushi. A special thank you to my committee members as well, who brought great perspective and insight to my research.

Thank you to all of my fellow members of the Perceptual Robotics Laboratory for the countless hours discussing ideas, developing theory, fielding robots, debugging code, waiting for DVL lock, lunch in Pierpont, and everything else. In order of appearance: Ayoung, Gaurav, Nick, Jeff, Paul, Wolcott, Schuyler, Jie, Arash, GT, Enric, Alex, Vozar, Josh, Sparki, Vittorio, and Derrick. There isn't a finer bunch of researchers around.

Thank you to my parents and my family, who have supported and encouraged me from the very beginning, even from the early days of finding a love of engineering while building with Legos. I love you all.

None of this would have been possible without the devotion and sacrifice of my very loving wife and best friend, Allison. Thank you for your steadfast support through this endeavor and for showing me the meaning of love. You are deserving of congratulations! And thank you to my adventurous son, Samson. You bring so much joy to my life. I love you both.

Finally and most importantly, I am forever thankful to my Lord Jesus Christ, who by grace saved my life through faith. I pray that this thesis is a reflection of you and not of me.

> *Be watchful, stand firm in the faith, act like men, be strong. Let all that you do be done in love.* 1 Corinthians 16:13-14

This work was supported by the Office of Naval Research under award N00014-12-1-0092 and by the SMART Scholarship for Service program.

Contents

List of Figures

List of Tables

List of Acronyms

AUV autonomous underwater vehicle

BoW bag-of-words

BRM Belief Roadmap

CLAHS contrast-limited adaptive histogram specification

CLT Chow-Liu tree

COLAMD column approximate minimum degree

CVaR conditional value-at-risk

DIDSON Dual frequency IDentification SONar

DOF degree of freedom

DVL Doppler velocity log

EM expectation-maximization

EKF extended Kalman filter

GLC generic linear constraint

GP Gaussian process

GTSAM Georgia Tech Smoothing and Mapping

HAUV Hovering Autonomous Underwater Vehicle

KLD Kullback-Leibler divergence

LQG linear quadratic Gaussian

LQR linear quadratic regulator

MAP maximum *a posteriori*

MDP Markov decision process

MHL Marine Hydrodynamics Laboratory

MPC model predictive control

MRF Markov random field

NBV next-best-view

OCE optimized certainty equivalent

ONR Office of Naval Research

PCCS pose-constrained correspondence search

PDN perception-driven navigation

pdf probability density function

POMDP partially observable Markov decision process

PRM Probabilistic Roadmap

RANSAC random sample consensus

RHC receding horizon control

RRG Rapidly-exploring Random Graph

RRT Rapidly-exploring Random Tree

RRT* asymptotically-optimal Rapidly-exploring Random Tree

RIG Rapidly-exploring Information Gathering

SIFT Scale Invariant Feature Transform

SLAM simultaneous localization and mapping

SURF Speeded Up Robust Features

T-RRT transition-based RRT

T-RRT* transition-based RRT*

UAV unmanned aerial vehicle

VaR value-at-risk

Abstract

Autonomous mobile robots operating in *a priori* unknown environments must be able to integrate path planning with simultaneous localization and mapping (SLAM) in order to perform tasks like exploration, search and rescue, inspection, reconnaissance, target-tracking, and others. This level of autonomy is especially difficult in underwater environments, where GPS is unavailable, communication is limited, and environment features may be sparsely-distributed. In these situations, the path taken by the robot can drastically affect the performance of SLAM, so the robot must plan and act intelligently and efficiently to ensure successful task completion.

This document proposes novel research in belief-space planning for active visual SLAM in underwater environments. Our motivating application is ship hull inspection with an autonomous underwater robot. We design a Gaussian belief-space planning formulation that accounts for the randomness of the loop-closure measurements in visual SLAM and serves as the mathematical foundation for the research in this thesis. Combining this planning formulation with sampling-based techniques, we efficiently search for loop-closure actions throughout the environment and present a two-step approach for selecting revisit actions that results in an opportunistic active SLAM framework. The proposed active SLAM method is tested in hybrid simulations and real-world field trials of an underwater robot performing inspections of a physical modeling basin and a U.S. Coast Guard cutter.

To reduce computational load, we present research into efficient planning by compressing the representation and examining the structure of the underlying SLAM system. We propose the use of graph sparsification methods online to reduce complexity by planning with an approximate distribution that represents the original, full pose graph. We also propose the use of the Bayes tree data structure—first introduced for fast inference in SLAM—to perform efficient incremental updates when evaluating candidate plans that are similar. As a final contribution, we design risk-averse objective functions that account for the randomness within our planning formulation. We show that this aversion to uncertainty in the posterior belief leads to desirable and intuitive behavior within active SLAM.

Chapter 1

Introduction

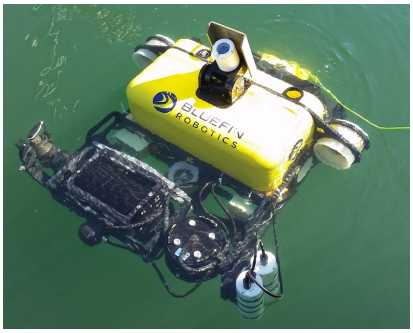
1.1 Motivation

Autonomous mobile robots operating in real-world environments must fulfill three core competencies: localization, mapping, and planning. Solutions to these problems are important in order to perform tasks like exploration, search and rescue, inspection, reconnaissance, target-tracking, and others [113]. Research in these areas has been applied to a wide range of mobile robotic platforms, including unmanned aerial vehicles (UAVs) [3, 22], self-driving cars [138, 139], and autonomous underwater vehicles (AUVs) [47, 133]. Under the topic of simultaneous localization and mapping (SLAM), localization and mapping have been well-studied in the past decades, resulting in formulations that allow robots to estimate their state and model their environment [5, 29, 36]. Robots must plan actions within the SLAM framework to be truly autonomous, but research toward integrating planning and SLAM solutions is still in its early years.

SLAM solutions are typically decoupled from the path or control policy given to the robot. Likewise, traditional planning algorithms assume that accurate localization is available and information about the environment is at least partially known [85]. Of course, real-world robots operate at the intersection of these topics where assumptions about the available prior information are not always true, posing an interesting challenge for robust autonomy.

Autonomy in marine environments can be especially difficult, but AUVs have been successfully deployed in many challenging environments [69, 102, 136, 137]. Underwater robots commonly operate in GPS-denied scenarios with limited communication, often over large unknown areas with sparsely-distributed features. In these situations, the path taken by the robot through the environment can drastically affect the performance of SLAM. Thus, real-world robotic systems demand a comprehensive, probabilistic framework for integrated localization, mapping, and planning.

Figure 1.1 The HAUV (a) used for autonomous hull inspection of ships like the *SS Curtiss* (b).



(a) HAUV

(b) *SS Curtiss*

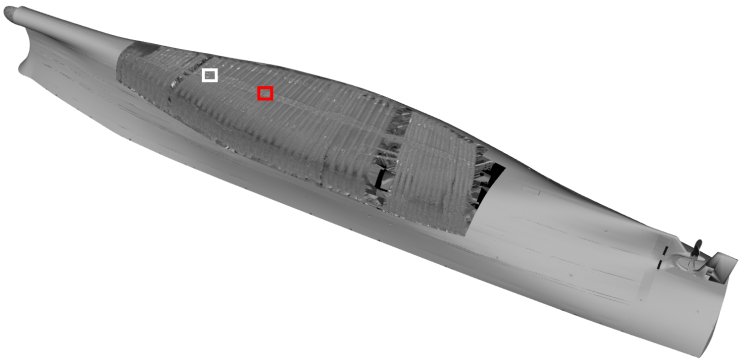
This document proposes advances in active SLAM frameworks for autonomous visual inspection of underwater environments with a mobile robot. The contributions of this work include: the development of a planning formulation for accurate prediction of the robot's belief, algorithms for finding candidate paths useful for closing loops in the SLAM system, efficient methods for planning by exploiting the representation of the problem, and a risk-averse optimization for producing desirable and intuitive behavior within active SLAM.

1.2 Review of Ship Hull Inspection with the HAUV

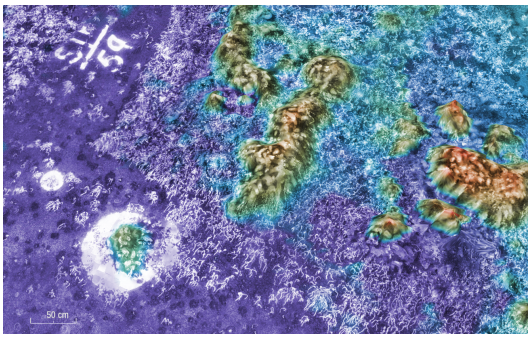
The main motivation for the work in this document is the application of autonomous ship hull inspection using a Hovering Autonomous Underwater Vehicle (HAUV), shown in Fig. 1.1(a). This application is the focus of a large joint research project between the University of Michigan, Massachusetts Institute of Technology, Carnegie Mellon University, and Bluefin Robotics [60]. The HAUV uses its perceptual sensors to survey the underwater portion of ships like the *SS Curtiss* (Fig. 1.1(b)) in order to detect foreign objects, verify structural integrity, identify corrosion, etc. These tasks are traditionally carried out by human divers, but the underwater environment can be very dangerous. Developing a robust robotic system for inspection removes humans from potential hazards and saves on expensive diver deployment and ship downtime costs. In addition, the rich metric models of the environment built by the robot are useful for tasks other than inspection and provide a nice visualization of the environment. See Fig. 1.2 for examples of visualizations derived from underwater imagery collected by the HAUV.

The HAUV was developed specifically for in-water autonomous ship hull inspection and features three aspects that set it apart from other AUVs: its small size, ability to hover while submerged, and hull-relative navigation capability [59]. The vehicle's main proprioceptive

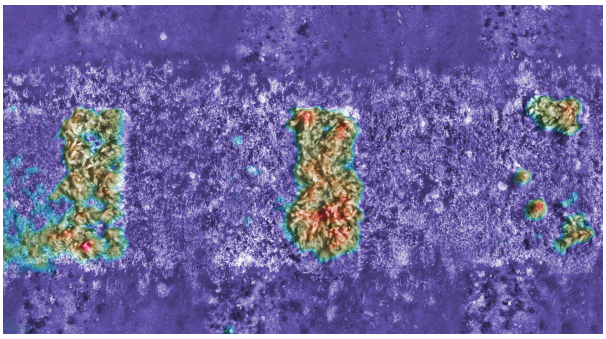
Figure 1.2 Figures courtesy of Ozog [100], Ozog and Eustice [101]. (a) Photomosaic augmented with 3D depth information derived from stereo camera imagery collected by the HAUV. (b) (c) Zoomed views show the rich visualizations possible as a result of autonomous ship hull inspection.



(a) 3D Photomosaic



(b)

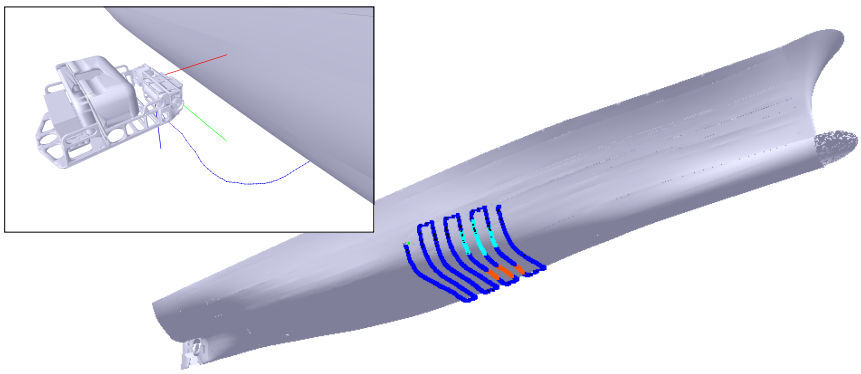


(c)

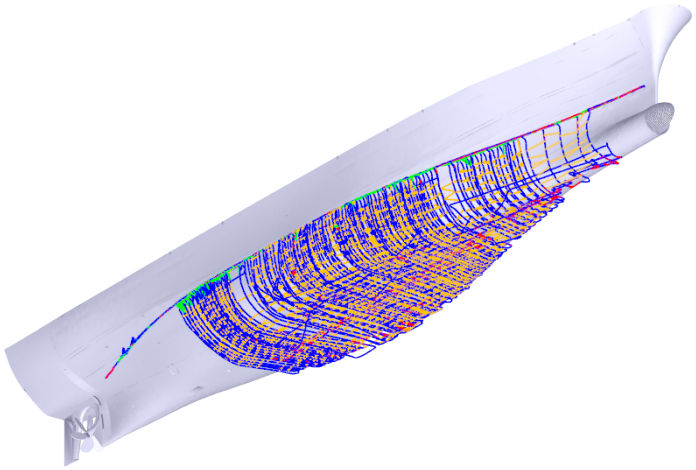
sensor is a Doppler velocity log (DVL), with which it performs dead-reckoned navigation via Doppler velocimetry [50] in lieu of traditional inertial methods [81] or approaches based on acoustic beacon networks [129]. Measurements from the DVL allow the robot to maintain a uniform standoff from the inspection surface and provide the ability to perform hull-relative navigation. The HAUV is also equipped with underwater cameras (stereo and monocular) and an imaging sonar (Dual frequency IDentification SONar (DIDSON)) to perceive the environment. The perceptual sensors and DVL are positioned on an actuated tray that rotates to always point normal to the inspection surface.

To survey the non-complex area of the ship, the robot performs visual SLAM while following a nominal exploration policy that produces a lawn-mower-like trajectory through the area of interest, visualized in Fig. 1.3. This SLAM problem has many challenges unique to confined underwater environments; for instance, GPS is unavailable underwater and communication is extremely limited without a tethered connection to the robot. In addition, camera registrations rely on visual features that tend to be sparsely-distributed and imagery is prone to suffer from backscatter in turbid water conditions. Yet, when visual registrations

Figure 1.3 (a) Visualization of the HAUV performing visual ship hull inspection following the nominal exploration policy. (b) Over multiple missions of long durations, the inspection provides dense coverage of the underwater portion of the hull.



(a)



(b)

can be made, the camera provides an accurate method for correcting drift in the SLAM estimate.

In typical operation, the robot navigates following the nominal exploration policy in an open-loop fashion. This type of trajectory leads to unbounded uncertainty growth in the SLAM estimate. The robot can constrain its uncertainty (and uncertainty in the map) by gathering *loop-closure* measurements that re-observe previously-seen portions of the environment. Efficiently seeking and executing large loop-closure actions is the basis of the work proposed in this thesis.

1.3 Review of SLAM

SLAM is a fundamental problem in robotics where a robot uses its sensors to collect noisy observations of its surroundings in order to estimate a map of its environment and localize itself within the map. Researchers in robotics have been studying SLAM for decades, such that there are well-known probabilistic tools for handling the sensor data and solving the underlying estimation problem [5, 29, 36]. In general, the SLAM problem can be divided into two sub-problems: (*i*) a "front-end" system that parses sensor measurements, and (*ii*) a "back-end" solver that optimizes over robot poses and map features.

Early approaches to the SLAM problem tracked the most recent robot pose and landmarks throughout the environment in a Kalman filter [87, 116]. Here, the SLAM estimate is represented as a multivariate Gaussian with mean vector and fully dense covariance matrix. Complexity of the Kalman filter, however, grows with the size of the map, as the measurement updates are cubic and memory requirements are quadratic in the state dimension. Thrun et al. [120] observed that the information matrix (inverse of the covariance matrix) of the estimate is approximately sparse, leading to more efficient solutions using an *information* filtering approach that forced sparsity. The information filtering approach features constant time measurement updates and linear memory requirements. Extending the seminal work of Lu and Milios [92], Eustice et al. [40] showed that by considering a delayed-state information filter, the information matrix of the SLAM problem is *exactly* sparse, leveraging the benefits of the information parameterization without sparse approximation errors. Most SLAM systems today formulate the problem in the exactly-sparse sense by optimizing over the entire robot trajectory.

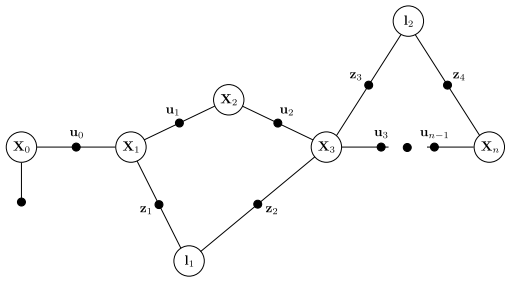
The *full* SLAM formulation considers optimizing over the entire history of robot poses and landmarks. This problem can be solved using the maximum *a posteriori* (MAP) estimate, given the prior observations of the robot motion and landmarks in the environment:

$$X^*, \mathcal{L}^* = \arg\max_{X,\mathcal{L}} p(X, \mathcal{L}|\mathcal{U}, \mathcal{Z}), \tag{1.1}$$

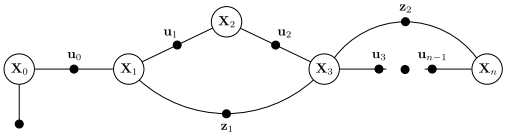
where $\mathbf{x}_i \in X$ are the robot poses, $\mathbf{l}_k \in \mathcal{L}$ are the landmark poses, $\mathbf{u}_i \in \mathcal{U}$ are the control inputs (or motion observations), and $\mathbf{z}_j \in \mathcal{Z}$ are the perceptual observations of map features. The full SLAM formulation is shown in Fig. 1.4(a) in the form of a factor graph.

The formulation considered in this work is *pose* SLAM (Fig. 1.4(b)), where there is no explicit representation of landmarks, but rather features observed in the environment are used to construct a relative measurement between robot poses [92]. In this case, the MAP

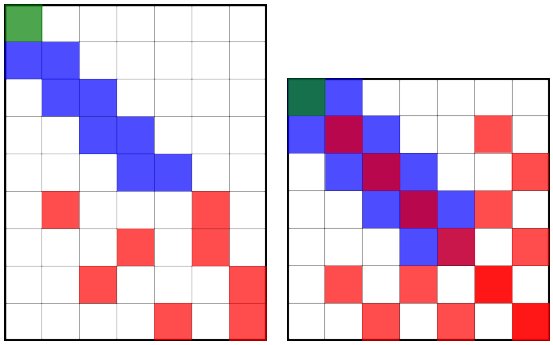
Figure 1.4 Factor graph representations of the full SLAM (a) and pose SLAM (b) formulations. The corresponding measurement Jacobian (A) and information matrix ($\Lambda = A^\top A$) for each system are shown below the factor graphs, with matrix block entries colored by factor type. Prior measurements (green), odometry measurements (blue), and loop-closures (red) are all represented. In the full SLAM system (a)(c)(d), loop-closures include measurements to landmarks. The rows of the measurement Jacobian A are ordered by prior measurement first, then odometry measurements, then loop-closures. The columns correspond to the following ordering: $\{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{l}_1, \mathbf{l}_2\}$. In the pose SLAM system (b)(e)(f), the columns of A correspond to an ordering of $\{\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\}$.



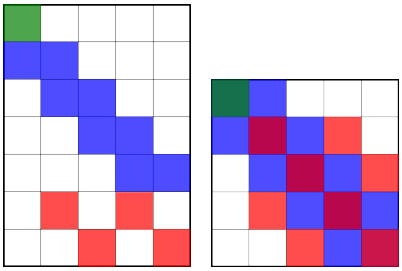
(a) Full SLAM



(b) Pose SLAM



(c) A (d) Λ



(e) A (f) Λ

estimate becomes

$$X^* = \arg\max_X p(X|\mathcal{U}, \mathcal{Z}), \tag{1.2}$$

and the model of the environment is derived from the robot trajectory itself. This formulation is especially beneficial when the main perceptual sensors are cameras or laser scanners and the environment features are difficult to repeatedly observe or are too numerous to track.

Recent research in SLAM has turned to optimization-based solutions in order to avoid the commitment to a static linearization point associated with filtering-based methods [29, 73]. Assuming measurement models with additive Gaussian noise, these methods formulate the

optimization of (1.1) or (1.2) as a nonlinear least-squares problem:

$$\begin{aligned} X^*, \mathcal{L}^* &= \arg\max_{X,\mathcal{L}} p(X, \mathcal{L}|\mathcal{U}, \mathcal{Z}) = \arg\min_{X,\mathcal{L}} \; -\log p(X, \mathcal{L}|\mathcal{U}, \mathcal{Z}) \\ &= \arg\min_{X,\mathcal{L}} \Big\{ \sum_i \|\mathbf{x}_i - f_i(\mathbf{x}_{i-1}, \mathbf{u}_{i-1})\|^2_{\Omega^i_w} + \sum_j \|\mathbf{z}_j - h_j(\mathbf{x}_{i_j}, \mathbf{l}_{k_j})\|^2_{\Omega^j_v} \Big\}, \end{aligned} \tag{1.3}$$

where f_i and h_j are the measurement models with zero-mean additive Gaussian noise with covariances ${\Omega^i_w}^{-1}$ and ${\Omega^j_v}^{-1}$, and we define $\|e\|^2_\Omega = e^\top \Omega e$, where Ω is the inverse covariance matrix (i.e., information matrix). Linearizing about the current estimate, the problem (1.3) collapses into a linear least-squares form for the state update vector, solved with the standard normal equations:

$$\begin{aligned} &\arg\min_{\Delta\Theta} \|A\Delta\Theta - \mathbf{b}\|^2, \\ &\Delta\Theta = \left(A^\top A\right)^{-1} A^\top \mathbf{b}, \end{aligned} \tag{1.4}$$

where the vector Θ includes the poses and landmarks, A is the stacked whitened measurement Jacobian, and $\mathbf{b}$ is the corresponding residual vector. Under the assumption of independent measurements, this formulation leads to an information matrix ($\Lambda = A^\top A$) that is inherently sparse, as each observation model depends on only a small subset of poses and landmarks. Thus, modern back-end solvers leverage sparsity patterns to efficiently find solutions.

We solve the nonlinear problem by re-linearizing about the new solution and solving again, repeating until convergence (with Gauss-Newton, for instance). Each linear problem is most commonly solved by direct methods like Cholesky decomposition of the information matrix or QR factorization of the measurement Jacobian [29, 73]. Aside from direct methods, iterative methods like relaxation-based techniques [35] and conjugate gradients [30, 119] have also been applied to solve large linear systems in a more memory-efficient and parallelizable way.

1.3.1 Graphical Representations

A common representation of SLAM in a graphical form is the *factor graph*. A factor graph is a bipartite graph with two types of components: nodes that represent variables and factors that represent constraint potentials over the variables. The factor graph for the SLAM problem consists of nodes representing robot and landmarks poses, and factors representing measurements, as seen in Fig. 1.4. If each measurement is encoded in a factor, $\psi_i(\mathbf{x}_i, \mathbf{l}_i)$, where $\mathbf{x}_i$ and $\mathbf{l}_i$ are the robot and landmark poses corresponding to measurement i (and we assume all measurements are independent), the nonlinear least-squares problem can be written as

$$X^*, \mathcal{L}^* = \arg\min_{X,\mathcal{L}} \sum_i \psi_i(\mathbf{x}_i, \mathbf{l}_i), \tag{1.5}$$

such that the optimization minimizes the sum of squared errors of all the factor potentials. The factor graph form directly corresponds to the structure of the measurement Jacobian A. The information matrix $\Lambda = A^\top A$ corresponds to another graphical model, the Markov random field (MRF).

We will see that representing the SLAM problem in graphical form is very useful for designing algorithms, even in the planning domain. We refer to the factor graph frequently throughout the rest of this document.

1.3.2 Underwater Visual SLAM

Cameras are prevalent perceptual sensors in robotics research because of their low cost but highly accurate and rich data. Their popularity has led to research in visual SLAM, where measurements derived from the camera are included in the inference. Within the pose SLAM formulation of the underwater inspection application, feature-based registrations between overlapping images [54] produce pairwise constraints between robot poses. A typical pairwise registration pipeline is shown in Fig. 1.5 and is described as follows:

1. Given two overlapping images collected by the robot, first perform radial undistortion on each image and enhance with contrast-limited adaptive histogram specification (CLAHS) [39].

2. Extract features such as Scale Invariant Feature Transform (SIFT) [91] or Speeded Up Robust Features (SURF) [8] from each image.

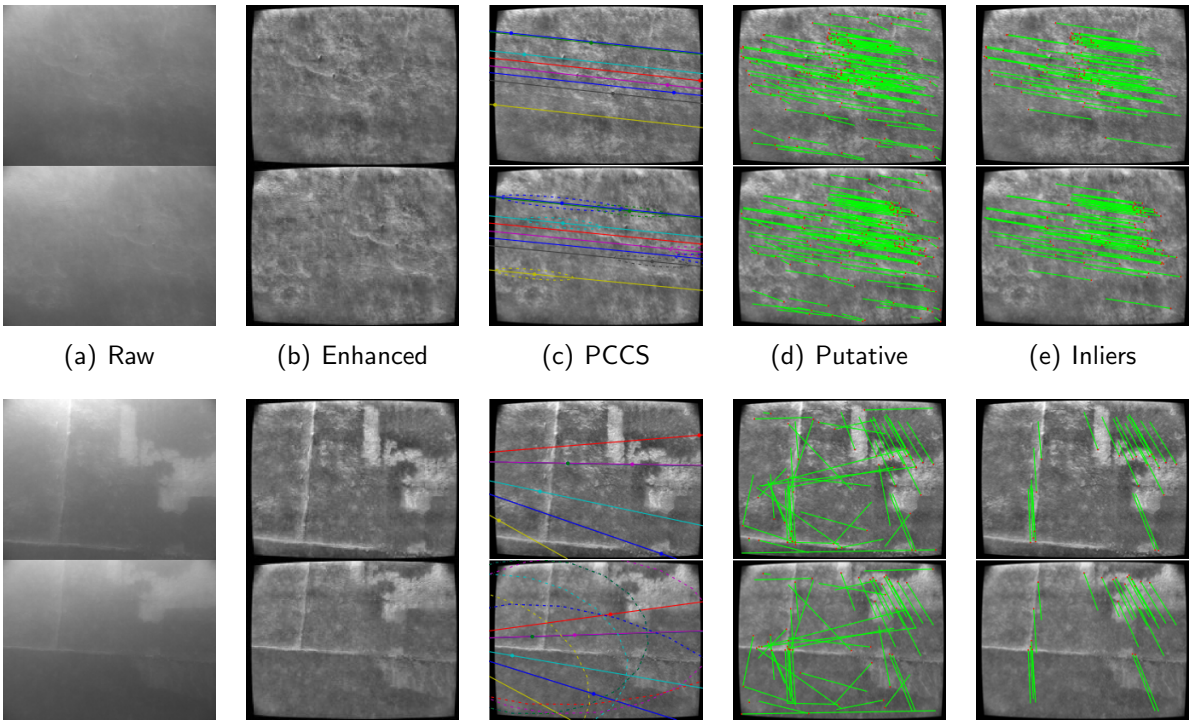
3. Match features between the images using a nearest-neighbors search in the high-dimensional feature space assisted by pose-constrained correspondence search (PCCS) [42].

4. Fit a projective model among feature matching inliers using a geometric consensus algorithm such as random sample consensus (RANSAC) [44].

5. Perform a two-view bundle adjustment problem to solve for the 5-degree of freedom (DOF) bearing-only transformation between camera poses and first-order covariance estimate [52].

The 5-DOF camera measurement produces a low-rank (modulo scale) relative-pose constraint between two robot poses in the SLAM graph. In visual SLAM, these measurements serve as loop-closure constraints.

Figure 1.5 Figures courtesy of Kim and Eustice [79]. Underwater visual SLAM: The pairwise image registration pipeline is shown for two registration hypotheses. The top row shows a feature-poor image set that registers successfully because of strong relative constraints between poses that guide feature-matching (PCCS). The bottom row is also a successful registration, but largely due to the strong features in the images. Steps in the registration pipeline are shown from left to right: (a)(f) Raw overlapping images. (b)(g) Undistorted and enhanced, before extracting features. (c)(h) Feature matching is guided by PCCS. (d)(i) Putative correspondences. (e)(j) Geometric consensus is used to identify inliers. Finally, a two-view bundle adjustment solves for the 5-DOF relative pose constraint.



(a) Raw (b) Enhanced (c) PCCS (d) Putative (e) Inliers

(f) Raw (g) Enhanced (h) PCCS (i) Putative (j) Inliers

1.4 Review of Robot Planning

The SLAM formulation assumes control actions are inputs to the system, such that the inference process is decoupled from any notion of path planning. But, it is well-known that the performance of SLAM is greatly influenced by the robot's trajectory through the environment; the path directly affects the accuracy of localization, the quality of the resulting map, the convergence of the solution, the rate of area coverage, and the ability to execute tasks. This calls for research toward the integration of planning and SLAM into a comprehensive system for robust autonomy.

1.4.1 Deterministic Planning

Traditional path planning for robots considers the problem of finding a collision-free path from a start pose to a goal pose given known state and known environment. There are two concerns with this planning problem: *feasibility* ("does a path exist?") and *optimality* ("what is the best path?"). Early approaches for solving these point-to-point queries discretized the free space into grid cells and applied forward search algorithms [85]. Systematic search algorithms like breadth-first and depth-first search are *complete*, and thus will find a solution if one exists. In the case of a performance criterion like shortest distance, search strategies like Dijkstra's [32] or A* [53] are preferred, as these methods are both complete and *optimal*. Alternatively, value iteration can also be applied to find optimal plans.

Discrete planning has its disadvantages. Notably, discrete planners are only complete (and optimal) up to grid resolution and thus have trouble expanding to continuous and high-dimensional spaces. In addition, it may be difficult to explicitly represent the free space in the environment. In response, sampling-based planning has been very successful in addressing these issues. Sampling-based algorithms provide a stochastic discretization of high-dimensional continuous spaces [24, 85]. The general idea behind hallmark algorithms like the Probabilistic Roadmap (PRM) [76] or Rapidly-exploring Random Tree (RRT) [86] is to sample robot configurations, check for collisions, and connect collision-free paths between samples. These algorithms are efficient for high-dimensional spaces, relying on the concept that collision-checking a sampled point is generally easier than explicitly representing free space. As the number of samples tends to infinity, these planners are guaranteed to find a solution if it exists. Therefore, they are *probabilistically complete*. Recent extensions to sampling-based planning have provided *asymptotic optimality* guarantees by connecting samples intelligently [75].

1.4.2 Planning under Uncertainty

Unfortunately, all of the above planning algorithms assume deterministic (known) information about the robot's state and environment. Real-world robotic systems must make decisions with imperfect state information by planning in the robot's *belief-space*, the space of probability distributions of the state space. Generally, three sources of uncertainty must be considered during planning:

- *motion uncertainty*, where the control input applied to the robot results in a stochastic action step,
- *sensing uncertainty*, where the robot receives noisy and incomplete observations of its

state and environment from its sensors, and

- *environment uncertainty*, where the robot may be operating in an *a priori* unknown or partially-known environment.

As a result, the planning problem is an instance of a *partially-observable Markov decision process* (POMDP), a structure of problem for planning actions when the state is not directly observable. The objective is to compute a policy that maps the belief space to the space of control actions, but in general the solution to this problem is intractable due to the extreme dimensionality of the underlying system [72]. Thus, much research in planning under uncertainty has focused on approximations to the POMDP in order to plan for robotic tasks. A thorough investigation of current research in belief-space planning in given in Chapter 2.

Active SLAM

The topic of *active* SLAM can be defined as a planning under uncertainty problem with a goal of reducing uncertainty or maximizing information in the SLAM posterior. It generally consists of three components: (*i*) searching for candidate control actions, (*ii*) defining a multi-objective performance criterion to evaluate desirable outcomes, and (*iii*) optimizing this criterion to select a set of control actions [15, 16, 80]. Carrillo et al. [16] defined the general form of the performance criterion as an objective function that trades off costs related to uncertainty in the parameters of the SLAM system and costs related to expected control effort (i.e., path length, energy consumption, operation time, etc.).

In the robotics literature, the integration of planning with SLAM has its roots in active exploration [49, 114, 117], where research focused on reducing uncertainty in the map representation. More recent work examined path planning for information gathering [57], a related problem to active SLAM formulated as maximizing information subject to budget constraints. Active SLAM systems like those of Chaves et al. [20], Kim and Eustice [80], Valencia et al. [122] focused more on the interaction between planning and SLAM, and how the performance and efficiency of SLAM is improved with intelligent decisions regarding which paths to travel. A more detailed literature review on active SLAM is presented in Chapter 3.

1.5 Thesis Outline

This work addresses the following problems common to underwater visual SLAM:

1. Following an open-loop nominal exploration policy, the robot accumulates navigation error and its uncertainty grows unbounded.

2. The inference problem (both for SLAM and planning) becomes increasingly expensive in terms of computation as the dimensionality of the system grows in spatial extent and duration.

3. Visual features in the underwater environment are sparsely-distributed, such that much of the imagery collected by the robot has little to no registrability. In addition, at planning time it is unknown which camera registrations will be acquired.

To address the problems above, we propose the following contributions within the development of an active SLAM system for underwater environments[1]:

1. We design a planning formulation for accurate prediction that accounts for the stochasticity of the camera measurements and introduce random variables describing their acquisition. We employ Gaussian process (GP) regression to probabilistically model the visual registrability of the environment and derive the associated acquisition parameters.

2. We combine sampling-based techniques with our planning formulation to efficiently search for loop-closure revisit paths throughout the environment. We present a two-step approach for deciding when to close loops that results in an opportunistic active SLAM framework.

3. We propose research into efficient planning by compressing the representation and examining the structure of the underlying SLAM system. We propose the use of graph sparsification methods online to reduce computational complexity by planning with a sparsified approximate distribution in place of the full system. We also propose the use of the Bayes tree data structure within active SLAM in order to perform fast incremental updates when evaluating candidate plans that are very similar to each other.

4. We design risk-averse objective functions for selecting revisit paths that account for the randomness within our planning formulation. We show that this aversion to uncertainty in the formulation leads to desirable and intuitive behavior within active SLAM. Our optimization is built on an analytic approach to computing the posterior belief distribution.

1.5.1 Document Roadmap

The contributions are detailed in the following chapters:

[1] Portions of these contributions previously appeared in Chaves and Eustice [19], Chaves et al. [20, 21, 23].

Chapter 2 We propose a belief-space planning formulation considering the application of active visual SLAM. We derive the resulting belief as a function of the random measurements and their associated random acquisition variables, making the belief itself also random. This formulation serves as the mathematical foundation for the remaining chapters.

Chapter 3 We propose an approach to the active visual SLAM problem by combining the planning formulation of Chapter 2 with sampling-based techniques. To survey the target environment efficiently with bounded uncertainty, we design a path selection process that is opportunistic. The performance of the full active SLAM framework for autonomous ship hull inspection is tested in hybrid simulations and real-world field trials.

Chapter 4 We investigate two concepts related to improving the efficiency of information-theoretic planning evaluations. First, we reduce the dimensionality of the system by compressing the representation with online graph sparsification. Second, we leverage the structure in the problem by using the Bayes tree to efficiently evaluate candidate plans.

Chapter 5 Inspired by expected utility theory, we design risk-averse objective functions with respect to robot uncertainty. We compute the posterior belief distribution with an analytic approach and present simulation results that are in accordance with rational decision-making behavior.

Chapter 6 We conclude by summarizing the contributions of this thesis and suggesting areas of future work.

Appendix A We provide a brief overview of the HAUV.

Chapter 2

Belief-space Planning Formulation for Visual SLAM

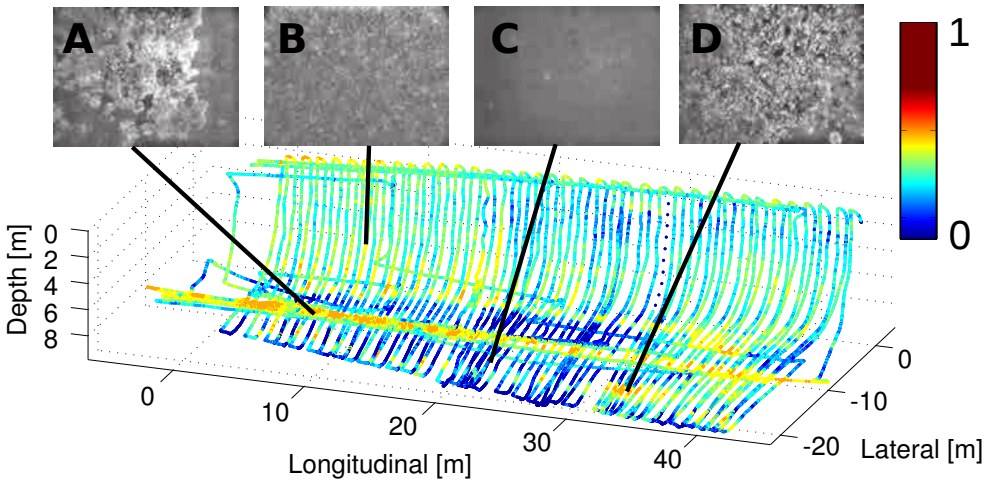
2.1 Introduction

Planning for probabilistic robotics can be formalized as a partially observable Markov decision process (POMDP) [121]. Solving real-world POMDPs is generally intractable however, so we must make some approximations in order to perform online decision-making for robotic tasks. In this chapter, we present our framework for *Gaussian belief-space planning* as an approximation to the underlying POMDP. We represent the distribution over robot states as a Gaussian belief, which we track using Bayesian inference (i.e., solving the simultaneous localization and mapping (SLAM) problem). By predicting the evolution of the belief into the future, we can perform an optimization to select a set of control actions for the planning problem of interest.

We formulate our belief-space planning framework considering its use with underwater visual SLAM. As the robot moves through the environment, the SLAM system tracks its belief by incorporating successful measurements when they are received. However at planning time, it is unknown what the values of future measurements will be. Therefore, in order to accurately model the prediction of the belief, we treat future measurements as *random* [126], and derive the predicted belief as a function of the random measurements.

In addition, we propose the inclusion of stochastic measurement acquisition variables into the planning formulation. The acquisition variables are Bernoulli random variables that model whether or not each future measurement is actually acquired. For example, an acquisition variable might describe whether a landmark is in the field of view of the robot, whether two camera images overlap and are registered, or whether a message is received over

Figure 2.1 Representative sample images captured on the underwater portion of the *SS Curtiss* hull. The underwater feature distribution is highly diverse, ranging from feature-less (B,C) to feature-rich (A,D) images. The robot poses are color-coded by their local saliency score, S_L.

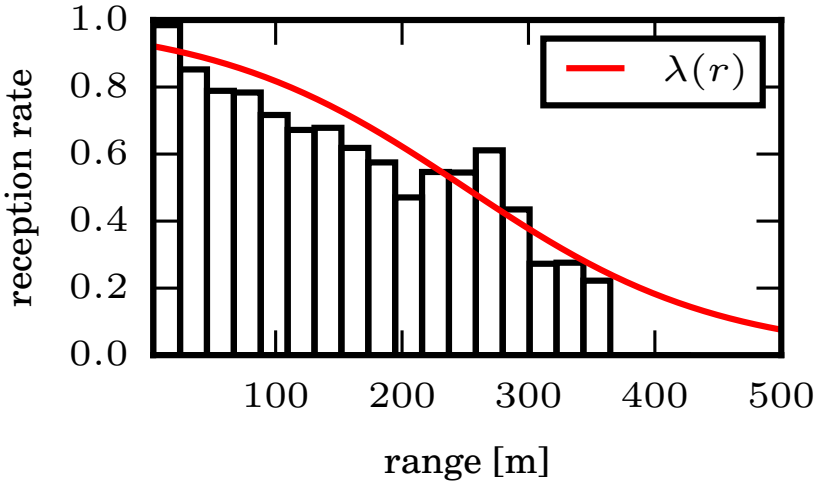


a lossy communication channel. Consider our motivating application of ship hull inspection; visual features in this underwater environment tend to be sparsely distributed such that much of the imagery collected by the robot is not useful for registration (Fig. 2.1). The probability of a successful two-view registration (i.e., probability of acquisition) is directly related to a measure of visual saliency of the proposed camera images [79]. Similarly, the probability of successful packet reception over an underwater acoustic channel is directly related to the transmission range between the two modems [134], as shown in Fig. 2.2. In both of these examples, accurately predicting future measurements requires accurately modeling whether the measurements will be acquired. Like the measurements themselves, these acquisition variables are also unknown at the time of planning, and therefore random.

This chapter presents our belief-space planning formulation that includes the random measurements and random acquisition variables. We show that including the randomness in the prediction leads to a belief that is a "distribution of distributions" [55]. In other words, our formulation results in a distribution of beliefs, where a belief itself describes a distribution over robot states.

We also show in this chapter how the planning formulation fits with the visual SLAM framework for our application of ship hull inspection. We propose the use of Gaussian process (GP) regression to model the distribution of visual saliency throughout the environment, hence allowing us to accurately predict future measurements for planning. The formulation in this chapter serves as the mathematical foundation for the rest of the work in this document.

Figure 2.2 Figure from Walls et al. [134]. The probability of successful packet reception ($\lambda(r)$) for underwater acoustic transmission is directly related to the broadcast range. Acoustic signals are often used for communication between multiple underwater vehicles and a topside support ship.



2.1.1 Related Work

The planning problem for probabilistic robotics is formally defined as a POMDP, where the goal is to compute a control policy mapping belief states to actions [72]. It is well-known that globally-optimal solutions to this problem are very difficult to compute and generally intractable for problems of interesting size [103]. As a result, the planning under uncertainty community has focused on finding approximations in order to handle real-world robotic applications.

Point-based algorithms for POMDPs sample representative points in belief space and iteratively apply value updates to compute a policy [4, 84, 108]. Without restricting the representation of the belief, Bai et al. [4] used Monte Carlo sampling [118] to handle continuous state and observation spaces in POMDPs and computed a global "approximately optimal" policy offline. Porta et al. [108] parameterized the belief as a mixture of Gaussians or a set of particles and applied a point-based solver.

A common trajectory optimization approach to belief-space planning is to represent the belief as a Gaussian distribution, track its evolution (the *belief dynamics*) with an extended Kalman filter (EKF), and use dynamic programming to compute a policy. Platt et al. [106] performed Gaussian belief-space planning by assuming maximum-likelihood observations from future control actions, allowing direct application of optimal control techniques like linear quadratic regulator (LQR) to compute a policy. Erez and Smart [38] and Du Toit and Burdick [34] offered a similar treatment of the problem, with the latter providing a receding horizon control (RHC) strategy for planning. Van den Berg et al. [126] removed the

maximum-likelihood observation assumption and computed a locally-optimal policy using iterative linear quadratic Gaussian (LQG) by approximating the value function around a nominal trajectory, effectively focusing the solution to relevant regions of the belief space. Combining policy computation with sampling-based planning, Agha-mohammadi et al. [2] computed feedback controllers over edges in a roadmap using stationary LQG.

Rather than compute a policy, other trajectory optimization methods directly optimize a sequence of control actions in a locally-optimal framework [67, 88, 104, 105, 134]. Walls et al. [134] assembled the EKF equations into a batch computation for the belief dynamics over a horizon of control actions and performed a parameterized optimization. Indelman et al. [67] used gradient descent and model predictive control (MPC) to continuously recompute a trajectory given a belief calculated in the information-smoothing form. We additionally use the information-smoothing approach in our formulation, modeled after common graph-based SLAM solutions. Also like that of Indelman et al. [67] and Leung et al. [88], we consider a *shooting* approach, where we optimize over control actions only. This is in contrast to partial or full collocation for direct trajectory optimization, where the belief is explicitly included in the decision variables of the optimization [104, 105].

Sampling-based or simulation-based belief-space approaches select the sampled sequence of actions (or trajectory) to perform the optimization, trading off a discretization of the action space in order to search for globally-optimal trajectories [13, 55, 57, 80, 109, 124]. The formulation presented in this chapter is applicable to sampling-based or direct trajectory optimization approaches alike, which we highlight in later chapters.

Regarding measurement acquisition in planning, Vitus and Tomlin [130] optimized sensor placement in an environment given a weighting function defining the acquisition of the sensor observation by a traveling vehicle. They use a sigmoid to approximate the weighting function and provide gradient information to inform the optimization, an approach borrowed by Patil et al. [104]. These methods describe the spatial distribution related to measurement acquisition but do not model the *randomness* in the acquisition. Our approach is similar to the formulation of Sinopoli et al. [115], who studied Kalman filtering for control given measurements over a lossy communication channel. They derived the Kalman filter equations as functions of the stochastic acquisition variables. Bopardikar et al. [11] used this filtering formulation to compute a bound on error covariance propagation given sensor misdetections, and planned with this bound over a Belief Roadmap (BRM) [109]. Kim and Eustice [80] and Indelman et al. [67] included acquisition variables in belief-space planning for robotics given the full information-smoothing form, but their formulations removed the effect of the acquisition randomness in the resulting belief. Indelman et al. [67] accomplished this by using expectation-maximization (EM) in the belief update.

We design a planning formulation that integrates with the underwater visual SLAM framework of Kim and Eustice [79]. We propose the use of GP regression to predict the visual saliency throughout the underwater environment, in order to model the probability of acquisition for hypothesized pairwise camera registrations [110]. Recently, GPs have been widely used in learning spatial distributions and predicting sensor data. Vasudevan et al. [127] applied GP regression to model large and complex terrain maps. O'Callaghan and Ramos [98] accurately predict an occupancy map using a GP with a trained neural network kernel, despite dealing with noisy and sparse sensor information. Within marine environments, Barkby et al. [7] used a GP to predict bathymetric data in unmapped patches in order to perform SLAM without any actual sensor overlap. They achieve tractable prediction with the GP over a large dataset by using a sparse covariance function [95].

2.2 Gaussian Belief-space Planning

Here we develop a planning formulation for active SLAM. Solving the full POMDP is intractable, so we make an approximation to allow the belief to be represented as a parameterized distribution. In this case, as is common in probabilistic inference for robotics, we represent the belief with a Gaussian distribution and track the evolution of the distribution with Bayesian inference, i.e. our visual SLAM formulation. This approximation effectively reduces the POMDP to a Markov decision process (MDP) in *belief space*. We can choose optimal actions for the MDP according to the most likely state in the belief distribution (the belief mean) and an estimate of the available information (the belief covariance). This treatment of the planning problem is often referred to as *Gaussian belief-space planning* [106].

2.2.1 Optimal Planning Problem

The optimal planning problem consists of finding a sequence of control actions, $U_{0:K-1} = \{\mathbf{u}_k\}_{k=0}^{K-1}$, over a horizon of K planning steps that minimizes some objective function of the robot's belief over this horizon, formulated as

$$U^*_{0:K-1} = \underset{U_{0:K-1}}{\arg\min}\ J\left(\mathcal{B}_{0:K}, U_{0:K-1}\right), \tag{2.1}$$

where the set of beliefs $\mathcal{B}_{0:K} = \{\mathcal{B}_k\}_{k=0}^{K}$ contains the belief at each planning step k. Contrary to traditional planning in robotics where the optimization considers some deterministic state, we replace the state with the Gaussian belief distribution. The planning problem is concerned with finding future control actions for the robot to execute, so we must propagate the belief

forward in time according to the sequence of controls considered. (The sequence of control actions over the horizon of K steps is often called a *macro-action* [55].)

The objective function is comprised of stage costs and a final cost dependent on the predicted belief of the robot at each planning step:

$$J\left(\mathcal{B}_{0:K}, U_{0:K-1}\right) = \sum_{k=0}^{K-1} c_k(\mathcal{B}_k, \mathbf{u}_k) + c_K(\mathcal{B}_K). \tag{2.2}$$

Previous works solved for the control actions in a trajectory-smoothing optimization, such as gradient descent [67] or dynamic programming [126]. Alternatively, we can perform this optimization within a sampling-based planning framework by selecting the sampled set of control actions that minimizes the objective [13]. We will return to the discussion of the optimal planning problem after examining the belief of the robot and its evolution.

2.2.2 Belief Inference

At planning time, the belief of the robot is estimated from the visual SLAM formulation, $\mathcal{B}_0 = p(X_0|\mathcal{Z}_0, \mathcal{U}_0)$, and represented as a multivariate Gaussian: $\mathcal{B}_0 \sim \mathcal{N}\left({X_0}^*, {\Lambda_0}^{-1}\right)$. X_0 is the trajectory of discrete robot poses up to planning time. $\mathcal{Z}_0$ and $\mathcal{U}_0$ denote the history of measurements and controls, respectively.

To predict the belief forward in time, we must incorporate the measurements and controls related to the sequence of actions for the planning event. Thus, we define the belief at a given planning step $k \in [1, K]$ as

$$\mathcal{B}_k = p(X_k|\mathcal{Z}_0, \mathcal{U}_0, Z_{1:k}, U_{0:k-1}), \tag{2.3}$$

where $X_k = X_0 \cup \{\mathbf{x}_i\}_{i=1}^k$ is the trajectory of the robot that includes both real (historical) poses (X_0) and virtual, predicted (future) poses ($\{\mathbf{x}_i\}_{i=1}^k$). The predicted measurements are represented by the set $Z_{1:k}$. The belief distribution is represented by a multivariate Gaussian with mean and covariance matrix (given in terms of the inverse information matrix):

$$\mathcal{B}_k \sim \mathcal{N}\left({X_k}^*, {\Lambda_k}^{-1}\right), \tag{2.4}$$

found using the maximum *a posteriori* (MAP) estimate

$${X_k}^* = \arg\min_{X_k} -\log \mathcal{B}_k. \tag{2.5}$$

The motion model for transitioning to step $k+1$ from step k is

$$\begin{aligned} \mathbf{x}_{k+1} &= f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k), \\ \mathbf{w}_k &\sim \mathcal{N}\left(\mathbf{0}, {\Omega_w}^{-1}\right), \end{aligned} \tag{2.6}$$

where $\mathbf{w}_k$ is zero-mean additive Gaussian noise with covariance ${\Omega_w}^{-1}$. Equivalently, the transition probability is expressed as $p(\mathbf{x}_{k+1}|\mathbf{x}_k, \mathbf{u}_k) \sim \mathcal{N}\left(f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{0}), {\Omega_w}^{-1}\right)$. There may be multiple (n_k) measurements associated with each planning step, such that the set of observations at a single step k is $Z_k = \{\mathbf{z}_{k,j}\}_{j=1}^{n_k}$. The observation model for the random measurement j at planning step k is

$$\begin{aligned} \mathbf{z}_{k,j} &= h\left(X_k^j, \mathbf{v}_{k,j}\right), \\ \mathbf{v}_{k,j} &\sim \mathcal{N}\left(\mathbf{0}, {\Omega_v^{k,j}}^{-1}\right), \end{aligned} \tag{2.7}$$

where $\mathbf{v}_{k,j}$ is also zero-mean additive Gaussian noise with covariance ${\Omega_v^{k,j}}^{-1}$, and X_k^j are the state variables of interest involved in the measurement. The measurement likelihood function is then $p\left(\mathbf{z}_{k,j}|X_k^j\right) \sim \mathcal{N}\left(h\left(X_k^j, \mathbf{0}\right), {\Omega_v^{k,j}}^{-1}\right)$.

Using the standard assumption of an uninformative prior on the measurements, we can write the belief distribution from (2.3) as

$$\begin{aligned} &p(X_k|\mathcal{Z}_0, \mathcal{U}_0, Z_{1:k}, U_{0:k-1}) \propto \\ &p(X_0|\mathcal{Z}_0, \mathcal{U}_0) \prod_{i=1}^{k} p(\mathbf{x}_i|\mathbf{x}_{i-1}, \mathbf{u}_{i-1}) \prod_{j=1}^{n_i} p\left(\mathbf{z}_{i,j}|X_i^j\right). \end{aligned} \tag{2.8}$$

Inserting the Gaussian motion and observation models into the MAP estimate of (2.5) and (2.8), we arrive at a nonlinear least-squares problem common to graph-based SLAM [29]:

$$\begin{aligned} {X_k}^* = \arg\min_{X_k} \Big[&\|X_0 - {X_0}^*\|_{\Lambda_0}^2 + \sum_{i=1}^{k} \|\mathbf{x}_i - f(\mathbf{x}_{i-1}, \mathbf{u}_{i-1}, \mathbf{0})\|_{\Omega_w}^2 + \\ &\sum_{i=1}^{k} \sum_{j=1}^{n_i} \left\|\mathbf{z}_{i,j} - h\left(X_i^j, \mathbf{0}\right)\right\|_{\Omega_v^{i,j}}^2 \Big], \end{aligned} \tag{2.9}$$

where the $\mathbf{z}_{i,j}$ are random because they are unknown at planning time. Here, we use the notation $\|\mathbf{e}\|_\Omega^2 = \mathbf{e}^\top \Omega \mathbf{e}$. We can compute a linearization point for the problem by compounding the given set of controls, yielding the nominal mean estimate $\bar{X}_k(U_{0:k-1}) = \{{X_0}^*, \bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_k\}$. Linearizing about this nominal mean estimate, the problem collapses into the following

representation for the state update vector ΔX_k:

$$\left\| A_k(U_{0:k-1})\Delta X_k - \mathbf{b}_k(Z_{1:k}, U_{0:k-1}) \right\|^2, \tag{2.10}$$

where

$$A_k = \begin{bmatrix} \begin{bmatrix} \Lambda_0^{\frac{1}{2}} & 0 \end{bmatrix} \\ \mathcal{D}(\Omega_w)^{\frac{1}{2}} F_k \\ \mathcal{D}\left(\Omega_v^{i,j}\right)^{\frac{1}{2}} H_k \end{bmatrix}, \quad \mathbf{b}_k = \begin{pmatrix} \mathbf{0} \\ \mathcal{D}(\Omega_w)^{\frac{1}{2}} \mathbf{b}_k^f \\ \mathcal{D}\left(\Omega_v^{i,j}\right)^{\frac{1}{2}} \mathbf{b}_k^h \end{pmatrix}. \tag{2.11}$$

Here, $\mathcal{D}(\,\cdot\,)$ denotes a diagonal matrix with the specified elements, F_k and H_k are the sparse Jacobians from the motion and observation models, respectively, and $\mathbf{b}_k^f$ and $\mathbf{b}_k^h$ are the corresponding residual vectors with stacked elements

$$\mathbf{b}_i^f = f(\bar{\mathbf{x}}_{i-1}, \mathbf{u}_{i-1}, \mathbf{0}) - \bar{\mathbf{x}}_i, \tag{2.12}$$

$$\mathbf{b}_{i,j}^h = \mathbf{z}_{i,j} - h\left(\bar{X}_i^j, \mathbf{0}\right). \tag{2.13}$$

Recall that the $\mathbf{z}_{i,j}$ within $\mathbf{b}_{i,j}^h$ are unknown.

Solving (2.10) around the linearization point $\bar{X}_k$, we find the update vector as a function of the random measurement variables:

$$\Delta X_k(Z_{1:k}, U_{0:k-1}) = \left(A_k^\top A_k\right)^{-1} A_k^\top \mathbf{b}_k. \tag{2.14}$$

Thus, the belief at planning step k is represented by the mean vector ${X_k}^* = \bar{X}_k + \Delta X_k$ and the associated information matrix,

$$\Lambda_k(U_{0:k-1}) = A_k^\top A_k. \tag{2.15}$$

The information matrix (and hence covariance) is not a function of the random measurement values, only the information contributed by the measurements.

Maximum-likelihood Observations versus Random Observations

Many previous works considered maximum-likelihood observations in place of the random observations we consider above in order to directly apply optimal control techniques to the resulting belief [34, 38, 106]. Under the maximum-likelihood assumption, the measurement residual term vanishes: $\mathbf{b}_{i,j}^h = \mathbf{z}_{i,j} - h\left(\bar{X}_i^j, \mathbf{0}\right) = \mathbf{0}$. In turn, the update vector equals zero, $\Delta X_k = \mathbf{0}$, such that the mean vector is equivalent to the nominal estimate, ${X_k}^* = \bar{X}_k$. In this way, the belief mean is *deterministic*, as the maximum-likelihood assumption removes

the randomness from the formulation.

Instead, we maintain the random measurements throughout the formulation [126], such that the residual $\mathbf{b}_{i,j}^h$ and state update vector ΔX_k are both also random. This leads to a *random* belief mean, ${X_k}^*$, so we must take the expectation with respect to the measurements in the objective function (2.2).

2.2.3 Considering Random Acquisitions

In addition to unknown measurement values, it is also unknown at planning time whether or not each measurement will be acquired. Therefore, we introduce a Bernoulli random variable for each measurement that models its acquisition. The set of random acquisition variables at a given planning step k is $\Gamma_k = \{\gamma_{k,j}\}_{j=1}^{n_k}$, where n_k is the number of possible measurements at step k. Therefore, we modify the observation model from (2.7) and use the following Gaussian mixture model for the update measurements:

$$\begin{aligned} &\mathbf{z}_{k,j} = h\left(X_k^j, \mathbf{v}_{k,j}(\gamma_{k,j})\right), \\ &\mathbf{v}_{k,j}(\gamma_{k,j}) \sim \mathcal{N}\left(\mathbf{0}, \left(\gamma_{k,j}\Omega_v^{k,j} + (1-\gamma_{k,j})\Omega_0\right)^{-1}\right), \end{aligned} \tag{2.16}$$

where $\Omega_v^{k,j}$ is the information contributed by a successful measurement and Ω_0 is the information contributed by an unsuccessful measurement. It is easily identified that unsuccessful measurements add zero information; that is, the second term of (2.16) vanishes, resulting in

$$\mathbf{v}_{k,j}(\gamma_{k,j}) \sim \mathcal{N}\left(\mathbf{0}, \left(\gamma_{k,j}\Omega_v^{k,j}\right)^{-1}\right). \tag{2.17}$$

Including the set of acquisition variables, the factored probability in (2.8) becomes

$$\begin{aligned} &p(X_k|\mathcal{Z}_0, \mathcal{U}_0, Z_{1:k}, \Gamma_{1:k}, U_{0:k-1}) \propto \\ &p(X_0|\mathcal{Z}_0, \mathcal{U}_0) \prod_{i=1}^{k} p(\mathbf{x}_i|\mathbf{x}_{i-1}, \mathbf{u}_{i-1}) \prod_{j=1}^{n_i} p\left(\mathbf{z}_{i,j}|\gamma_{i,j}, X_i^j\right) p\left(\gamma_{i,j}|X_i^j\right). \end{aligned} \tag{2.18}$$

Online, each $\gamma_{i,j}$ of interest is observed by the robot upon receiving the associated measurement $\mathbf{z}_{i,j}$. Acquisition variables corresponding to measurements not received do not inform the estimate of the state. But within the prediction, it is unknown which measurements will be received ahead of time. Previous work incorporated the random acquisition variables within an EM framework [67], but this method eliminates the randomness in the formulation by evaluating $\gamma_{i,j}$ at its mean value of $p(\gamma_{i,j} = 1) = \lambda_{i,j}$ within the MAP estimate. Instead, we want to maintain the randomness of the acquisition variables throughout the formulation.

We approximate the acquisition variables as independent, and therefore uninformative to the estimate, such that $p\left(\gamma_{i,j}|X_i^j\right) \approx p(\gamma_{i,j})$, allowing (2.18) to take the form

$$p(X_0|\mathcal{Z}_0,\mathcal{U}_0)\prod_{i=1}^{k} p(\mathbf{x}_i|\mathbf{x}_{i-1},\mathbf{u}_{i-1})\prod_{j=1}^{n_i} p\left(\mathbf{z}_{i,j}|\gamma_{i,j},X_i^j\right). \tag{2.19}$$

This approach essentially borrows the rationale behind EM but delays taking the expectation over the acquisition variables until the evaluation of the objective function (described later in §2.2.4).

Re-inserting the Gaussian motion and observation models into the MAP estimate of (2.5) and (2.19), the nonlinear least-squares problem becomes

$$\begin{aligned} {X_k}^* = \arg\min_{X_k}\Big[\|X_0 - {X_0}^*\|_{\Lambda_0}^2 + \sum_{i=1}^{k}\|\mathbf{x}_i - f(\mathbf{x}_{i-1},\mathbf{u}_{i-1},\mathbf{0})\|_{\Omega_w}^2 + \\ \sum_{i=1}^{k}\sum_{j=1}^{n_i}\gamma_{i,j}\left\|\mathbf{z}_{i,j} - h\left(X_i^j,\mathbf{0}\right)\right\|_{\Omega_v^{i,j}}^2\Big], \end{aligned} \tag{2.20}$$

where both $\gamma_{i,j}$ and $\mathbf{z}_{i,j}$ are random. Now, the compact representation for the state update vector ΔX_k,

$$\|A_k(U_{0:k-1})\Delta X_k - \mathbf{b}_k(Z_{1:k},U_{0:k-1})\|_{\mathcal{G}_k(\Gamma_{1:k})}^2, \tag{2.21}$$

includes the matrix of acquisition variables,

$$\mathcal{G}_k = \begin{bmatrix} I & & \\ & I & \\ & & \mathcal{D}(\gamma_{i,j}) \end{bmatrix}. \tag{2.22}$$

The update vector is now a function of the random measurements *and* the random acquisition variables,

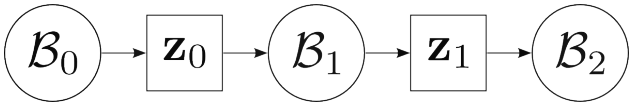
$$\Delta X_k(Z_{1:k},\Gamma_{1:k},U_{0:k-1}) = \left(A_k^\top\mathcal{G}_kA_k\right)^{-1}A_k^\top\mathcal{G}_k\mathbf{b}_k, \tag{2.23}$$

and the associated information matrix is a function of the acquisition variables as well:

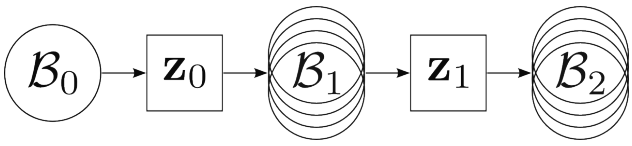
$$\Lambda_k(\Gamma_{1:k},U_{0:k-1}) = A_k^\top\mathcal{G}_kA_k. \tag{2.24}$$

This gives us a stochastic belief as a function of the random measurements and acquisition variables at each planning step, k. The posterior belief distribution is depicted in Fig. 2.3.

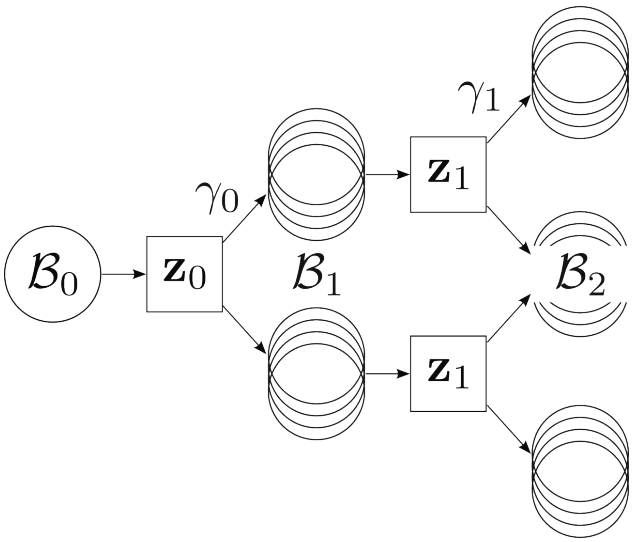
Figure 2.3 The posterior belief distribution resulting from prediction over the planning horizon. (a) Given maximum-likelihood measurements, the predicted belief is deterministic. (b) Including random measurements results in a distribution of posterior beliefs. (c) Further, including a random acquisition variable with each measurement branches the belief distribution.



(a) Maximum-likelihood measurements



(b) Random measurements



(c) Random measurements and random acquisitions

2.2.4 Objective Functions

Here we examine costs to insert into the general objective function of (2.2). We recall that we derived the belief as a function of the random measurements and acquisition variables, meaning we must take the expectation of the objective function with respect to these variables. Following the literature [67, 126], we consider costs at stage k that penalize control effort and the robot uncertainty:

$$c_k(\mathcal{B}_k, \mathbf{u}_k) = g_u(\mathbf{u}_k) + \mathop{\mathbb{E}}_{\Gamma_{1:k}} \left[g_\Lambda \left({\Lambda_k}^{-1} \right) \right], \tag{2.25}$$

where we leave the functions $g(\cdot)$ undefined for now. The expectation is taken on the uncertainty term because the information matrix Λ_k is stochastic due to the effect of the random acquisition variables. Similarly, we can write the final cost to penalize distance from a desired goal pose $\mathbf{x}_G$ and the final robot uncertainty:

$$c_K(\mathcal{B}_K) = \mathop{\mathbb{E}}_{Z_{1:K},\Gamma_{1:K}} \left[g_{\mathbf{x}}({\mathbf{x}_K}^* - \mathbf{x}_G) \right] + \mathop{\mathbb{E}}_{\Gamma_{1:K}} \left[g_{\Lambda} \left({\Lambda_K}^{-1} \right) \right]. \tag{2.26}$$

The expectation is taken for the goal pose term since the mean estimate of the belief is random, as it is a function of both the random measurements and the random acquisition variables, evidenced in (2.23).

2.3 Planning for Visual SLAM

Here we outline how we can apply the planning formulation to the visual SLAM foundation. Specifically, we show how we predict camera observations given a sequence of candidate control actions and how we handle their stochasticity in registration. In this section, we propose the use of GP regression to estimate the probability of successful registrations, or in other words, the probability of acquisition for each measurement.

2.3.1 Camera Registration Hypotheses

In our planning formulation, the measurements $\mathbf{z}_{i,j}$ stem from predicted camera registrations. We must predict camera registrations during planning in a way that models the online behavior of the visual SLAM system during execution. Following the work of Kim and Eustice [79], camera registrations are predicted according to the *saliency-informed visual SLAM* framework. The main components of this method are (i) retaining only visually-salient keyframes that are capable of being registered, and (ii) reducing the number of registration hypotheses using a measure of information gain combined with visual saliency.

This method relies on the visual saliency metrics defined by Kim and Eustice [78, 79], including *local saliency* (S_L). The local saliency score for an image is computed using the entropy of a bag-of-words (BoW) histogram from a vocabulary built online and takes on values ranging from 0 to 1 (non-salient to very salient). This score describes the texture-richness of an image and directly correlates to its registrability (Fig. 2.1).

For predicting registrations during planning, we adapt the saliency-informed visual SLAM framework in the following way. Given the predicted future poses $\{\bar{\mathbf{x}}_1, ..., \bar{\mathbf{x}}_k\}$, we search for existing target poses in the prior SLAM pose graph that may contain spatially-overlapping image views and that pass an initial saliency threshold (the same threshold used online during

SLAM). Then for each target image, we compute the geometric information gain associated with registering the target image to an image at the predicted virtual pose. Further, the information gain is scaled according to the local saliency score at the target image. This allows candidate registrations to be ranked according to a combined measure of usefulness (information gain) and likelihood of registration (local saliency). As a final step in predicting registrations, the top n_k candidates at each virtual pose are selected for incorporation into the planning formulation as measurements $\{\mathbf{z}_{k,j}\}_{j=1}^{n_k}$.

2.3.2 Visual Saliency Prediction

The local saliency score can only be calculated for images actually captured by the robot, and must be predicted for virtual poses defined by future control actions. Thus, estimating successful camera registrations hinges on accurate saliency prediction. To accomplish this task, we propose the use of GP regression to model the distribution of local saliency throughout the environment.

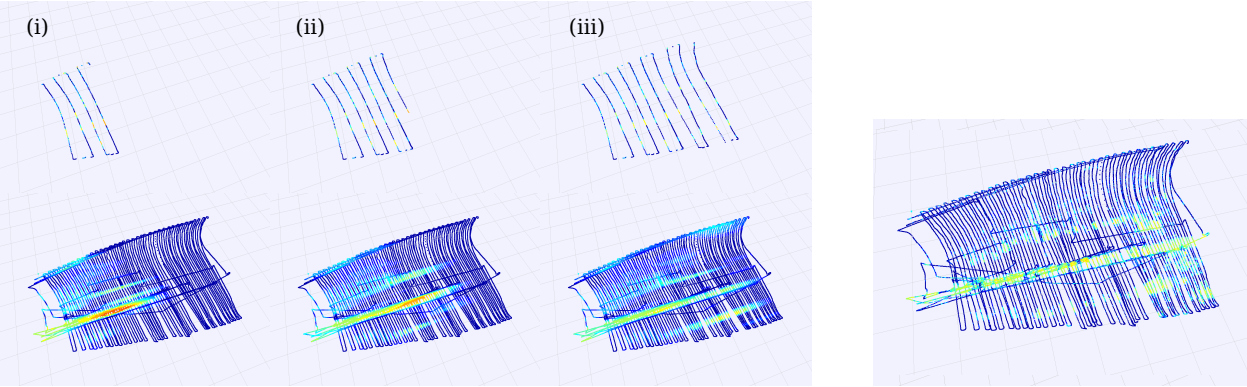
Previously for the ship hull inspection environment, the perception-driven navigation (PDN) algorithm used a simple interpolation scheme for predicting saliency scores throughout unmapped areas [80]. However, this method has no notion of whether the predicted score is accurate, and failure to accurately predict can lead to planned control actions that greatly overestimate their usefulness in planning. Instead, GP regression for visual saliency prediction is a more principled approach that accounts for the spatial distribution of visual features to model the parameters of the acquisition variables within the planning formulation. The variance of the prediction provided by the GP serves as a measure of accuracy for the predicted saliency score. More information on GP regression for learning can be found in Rasmussen and Williams [110].

Before planning, we train the GP using the captured images in association with their robot capture poses. The training data $\mathcal{D} = \{{X_0}^*, \mathbf{y}\}$ is composed of the prior poses of the robot at the m prior capture times, ${X_0}^* = \{{\mathbf{x}_i}^*\}_{i=1}^m$, and the local saliency scores of the m images, $\mathbf{y} = \{S_{L_i}\}_{i=1}^m$. Similar to Barkby et al. [7], we use a simple stationary covariance function suitable for large scale data [95]:

$$k(\mathbf{x}_i, \mathbf{x}_j; l, \sigma_0) = \begin{cases} \sigma_0 \left(\frac{2+\cos(2\pi d/l)}{3}(1 - \frac{d}{l}) + \frac{\sin(2\pi d/l)}{2\pi} \right), & \text{if } d < l \\ 0, & \text{otherwise,} \end{cases} \tag{2.27}$$

where l and σ_0 are hyperparameters. This covariance function achieves sparsity and scalability by truncating values for training pairs that exceed the weighted distance $d = \|\mathbf{x}_i - \mathbf{x}_j\|_{M_{\mathtt{GP}}} = [(\mathbf{x}_i - \mathbf{x}_j)^\top M_{\mathtt{GP}} (\mathbf{x}_i - \mathbf{x}_j)]^{1/2}$. For our application of ship hull inspection, we align the coordinate

Figure 2.4 (a) Predicted saliency maps from GP regression at different points during an example mission: (*i*) after 25% coverage with the nominal lawn-mower policy, (*ii*) 50% coverage, and (*iii*) 75% coverage. The training data is shown along the top row, with predicted saliency maps along the bottom row. (b) The actual saliency map of the environment, for qualitative comparison.



(a) Predicted saliency maps throughout a mission from GP regression

(b) Actual saliency map

frame with the survey area and choose $M_{\texttt{GP}} = \mathcal{D}(1/\Delta_d^2, 1/\Delta_h^2, 1, 0, 0, 0)$. Here, Δ_d is the distance between two along-track SLAM poses associated with the nominal exploration policy and Δ_h is the distance between two cross-track SLAM poses.

The GP prediction performance over unmapped areas is shown in Fig. 2.4. The top row of Fig. 2.4(a) depicts the pose graph of the robot at various points throughout a mission while following a nominal lawn-mower exploration policy. The bottom row of Fig. 2.4(a) shows dense coverage of saliency predictions from the GP both in the future—throughout the remaining survey area, and in the past—in unmapped areas between previously-traveled tracklines.

Fig. 2.5 shows the mean and variance from GP prediction along a sampled trajectory, considering the robot already surveyed five full tracklines. Fig. 2.5(b) shows that the variance increases as the predictions transition to farther-out areas not yet reached by the robot. Between past tracklines, the saliency prediction closely matches the actual value with small variance, shown closer in Fig. 2.5(c). This zoomed view also compares the GP prediction to the linear interpolation scheme from PDN in predicting mean values; however, the true strength of using the GP is in the measure of variance it also provides.

Together, the predicted mean and variance characterize a Gaussian probability density function (pdf) of the saliency score of a not-yet-seen image at the queried pose. The GP prediction returns a distribution of virtual pose saliency scores at the queried pose with pdf $f(S_{L_v})$, which we transform into the censored pdf $f'(S_{L_v})$ [51]. The censored pdf[1] ensures

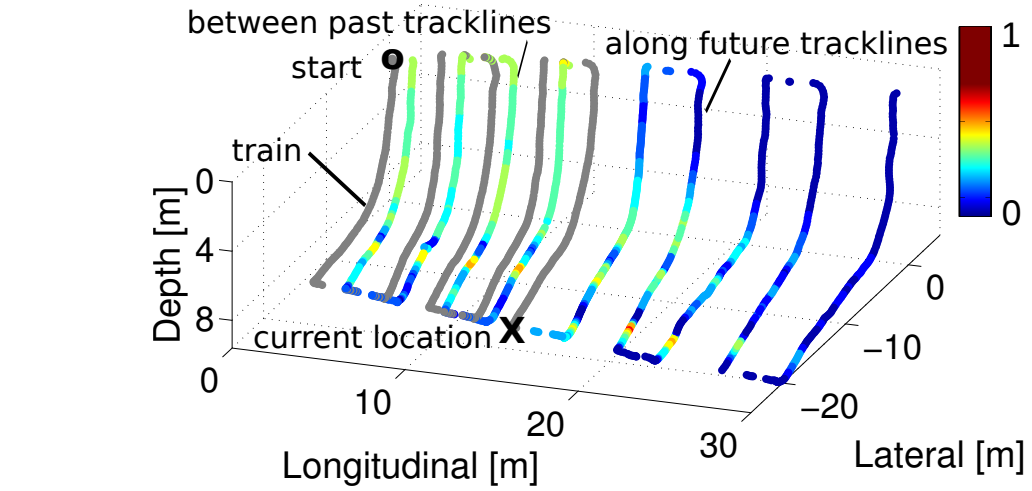
[1]Censoring involves assigning values outside of the valid range to the minimum and maximum valid bounds. This is not to be confused with truncating, which removes data outside of the valid range entirely.

that the predicted saliency is correctly represented within the acceptable range of values from 0 to 1. This censored saliency distribution will become useful in later chapters when we further define the forms of the objective functions in (2.25) and (2.26). At that time, we will show how the GP allows us to model the acquisition variables $\Gamma_{1:k}$.

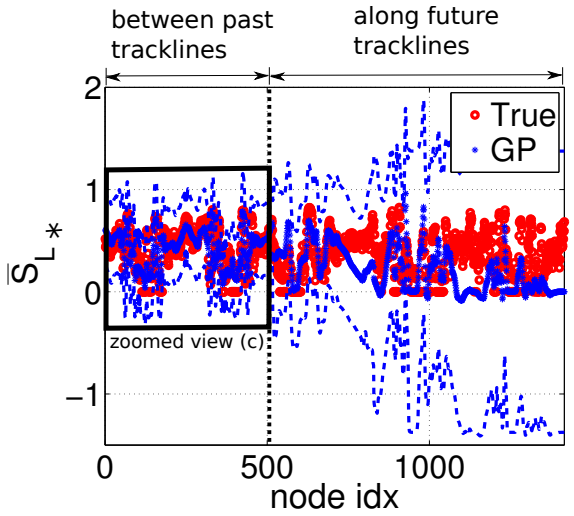
2.4 Chapter Summary

In this chapter we proposed a belief-space planning formulation that includes random measurements and random acquisition variables. The acquisition variables are Bernoulli random variables included in the belief prediction that model whether a sensor observation will actually be received. We represent the belief as a Gaussian distribution and track its evolution with the visual SLAM framework underlying our application of underwater ship hull inspection. In addition, we proposed the use of GP regression to model the distribution of visual saliency throughout the environment in order to accurately predict pairwise camera registrations given future control actions. Our planning formulation performs an optimization over a sequence of control actions with a general objective function that includes information-theoretic components and allows us to perform autonomous decision-making under uncertainty. The formulation from this chapter serves as the mathematical foundation for the rest of the research presented in this document. Specifically in future chapters, we will use the proposed planning formulation in the context of active SLAM for autonomous ship hull inspection.

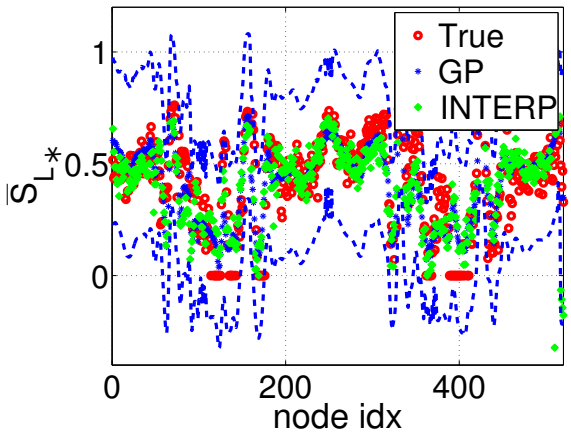
Figure 2.5 (a) The predicted saliency map from the GP along a sampled trajectory. The 'X' indicates the current robot location while gray dots represent the historical poses of the robot for training the GP. The predicted saliency scores are color-coded with respect to their saliency level. (b) Predicted mean saliency values with variance, overlaid on the true saliency scores. (c) A zoomed view for predictions between past tracklines. Two dotted blue lines indicate the 3-σ envelope of the prediction. Red circles are the true saliency scores and green diamonds are the interpolated prediction used by PDN.



(a) Sampled trajectory



(b) Variance



(c) Zoomed view

Chapter 3

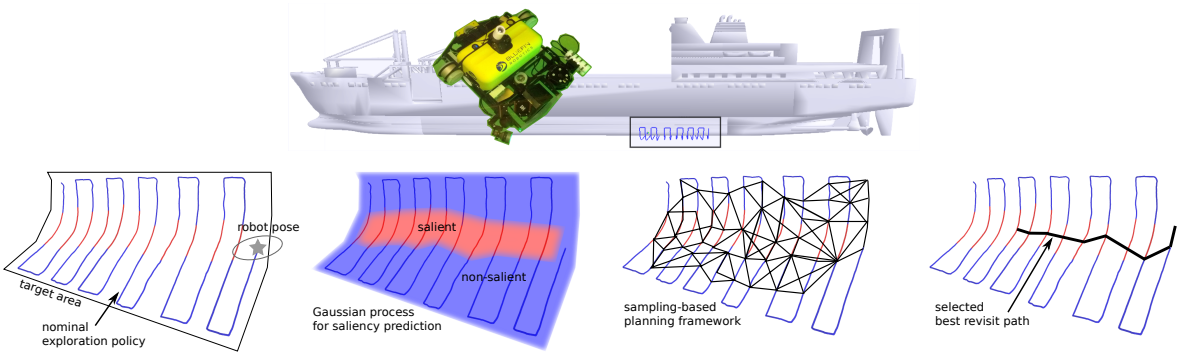
Sampling-based Algorithm for Active SLAM

3.1 Introduction

This chapter presents an algorithm for active simultaneous localization and mapping (SLAM) as applied to autonomous ship hull inspection. Our objective is to provide a robust framework for long-term autonomous inspection in the difficult underwater environment. We consider a robot surveying an *a priori* unknown target environment subject to a desired navigation uncertainty threshold. The robot explores the target environment following a nominal exploration policy—a strategy that ensures efficient area coverage but leads to an open-loop trajectory without loop-closures. Without closing loops in the SLAM formulation, the robot accumulates navigation error and its uncertainty grows unbounded. Hence, we propose an opportunistic active SLAM framework for guiding the robot to make loop-closure revisit actions throughout the mission.

An illustration of the proposed approach is given in Fig. 3.1. We propose the use of sampling-based techniques combined with information filtering to search and evaluate candidate paths throughout the environment. To model the predicted visual loop-closure utility of unmapped areas, we employ the Gaussian process (GP) regression method of Chapter 2. Our active SLAM framework is opportunistic in that it trades off convenience versus necessity; it does not wait until the uncertainty threshold is reached when selecting revisit actions to close loops in the SLAM formulation.

Figure 3.1 Overview of the proposed active SLAM framework, demonstrated for ship hull inspection. Given a target area and nominal exploration policy, the robot explores the environment subject to an acceptable navigation uncertainty. We use GP regression to predict the visual saliency of the environment and a sampling-based planning algorithm to find and evaluate loop-closure revisit paths in order to drive down the robot's uncertainty.



3.1.1 Related Work

Autonomous underwater vehicles (AUVs) are commonly used for tasks like seafloor mapping and visual inspections, so naturally there has been recent work in planning for underwater coverage problems. Hollinger et al. [58] and Englot and Hover [37] presented advances in coverage planning under uncertainty for inspections of three-dimensional structures. These works consider the uncertainty in the model of the structure when planning a traveling salesman tour of views that provide full inspection coverage.

In the context of bathymetric mapping, Galceran et al. [48] proposed a pre-planned survey design method that orders parallel tracklines in the mission according to an initial estimate of the environment saliency. They then propagate a particle filter over the planned trajectory to evaluate pose uncertainty, and insert horizontal cross-tracklines whenever the predicted uncertainty reaches a threshold. Li and Johnson-Roberson [89] presented a method for efficient seafloor mapping by performing two passes over the environment: a high-level survey for coarse resolution followed by an optimized low-level survey focused on improving the mapping resolution in specific areas of interest. These methods require a rough prior map or first pass over the target environment before planning more intelligent trajectories that improve the mapping quality. Our approach attempts to perform efficient inspection without *a priori* knowledge of the environment by planning online in conjunction with SLAM.

Our method searches for candidate actions using sampling-based planning techniques. Sampling-based planning originates from planners like the Probabilistic Roadmap (PRM) [76] and Rapidly-exploring Random Tree (RRT) [86], which sought to efficiently solve queries in high-dimensional configuration spaces. These planners strictly consider completeness—the

ability to find a path if one exists. These algorithms are *probabilistically complete*, meaning that the probability of finding a feasible path if one exists goes to one in the limit of infinite samples. Over the years, many variations of the PRM and RRT have been proposed to tailor their operation to different applications and achieve better performance [24, 85]. One such algorithm of interest for the work in this chapter is the transition-based RRT (T-RRT) [68]. This planner was proposed to solve queries when a costmap is defined over the configuration space. The T-RRT uses a transition test to accept or reject samples depending on a criterion related to the concept of minimizing mechanical work.

Karaman and Frazzoli [75] proposed the Rapidly-exploring Random Graph (RRG) and asymptotically-optimal Rapidly-exploring Random Tree (RRT*) to extend sampling-based planning to include the notion of optimality. These algorithms are adaptations of the PRM and RRT that incrementally improve the shortest-distance path to the goal with guarantees of optimality in the limit. Thus, these planners are both probabilistically complete and *asymptotically optimal*. Devaurs et al. [31] merged the concepts of the RRT* and T-RRT to propose the transition-based RRT* (T-RRT*).

Classical sampling-based planning, however, assumes deterministic states and full knowledge of the environment (or configuration space). Real-world autonomous robots generally cannot operate under these assumptions, so the general approach is to track their belief over state by performing Bayesian inference as they move throughout the world. Sampling-based planning was only recently extended to include notions of uncertainty associated with probabilistic robotics. Prentice and Roy [109] presented the Belief Roadmap (BRM) algorithm that combines the PRM with extended Kalman filtering to solve queries related to arriving at a goal pose with minimum uncertainty. Van den Berg et al. [124] presented a method for planning with stochastic measurements and motion that employed linear quadratic Gaussian (LQG) control for tracking and controlling the robot and the RRT for searching trajectories. Bry and Roy [13] developed a similar formulation that included the RRG. Both of these approaches consider the dynamics and control of the robot in order to correctly model the distribution required for checking collisions in point-to-point planning queries.

Hollinger and Sukhatme [57] generalized the sampling-based planning under uncertainty approaches to accomplish information gathering. Their Rapidly-exploring Information Gathering (RIG) class of algorithms seeks to maximize information-theoretic objectives subject to a budget constraint for the trajectory. The information-theoretic objective is often *submodular*, meaning the information available from a future action is dependent upon the prior information already gathered. As a result, the authors proposed a conservative strategy for pruning candidate paths in order to maintain computational tractability within the search space.

Research in the area of active SLAM stems from the seminal works of Bajcsy [6], Whaite and Ferrie [135], and Feder et al. [43] and generally focuses on finding control actions to reduce the uncertainty or maximize the information of the SLAM posterior. Bourgault et al. [12] proposed a greedy active exploration strategy based on the competing information gain objectives of maximizing localization accuracy and minimizing map entropy. Gonzalez-Banos and Latombe [49] extended the next-best-view (NBV) problem from computer vision [26] to include safe robot navigation in an unknown environment and constrained action selection according to some minimal sensor overlap to facilitate image registration. Rather than consider a single best action step, Sim and Roy [114] performed global active exploration using a breadth-first search over a discrete grid and restricting trajectories to omit cycles. Stachniss et al. [117] used a Rao-Blackwellized particle filter to efficiently track map hypotheses and compute the information gain associated with candidate actions.

Work by Valencia et al. [122, 123] considered path planning and active SLAM using the pose SLAM formulation that we also employ. The authors proposed a method for directly using the resulting pose graph as a roadmap over which to plan uncertainty-aware trajectories [123]. They also developed an active SLAM framework that computes information gain with a coarse map representation independent of the pose SLAM estimate, and replans whenever the estimate changes significantly [122].

Indelman et al. [67] performed active SLAM with the full SLAM formulation and a trajectory optimization approach. Like other works interested in trajectory optimization for active SLAM [105], these algorithms find locally-optimal solutions to a sequence of point-to-point planning queries touring the environment.

Much of the work in this chapter builds upon the perception-driven navigation (PDN) method presented by Kim and Eustice [80]. The PDN approach considers the same problem formulation and application of ship hull inspection. It plans revisit actions by clustering visually-salient poses in the SLAM graph to form waypoints, planning a saliency-weighted direct path to each waypoint with A*, and computing a reward for traveling along each candidate path. Our proposed algorithm improves upon this method by removing the need for clustering while evaluating many more paths using sampling-based planning techniques. In addition, we predict the visual saliency through the environment with the GP regression of §2.3.2. Our algorithm is also opportunistic, unlike PDN, in that it does not wait until the uncertainty threshold is reached to execute a beneficial revisit action.

3.1.2 Problem Statement

We consider a robot performing pose SLAM to survey an *a priori* unknown environment subject to a desired uncertainty threshold. The intention of the uncertainty threshold is to define an acceptable quality of the large-scale ship hull inspection, where the end goal is to produce a metrically-accurate map of the underwater portion of the ship. The inspection problem is formulated with the following assumptions:

1. The boundaries of the target environment are given.
2. An open-loop nominal exploration policy provides dense coverage of the target environment in efficient time (lawnmower tracklines for sensor coverage with the Hovering Autonomous Underwater Vehicle (HAUV)).
3. The user defines the navigation uncertainty threshold as a function of the robot pose covariance.
4. The environment is free of obstacles and the inspection surface is locally-planar. The effects of water current, turbidity, and vehicle dynamics are assumed negligible.
5. No other prior information is provided. Planning is performed online in conjunction with SLAM.

Considering these assumptions, the active SLAM algorithm is tasked with searching, deciding, and executing revisit actions that guide the robot to previously-seen portions of the environment in order to gather loop-closure camera registrations. The goal is to bound navigation uncertainty while maintaining efficient area coverage throughout the inspection mission. Without this type of active SLAM framework, the inspection suffers from accumulated errors over long durations. In these cases of long open-loop behavior, modeling inaccuracies in the SLAM system cause the robot to risk losing localization, leading to data association difficulties when proposing loop-closures. With a lost robot, even appearance-based approaches for closing loops consistently fail in the underwater environment [102]. Thus, we adopt an incremental approach to bounding uncertainty by performing multiple loop-closure actions throughout the mission. Our approach is built on the planning formulation of Chapter 2.

3.2 Opportunistic Active SLAM

The sampling-based active SLAM algorithm in this chapter can be broken into three components: (*i*) path generation, (*ii*) path evaluation, and (*iii*) path selection. *Path generation*

Algorithm 1 Opportunistic Active SLAM – Intermediate Approach

Input: SLAM pose graph, exploration policy
Output: best path $\mathcal{P}*$
Initialize:

```
while (SLAM is not finished) do
  x_c = GetSlamUpdate()
  if (time elapsed > Δt) then
    {x_l} = ConstructLookahead()
    for (candidate C_i in {C_i}) do
      {f_n^i} = PathGeneration(C_i, {x_l})
      EvaluateCandidate({f_n^i})
    end for
    ComputeMaxCost()
    UpdateBestCandidate()
    DecideAndExecute()
  end if
end while
```

considers the search for a number of candidate paths throughout the environment with a sampling-based framework. *Path evaluation* consists of quantizing each of these candidate paths according to their utility, and *path selection* refers to the automated process of choosing a path from the set of candidates for the robot to execute.

On the way to presenting the proposed method, we will first outline the three-phase structure and present an intermediate algorithm for active SLAM. This intermediate algorithm offers simple extensions over the PDN method for active SLAM in underwater environments, and is useful for illustrating some contributions that follow in later chapters. The final, proposed algorithm is the focus of this chapter and examines further extensions by combining the path generation and path evaluation phases into an integrated approach based on sampling-based techniques and information filtering. This final algorithm is presented later in §3.3. First, the intermediate algorithm is presented below and in Algorithm 1, as we step through the three phases of path generation, path evaluation, and path selection.

3.2.1 Saliency-weighted T-RRT* for Path Generation

The first component of any active SLAM algorithm is to generate candidate actions that the robot might execute [15]. PDN accomplished path generation by clustering visually-salient poses, selecting a representative waypoint for each cluster, and planning an out-and-back path from the current robot pose to each waypoint using A* with a saliency-weighted distance heuristic [80]. The approach we illustrate for path generation in the intermediate algorithm is similar, but features a number of extensions. The first difference is that we remove the need

Algorithm 2 Opportunistic Active SLAM: Functions

Function: `PathGeneration`$(\mathcal{C}_i, \{\mathbf{x}_l\})$

$\mathcal{P}_i = \texttt{GetTrrts}(\mathcal{C}_i)$
if ($\mathcal{P}_i$ does not exist) **then**
  $\mathcal{P}_i = \texttt{CreateTreeWithRevisitPose}()$
end if
while ($\{\mathbf{x}_l\}$ not connected) **do**
  $v_s = \texttt{SampleVertex}(\mathcal{P}_i)$
  $v_n = \texttt{GetNearestVertex}(\mathcal{P}_i, v_s)$
  if ($\texttt{TransitionTest}(\mathcal{P}_i, v_n, v_s)$) **then**
    $\texttt{Connect}(\mathcal{P}_i, v_n, v_s)$
    $\texttt{RewireTree}(\mathcal{P}_i)$
  end if
  $\texttt{TryConnect}(\mathcal{P}_i, \{\mathbf{x}_l\})$
end while
$\{f_n^i\} = \texttt{ComputeFactorsOnPath}(\mathcal{P}_i)$
return $\{f_n^i\}$

Function: `TransitionTest`$(\mathcal{P}_i, v_n, v_s)$

$(s_n, s_s) = \texttt{GetSaliencyScores}(v_n, v_s)$
if ($s_s > s_n$) **then**
  return True
end if
$p = \exp \frac{-(s_n - s_s)}{K * T}$
if ($\texttt{Rand}(0, 1) < p$) **then**
  $T = T/\alpha$
  $numFail = 0$
end if
if ($numFail > numFailMax$) **then**
  $T = T * \alpha$
  $numFail = 0$
else
  $numFail = numFail + 1$
end if
return False

for clustering and instead sample salient poses from the existing pose graph as waypoints. Each of these waypoints serves as a revisit pose for a candidate path.

We generate trajectories for each candidate with a *saliency-weighted T-RRT**, presented in the `PathGeneration`() function of Algorithm 2. This sampling-based planner can be briefly described by the ideas it borrows from other planners [31]. At the most basic level, the planner resembles the RRT* [75] with a rewiring scheme for improving the solution quality. We do not adapt the radius for rewiring as a function of vertices in the tree, however, and instead use a constant radius since our environment expands as the robot explores. The cost assignment in the rewiring step (`RewireTree`()) uses the saliency-weighted distance metric adopted from PDN:

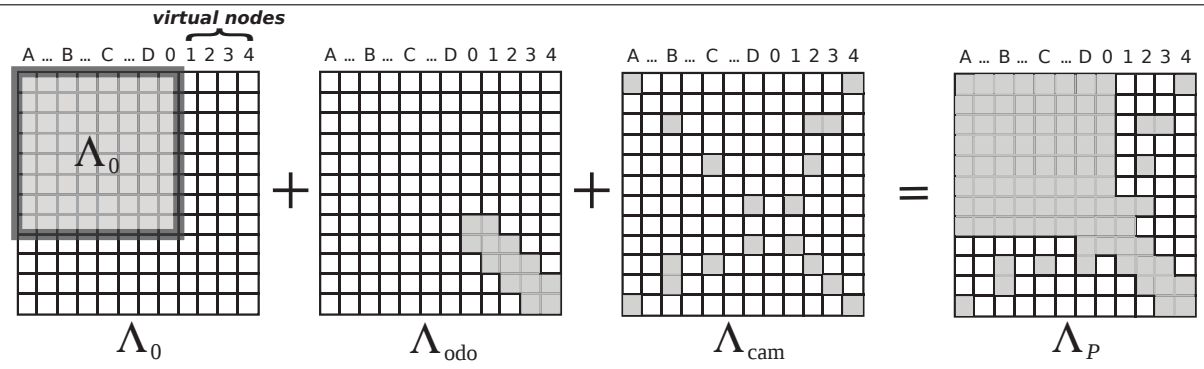
$$d_s(\mathbf{x}_i, \mathbf{x}_j) = (2 - S_{L_j})\|\mathbf{x}_i - \mathbf{x}_j\|. \tag{3.1}$$

The weighted distance $d_s(\mathbf{x}_i, \mathbf{x}_j)$ preserves the Euclidean distance to pose j from pose i when the local saliency score at j is equal to one, and doubles this distance when j has zero saliency.

During tree construction, the T-RRT* uses a transition test based on visual saliency scores before deciding whether to add a sampled vertex to the tree [31, 68]. This focuses the growth of the tree to the areas of the environment expected to yield visual registrations. The transition test, outlined in the `TransitionTest`() function of Algorithm 2, uses an adaptive probability of accepting a sampled pose. Sampled poses with a higher saliency score than the nearest vertex already in the tree are always accepted. The probability of accepting sampled poses with a lower saliency score decreases exponentially with the difference in saliency. This probability adapts according to a temperature parameter T and the number of rejected transitions.

Each candidate maintains its own T-RRT* with a root vertex positioned at a revisit waypoint. As samples are added to the tree during construction, the branches of the planner follow the peaks of the visual saliency distribution throughout the environment. Round-trip trajectories are found by climbing the tree backwards, from any leaf vertex to the root vertex, and back. Over time, the T-RRT* for a candidate represents a bank of trajectories leading to the revisit waypoint that are asymptotically-optimal with respect to the saliency-weighted distance metric. In this intermediate algorithm, we maintain a set of candidates, $\{\mathcal{C}_i\}$, each with a distinct revisit waypoint, that enumerate the possible options for revisit actions throughout the mission. We process these candidates in parallel to consider multiple options at each planning event.

Figure 3.2 The path posterior information is found by adding sources of delta information from odometry measurements and camera registrations to the prior SLAM information. The odometry measurements have a block-tridiagonal structure and camera registrations result in loop-closing constraints to previous poses in the pose graph.



3.2.2 Path Evaluation

The second phase of the active SLAM algorithm is path evaluation, where we quantize each of the candidates according to its utility. Given a candidate path, $\mathcal{P}$, we track the belief over the candidate, represented by $\mathcal{B}_\mathcal{P} \sim \mathcal{N}(X_\mathcal{P}{}^*, \Lambda_\mathcal{P}{}^{-1})$. Tracking the belief is accomplished with the formulation from Chapter 2. At the beginning of a planning event, we retrieve the linearized snapshot of the SLAM system, $\mathcal{B}_0 \sim \mathcal{N}(X_0{}^*, \Lambda_0{}^{-1})$, and determine the predicted odometry and loop-closure measurements to incorporate into the belief associated with traveling the candidate path.

For our visual SLAM application, we can deconstruct the belief posterior information as the sum of the prior SLAM information, Λ_0, and sources of delta information corresponding to the expected odometry and camera registrations available along the path, such that

$$\Lambda_\mathcal{P} = \Lambda_0 + \Lambda_{\texttt{odo}} + \Lambda_{\texttt{cam}}. \tag{3.2}$$

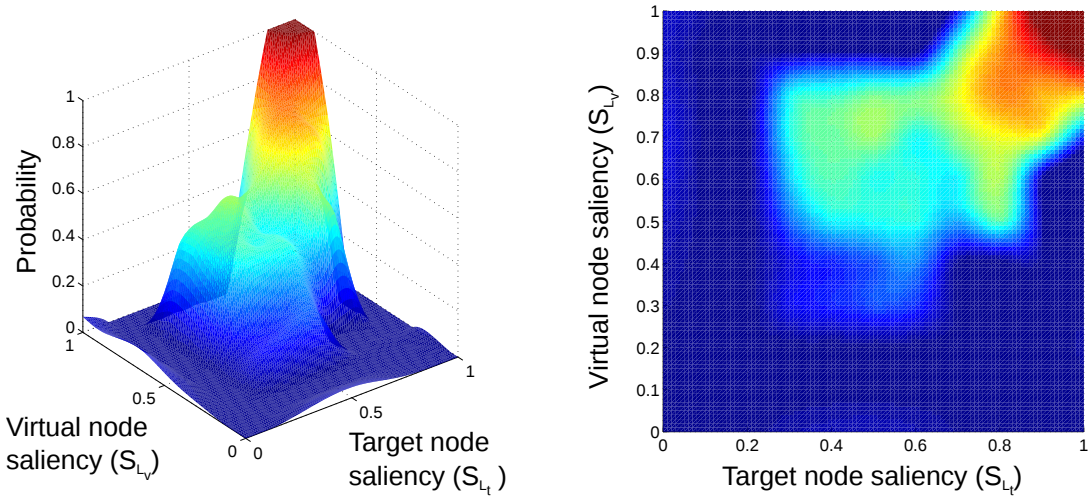
A toy example of how the information is summed along a path is given in Fig. 3.2. An odometry measurement, $\mathbf{x}_{i,i+1}$, is the relative-pose increment between sequential virtual poses in the path, $(\mathbf{x}_i, \mathbf{x}_{i+1}) \in \mathcal{P}$ [116]:

$$\mathbf{x}_{i+1} = \mathbf{x}_i \oplus \mathbf{x}_{i,i+1}. \tag{3.3}$$

Adding the expected odometry measurements along the path results in block-tridiagonal delta information given by

$$\Lambda_{\texttt{odo}} = \sum_{i=1}^{p} F_i{}^\top \Omega_w F_i, \tag{3.4}$$

Figure 3.3 The empirical probability of successful camera registration, used in (3.6). The probability is a function of target node saliency S_{L_t} and virtual node saliency S_{L_v}.



(a) Link probability function, $g(S_{L_t}, S_{L_v})$ (b) Top view

where F_i is the sparse Jacobian and ${\Omega_w}^{-1}$ is the covariance of the odometry model noise. The delta information corresponding to camera registrations expected along a revisit path is calculated by

$$\mathbb{E}[\Lambda_{\texttt{cam}}] = \sum_{i=1}^{p} \sum_{j=1}^{n_i} \lambda_{i,j} \cdot {H_{i,j}}^{\top} \Omega_v^{i,j} H_{i,j}, \tag{3.5}$$

where $H_{i,j}$ is the camera measurement Jacobian [77], $(\Omega_v^{i,j})^{-1}$ is the camera measurement noise covariance (assumed constant for convenience), and n_i is the number of camera registrations associated with virtual pose $\mathbf{x}_i$. Under the expectation, the information of a camera registration is scaled by its probability of acquisition, $\lambda_{i,j}$.

For each predicted registration (or *proposed link*), we can compute its probability of acquisition, $p(\gamma_{i,j} = 1) = \lambda_{i,j}$, given the saliency scores for the involved images and the knowledge of past registration attempts. We aggregate historical camera registration data to produce an empirical probability of successful registration as a function of target pose saliency and virtual pose saliency for overlapping pairs, $g(S_{L_t}, S_{L_v})$, shown in Fig. 3.3 and described further by Kim and Eustice [79]. Then, the probability of acquisition is computed as

$$p(\gamma_{i,j} = 1) = \lambda_{i,j} = \mathbb{E}_{f'}\left[g(S_{L_t}, S_{L_v})\right] = \int_0^1 g(S_{L_t}, S_{L_v}) f'(S_{L_v}) dS_{L_v}, \tag{3.6}$$

where $f'(S_{L_v})$ is the censored predicted saliency distribution at the virtual pose resulting from GP regression.

The belief is evaluated in an objective function designed to reflect the active SLAM

tradeoff between minimizing navigation uncertainty and maximizing area coverage rate. The planning objective is therefore to minimize the cost of a candidate path, computed by

$$\mathcal{J}(\mathcal{P}) = \alpha \cdot \frac{\mathbb{E}[g_\Lambda(\Lambda_\mathcal{P})]}{\mathcal{T}} + (1-\alpha) \cdot \frac{g_u(U_\mathcal{P})}{d_\mathcal{P}}, \tag{3.7}$$

where $g_\Lambda(\Lambda_\mathcal{P})$ is a function of the terminating pose covariance of the path and used as a measure of navigation uncertainty, and $g_u(U_\mathcal{P})$ computes the revisit (redundant) length of the candidate path. $\mathcal{T}$ is an upper bound on the acceptable uncertainty and $d_\mathcal{P}$ is an upper bound on the revisit path length, both defined by the user. The tuning parameter $\alpha \in [0, 1]$ controls the balance between uncertainty and revisit distance. Throughout the experiments in this chapter, we express the navigation uncertainty using the sixth-root of the D-optimal determinant,

$$\mathbb{E}[g_\Lambda(\Lambda_\mathcal{P})] = \mathbb{E}\left[\left|{\Lambda_\mathcal{P}}^{-1}\right|^{1/6}\right] \approx \left|\mathbb{E}[\Lambda_\mathcal{P}]^{-1}\right|^{1/6}, \tag{3.8}$$

in order to yield a quantity with units of m-rad[16], though the A-optimal trace can be used as well [114]. The expectation must be taken with respect to the random acquisition variables since the belief is a function of these unknown variables at planning time.

Per Eustice et al. [41], we recover the terminating covariance of a candidate, Σ_{nn}, by

$$\mathbb{E}[\Lambda_\mathcal{P}]\Sigma_{*n} = \mathrm{I}_{*n}, \tag{3.9}$$

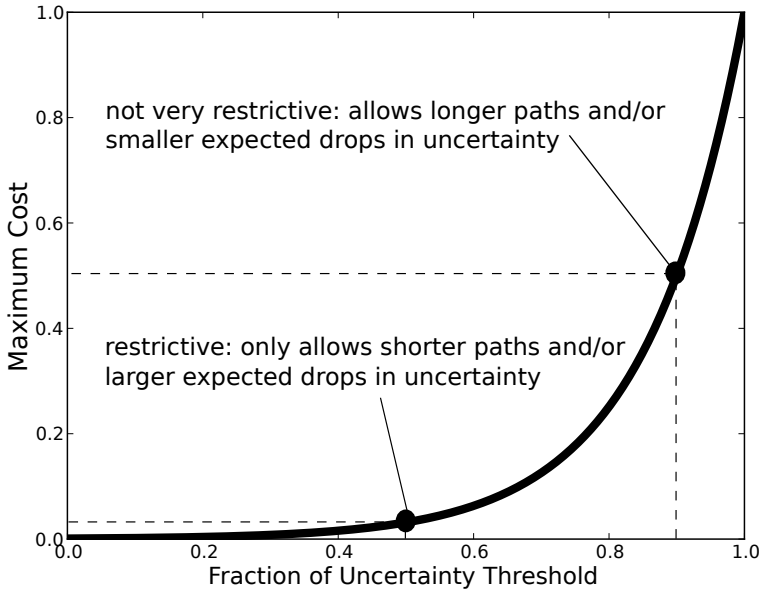
where Σ_{*n} and I_{*n} are the n^{th} block-columns of the covariance matrix and block identity matrix, respectively. Since the information matrix is exactly sparse for our visual SLAM formulation, this calculation can be performed efficiently using sparse Cholesky factorization.

3.2.3 Opportunistic Path Selection

The final phase of the active SLAM algorithm is path selection, which provides online decision-making for the robot regarding actions to execute. In our method, selecting a revisit action along a candidate path is not triggered by breaching the uncertainty threshold as in PDN [80]. Rather, the framework runs in conjunction with SLAM and is designed to be opportunistic in nature, such that the selection of a revisit action at any point during the mission is based on a tradeoff between convenience and necessity. To this end, we develop a two-step optimization to decide whether to continue the exploration policy or divert to make loop-closures along a candidate path:

1. we first predict the uncertainty along a *look-ahead horizon* to determine the maximum cost of a revisit path the framework should accept, and

Figure 3.4 The curve from (3.10) with $\beta = 1000$, used to determine the maximum allowable cost of a candidate revisit path during the planning process.



2. we then search for the candidate path with minimum cost less than this upper bound.

The look-ahead horizon is composed of the next L robot poses predicted from following the nominal exploration policy. Incorporation of the horizon gives the framework a sense of the expected measurements and predicted uncertainty, $g_\Lambda(\Lambda_{\texttt{exp}})$, for exploring with the nominal policy. We query the framework for $\Lambda_{\texttt{exp}}$ in order to compute the maximum acceptable cost for a candidate path, given by

$$J_{max} = \beta^{\left(\frac{g_\Lambda(\Lambda_{\texttt{exp}})}{\mathcal{T}} - 1\right)}, \tag{3.10}$$

where β is selected by the user. This exponential function yields a higher allowable cost as the exploration uncertainty increases (see Fig. 3.4). When the exploration uncertainty is low, the algorithm is restrictive when choosing paths, only selecting revisit actions if they are convenient and very beneficial. When the exploration uncertainty is high, the algorithm is willing to accept more costly revisit actions out of necessity. Instituting this upper bound on candidate cost is the first step in the optimization, accomplished within the function `ComputeMaxCost`().

The look-ahead horizon is also incorporated into the path generation process by designating each pose along the horizon as a valid goal point for the T-RRT* candidates. In this way, a valid round-trip trajectory (revisit path) is defined for each candidate when the T-RRT* intercepts the horizon.

The second step in the optimization is the search for the candidate with minimum cost below J_{max}, performed by the `UpdateBestCandidate()` function. If no candidate revisit actions are found with cost below J_{max}, exploration with the nominal policy continues. Every Δt, we begin a new cycle of the active SLAM algorithm, which runs in parallel with SLAM for the duration of the hull inspection mission.

3.3 Integrating Path Generation and Evaluation

The previous, intermediate active SLAM method contains some differences compared to the PDN method. Notably, path generation is accomplished with the T-RRT*, and the look-ahead horizon and two-step optimization contribute to an opportunistic behavior. There are many similarities that exist, however; both methods enumerate out-and-back actions to a limited number of revisit waypoints, and both proceed in the three distinct phases of path generation, evaluation, and selection.

Here we present our main active SLAM framework that integrates the path generation and evaluation phases. This integration allows us to evaluate many more candidate paths (hundreds, typically) while maintaining relative efficiency. The key lies in combining sampling-based techniques for path generation with information filtering for belief tracking and path evaluation. The proposed algorithm is presented in Algorithm 3. Rather than use *multiple* T-RRT*s for path generation and evaluate each candidate individually, we replace these processes with a *single* roadmap that simultaneously generates and evaluates candidates incrementally as the roadmap is constructed. This integrated approach is described below.

3.3.1 Sampling-based Graph Construction

We sample candidate paths for active SLAM using the RRG [13, 56, 75], which incrementally builds a roadmap of vertices and edges describing the connectivity through the configuration space of the robot. Contained at each vertex in the RRG is a list of partial candidate revisit paths (hereafter called just *candidates*, denoted by $\mathcal{P}_i$) that each describe a unique trajectory over edges in the RRG to arrive at the vertex. Every candidate at every vertex is tracked by the framework and represented by its belief mean, $X^*_{\mathcal{P}} = \bar{X}_{\mathcal{P}}$, and associated information, $\Lambda_{\mathcal{P}}$, from (3.2). The virtual poses in the mean vector, $\{\mathbf{x}_1, \cdots, \mathbf{x}_p\}$, arise from traveling along edges, and the information matrix $\Lambda_{\mathcal{P}}$ is the sum of the SLAM information, Λ_0, and the delta information gathered along the way. Fig. 3.5 displays an example RRG sampled on a typical SLAM pose graph built by the HAUV during ship hull inspection.

A benefit of using the information form to track candidate beliefs is that each of the sources

Algorithm 3 Opportunistic Active SLAM – Final Integrated Approach

Input: SLAM pose graph, exploration policy
Output: best path $\mathcal{P}*$
Initialize: vertices $V = \{\}$, edges $E = \{\}$, queue $Q = \{\}$

while (SLAM is not finished) **do**
  $\mathbf{x}_c = \texttt{GetSlamUpdate}()$
  if (time elapsed $> \Delta t$) **then**
    $\{\mathbf{x}_l\} = \texttt{ConstructLookahead}()$
    $\texttt{ComputeMaxCost}()$
    while (computation time remains) **do**
      $\mathbf{x}_{sample} = \texttt{SamplePose}()$
      $v_{new} = \texttt{ExtendToNearest}(\mathbf{x}_{sample})$
      $V_{near} = \texttt{FindNearVertices}(v_{new})$
      for (all v_k in V_{near}) **do**
        $E = E \cup \{\texttt{Connect}(v_k, v_{new}), \texttt{Connect}(v_{new}, v_k)\}$
        $Q = Q \cup \{\text{all } \mathcal{P} \text{ at } v_k\}$
      end for
      while (Q is not empty) **do**
        $\texttt{ProcessQueue}()$
      end while
      $\texttt{EvaluateFinishedCandidates}()$
      $\texttt{UpdateBestCandidate}()$
    end while
    $\texttt{DecideAndExecute}()$
  end if
end while

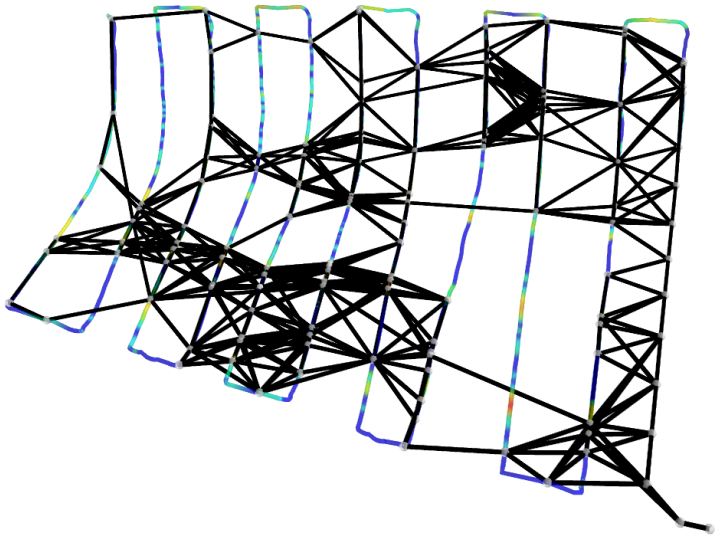
Function: $\texttt{ExtendToNearest}(\mathbf{x}_i)$

$v_{nearest} = \texttt{FindNearestVertex}(\mathbf{x}_i)$
$v_{new} = \texttt{SteerToward}(\mathbf{x}_i, v_{nearest})$
$V = V \cup v_{new}$
$E = E \cup \{\texttt{Connect}(v_{nearest}, v_{new}), \texttt{Connect}(v_{new}, v_{nearest})\}$
$Q = Q \cup \{\text{all } \mathcal{P} \text{ at } v_{nearest}\}$
return v_{new}

Function: $\texttt{ProcessQueue}()$

$\mathcal{P}_{parent} = \texttt{Pop}(Q)$
for (all $v_{neighbor}$ of $v(\mathcal{P}_{parent})$) **do**
  $\mathcal{P}_{child} = \texttt{PropagateCandidate}(e_{neighbor}, \mathcal{P}_{parent})$
  if (not $\texttt{PruneCandidate}(v_{neighbor}, \mathcal{P}_{child})$) **then**
    $\texttt{AddCandidateToVertex}(v_{neighbor}, \mathcal{P}_{child})$
    $Q = Q \cup \mathcal{P}_{child}$
  end if
end for

Figure 3.5 An example RRG sampled over a typical SLAM pose graph from a ship hull inspection. Black lines represent edges between vertices in the RRG.



of delta information from (3.2) can be divided into multiple components and attributed to the edges from which they originate. In this way, the total delta information for a candidate is simply the sum of the delta information contributed by each edge that it travels. Hence, (3.2) can be rewritten as

$$\Lambda_{\mathcal{P}} = \Lambda_0 + \Lambda^1_{\texttt{edge}} + \ldots + \Lambda^{n_e}_{\texttt{edge}}, \tag{3.11}$$

where

$$\Lambda^i_{\texttt{edge}} = \Lambda^i_{\texttt{odo}} + \Lambda^i_{\texttt{cam}}, \tag{3.12}$$

and $\Lambda^i_{\texttt{edge}}$ represents the delta information matrix encoded by the odometry and camera registration factors arising from the edge. A key insight is that the delta information $\Lambda^i_{\texttt{edge}}$ only needs to be computed once during construction, and is additively applied to a candidate's information when the edge is traversed. This benefit of the information form was alluded to by Valencia et al. [123] and is analogous to the one-step transfer function used by the BRM [109]. Edge construction is accomplished within the `Connect()` function of the proposed algorithm.

3.3.2 Candidate Propagation and Pruning

As the RRG is built, the algorithm grows a tree of candidates over the roadmap, where candidates can be thought of as leaves of the tree. The root of the tree is initialized at the most recent SLAM pose with initial information Λ_0. New leaves are generated by creating a new vertex from a sampled pose in the configuration space (v_{new}), connecting nearby

vertices (V_{near}) to the new vertex with new edges, and propagating the candidates from the nearby vertices over the new edges and recursively throughout the rest of the graph. Propagating a candidate over an edge (and hence creating a new leaf) amounts to branching the candidate (parent leaf) from the edge source vertex, creating a new candidate (child leaf) at the destination vertex, and calculating the child information by adding the delta information from the edge to the parent leaf information. Without pruning leaves, the tree encodes every possible path through the graph to reach any vertex from the root.

However, the number of candidates tracked by the framework can quickly become too large for computational feasibility. It is logical to prune leaves of the tree that are not useful given the objective. A conservative pruning strategy would eliminate only suboptimal leaves from the tree [13, 56], but we are willing to employ a more aggressive heuristic that sacrifices optimality for a large increase in speed, as suggested by Hollinger and Sukhatme [56] for submodular objective functions. Thus, we maintain a partial ordering of candidates at each vertex according to the distance and uncertainty metrics of (3.7):

$$\mathcal{P}_a > \mathcal{P}_b \Leftrightarrow g_u(U_{\mathcal{P}_a}) < g_u(U_{\mathcal{P}_b}) \wedge g_\Lambda(\Lambda_{\mathcal{P}_a}) < g_\Lambda(\Lambda_{\mathcal{P}_b}) + \epsilon, \tag{3.13}$$

where ϵ is a small factor to aid in pruning when the candidates are quite similar [13]. When (3.13) is true, candidate $\mathcal{P}_b$ has both a longer revisit distance and a higher uncertainty than $\mathcal{P}_a$, so $\mathcal{P}_b$ can be pruned from the vertex, as well as its children.

3.3.3 Opportunistic Path Selection

Like the intermediate algorithm, we opportunistically select paths using the two-step optimization that includes the look-ahead horizon and maximum acceptable cost for a candidate. However, in this final proposed framework the look-ahead horizon incorporates differently into the path generation process. Here, we add the poses in the horizon as vertices in the RRG and designate each as a valid goal point, such that the algorithm searches for candidates that divert from, and return to, the nominal exploration policy. (The horizon represents the next L steps of the nominal exploration policy. We used $L = 100$ in the hybrid simulation experiments, corresponding to a 20 m horizon. During field trials, the horizon corresponded to the length of an entire trackline.) From here, the algorithm proceeds with RRG construction and candidate path propagation.

For computing the maximum acceptable cost of a candidate in (3.10), we used $\beta = 1000$ in hybrid simulation and $\beta = 20$ in field trials.

3.3.4 Implementation Details

The computational complexity of the proposed algorithm can be roughly attributed to three sources: construction of the RRG, propagation of candidates throughout the RRG, and the evaluation of the cost of candidates for pruning and best path selection. The complexity of the construction phase is well-documented [75], although we include the additional computation required to predict the measurements available along each edge. Similarly, Hollinger and Sukhatme [56] present the complexity of propagating candidates throughout the RRG, which is exponential in the worst case but tractable for aggressive pruning strategies like the one we propose. Evaluation of the cost of a candidate using the Cholesky decomposition is generally $O(n^3)$, where n is the dimension of the information matrix, but lower with methods for sparse systems. A helpful feature of the incremental sampling-based nature of the algorithm is that it finds solutions quickly but searches to improve the best path as computation time remains.

Below we describe some implementation techniques to help reduce overall computation time, in addition to common methods like biasing the sampling toward salient regions or adjusting parameters related to RRG connectivity and pruning aggressiveness.

Edge Marginalization

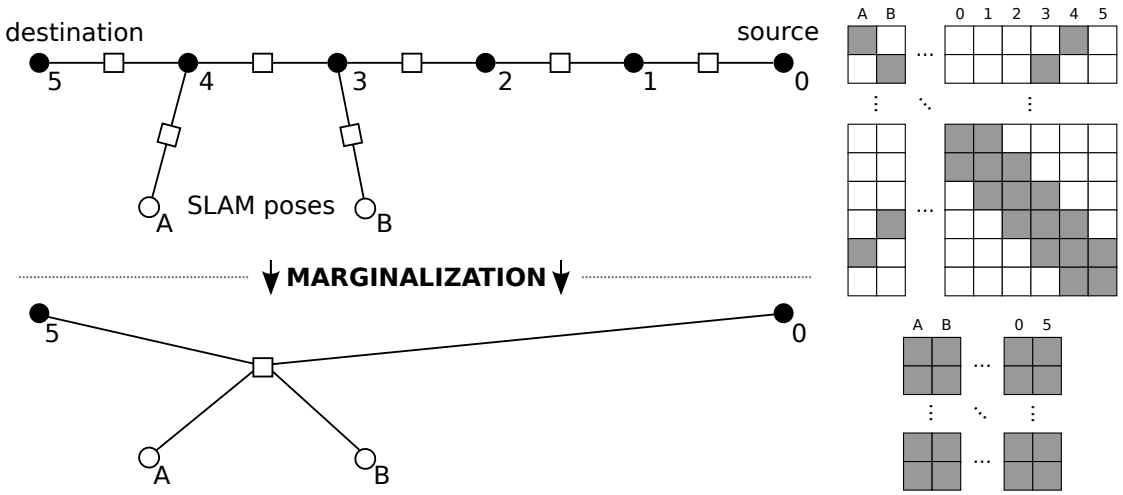
When an edge is traversed during propagation, the size of the candidate's information matrix grows proportional to the number of virtual poses added by the edge. For example, an edge adding five 6-degree of freedom (DOF) virtual poses to a candidate path adds five odometry factors to the pose graph and grows the candidate information matrix by 30 rows and columns. To alleviate this growth and leverage the information form parameterization, we can use marginalization to significantly reduce the dimensionality of the delta information of an edge during its construction.

Consider the example edge shown in Fig. 3.6(a); rather than explicitly representing all the factors along the edge, we condense their combined delta information ($\Lambda^i_{\texttt{edge}}$) into a single n-ary factor by marginalizing out the intermediate poses in the edge (i.e., edge poses that are not the source or destination). The single factor is represented by the marginalized delta information, $\Lambda^i_{\texttt{marg}}$, which replaces $\Lambda^i_{\texttt{edge}}$ but induces the exact same information into the factor graph. The marginalized delta information is found in the usual way via the Schur complement,

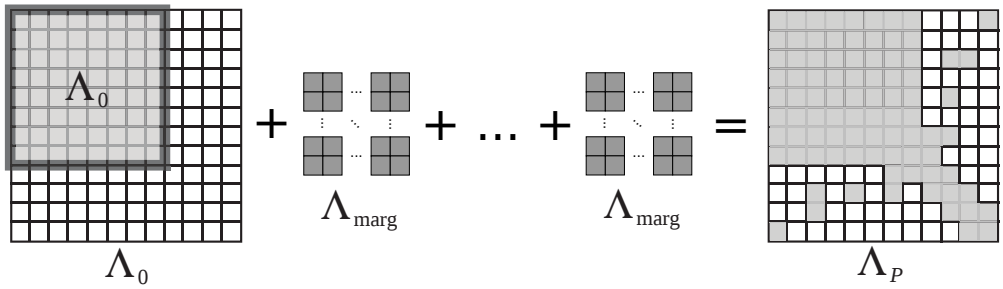
$$\Lambda^i_{\texttt{marg}} = \Lambda^i_{\texttt{aa}} - \Lambda^i_{\texttt{ab}} {\Lambda^i_{\texttt{bb}}}^{-1} \Lambda^i_{\texttt{ba}}, \tag{3.14}$$

where `a` represents the source and destination poses (as well as states from target poses corresponding to expected camera registrations) and `b` represents intermediate poses on the edge. This process is visualized in Fig. 3.6(a). Then, for implementation, (3.11) takes the

Figure 3.6 (a) The factor graph and delta information matrix arising from an edge before (top, $\Lambda_{\texttt{edge}}$) and after marginalization (bottom, $\Lambda_{\texttt{marg}}$) during construction. (b) The algorithm builds the posterior information incrementally from the sources of delta information arising from each edge in the path. The marginalization technique reduces the complexity of path evaluation within the algorithm.



(a) Edge construction with marginalization technique



(b) Calculating path posterior information using marginalized edges

final form,

$$\Lambda_{\mathcal{P}} = \Lambda_0 + \Lambda^1_{\texttt{marg}} + \ldots + \Lambda^{n_e}_{\texttt{marg}}, \tag{3.15}$$

where n_e is the number of edges traversed by the candidate.

As a result of this marginalization, a candidate's mean and information are augmented by only *one* new virtual pose upon traversing an edge, no matter how many virtual poses it originally contained.

Parallel Evaluation

In our C++ implementation of the algorithm, propagation and evaluation of a candidate path over an edge averages 33.2 ms for representative missions like those in §3.4 and §3.5. Roughly

80% of this time is due to calculating the terminating covariance via Cholesky factorization of $\Lambda_{\mathcal{P}}$. We can achieve significant speed-up by parallelizing evaluations of candidates within `ProcessQueue`() by performing the necessary Cholesky factorizations across multiple threads. For the *USCGC Escanaba* field trials presented in §3.5.2 that used this parallel implementation, the equivalent timing for propagation and evaluation of a candidate path was reduced to an average of 9.0 ms.

3.4 Hybrid Simulation Results

We tested the proposed active SLAM framework both in a hybrid simulation environment, presented here, and in real-world field trials with the HAUV, presented in the next section. The hybrid simulation incorporates real-world data collected in field trials but simulates the path traveled by the robot. In particular, the hybrid simulation presented here uses data collected during a ship hull inspection of the *SS Curtiss* by the HAUV in February 2011. More information about this experimental setup and dataset can be found in the work by Kim and Eustice [80].

3.4.1 Synthetic Saliency Environment

The first simulation features real odometry and depth measurements collected by the robot but uses synthetic imagery, such that salient and non-salient portions of the environment are assigned by the user. We compare the proposed active SLAM method against three other methods: an open-loop survey with no revisit actions, a preplanned deterministic strategy with revisit actions along every trackline, and PDN [80]. The deterministic case is simulated twice—over an optimistic salient region of the environment and over a pessimistic non-salient region. In all scenarios, the robot follows the same nominal exploration policy and the compared planning methods determine when to make diversions and which revisit paths to take.

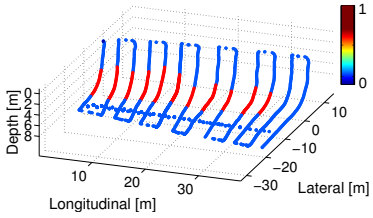
Results from the synthetic saliency simulation are shown in Fig. 3.7. While the path length of the deterministic method is inherently long, its performance at bounding the uncertainty of the robot is completely dependent upon whether the preplanned revisit paths travel a salient region. As such, we see its two extremes of the spectrum: a salient case that serves as a good baseline for "best-possible" uncertainty performance, and a non-salient case that underperforms even the open-loop survey. Considering the problem formulation of operating in an *a priori* unknown environment, the selection of salient preplanned revisits, and hence the performance of this method, are left completely up to chance.

Table 3.1 Summarized Results

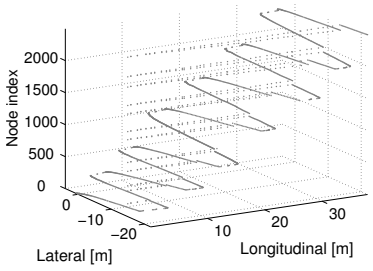
Method	**Path Length** [m]	**Avg. Uncertainty** [% of $\mathcal{T}$]
Hybrid Simulation—Synthetic Saliency		
Open-loop	386.5	121.8
DET (non-sal)	885.2	151.7
DET (sal)	876.2	33.4
PDN	583.3	71.3
Proposed	524.5	42.5
Hybrid Simulation—Real Imagery		
Open-loop	386.5	121.8
DET	865.8	64.3
PDN	527.2	67.2
Proposed	557.3	53.7
MHL Field Trial 1		
Open-loop	124.1	111.8
Proposed	233.1	61.0
MHL Field Trial 2		
Open-loop	77.2	99.3
DET	271.3	199.3
Proposed	124.5	60.4
USCGC Escanaba Field Trial 1		
Open-loop	109.8	97.2
DET	203.9	55.3
Proposed	214.3	56.2
USCGC Escanaba Field Trial 2		
Open-loop	94.4	90.0
DET	163.0	67.3
Proposed	169.8	63.8

PDN identifies salient areas of the environment online and thus performs well regarding path length and uncertainty. However, it is a naïve framework that only enumerates a few candidate revisit paths and simply waits until the uncertainty threshold is breached before gathering loop-closures. In contrast, the opportunistic nature and sampling-based approach of the proposed method evaluates hundreds of candidate paths for revisiting at any point during the mission. As a result, the proposed method outperforms the deterministic and PDN methods in terms of path length and results in uncertainty levels similar to the "best-possible" deterministic case. Summarized statistics for the hybrid simulations (and all field trials) are presented in Table 3.1.

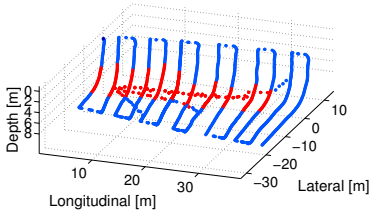
Figure 3.7 Hybrid simulation results with the synthetic saliency environment. The non-salient deterministic strategy (a) (b) results in no loop-closures and underperforms even the open-loop survey. Both PDN (c) (d) and the proposed method (e) (f) constrain the navigation uncertainty with loop-closures throughout the mission. The uncertainty versus path length plots are shown in (g). SLAM poses in the trajectory plots are color-coded by their visual saliency, from red (salient) to blue (non-salient). Visual loop-closures are represented by red links on the time elevation plots.



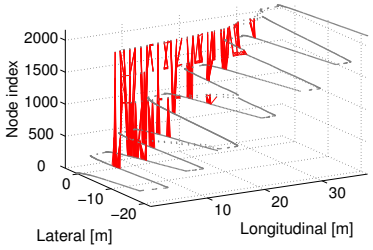
(a) Trajectory—DET (non-sal)



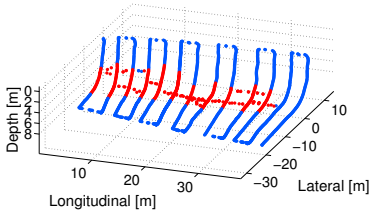
(b) Time Elevation



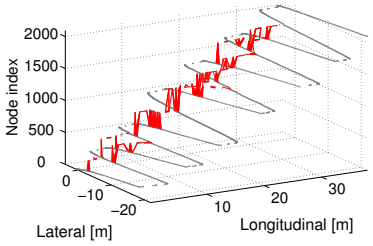
(c) Trajectory—PDN



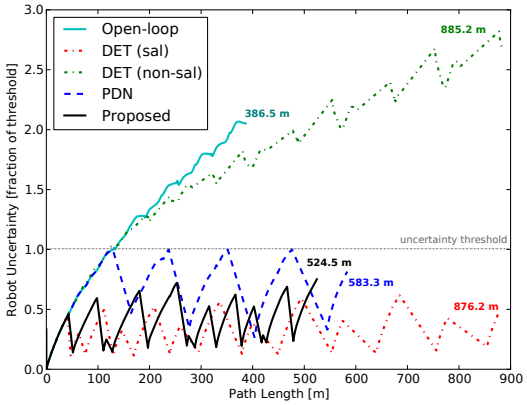
(d) Time Elevation



(e) Trajectory—Proposed

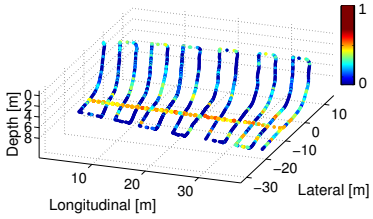


(f) Time Elevation

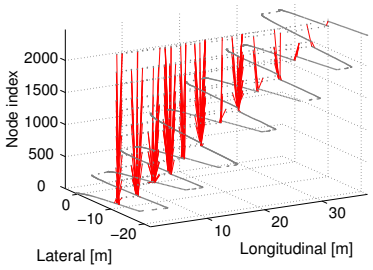


(g) Uncertainty vs. Path Length

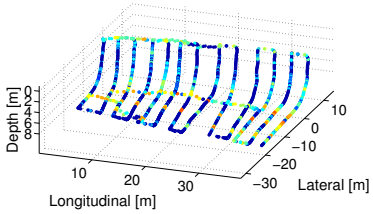
Figure 3.8 Hybrid simulation results using real imagery collected by the HAUV while surveying the *SS Curtiss*. The deterministic strategy (a) (b) results in an unnecessarily long path length. PDN (c) (d) results in the shortest path length but the proposed method (e) (f) yields the lowest average uncertainty for the mission. The uncertainty versus path length plots are shown in (g). Visual loop-closures are represented by red links on the time elevation plots.



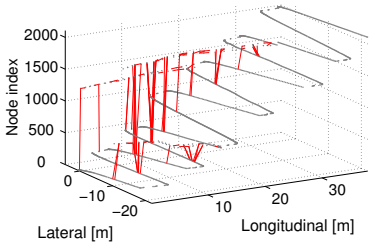
(a) Trajectory—DET



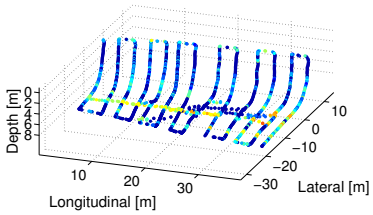
(b) Time Elevation



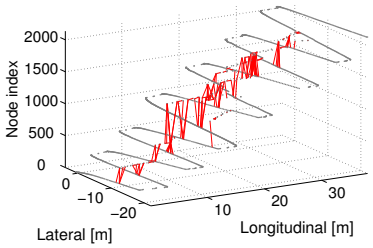
(c) Trajectory—PDN



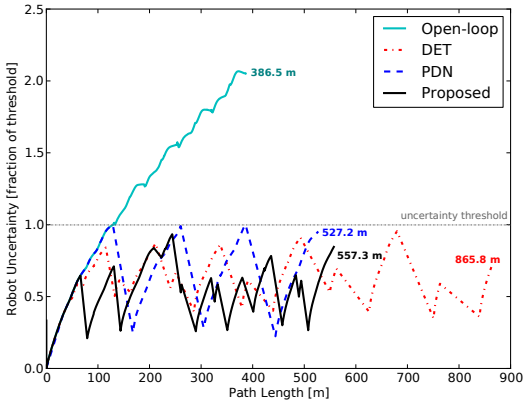
(d) Time Elevation



(e) Trajectory—Proposed

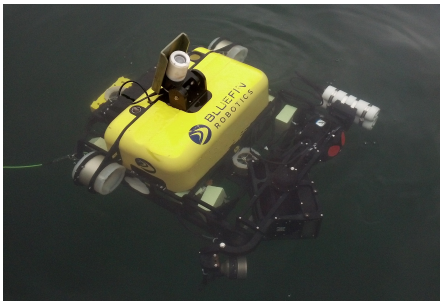


(f) Time Elevation



(g) Uncertainty vs. Path Length

Figure 3.9 (a) The HAUV, developed by Bluefin Robotics, in the water before field testing. (b) The HAUV performing a seafloor survey in the physical modeling basin at the University of Michigan's Marine Hydrodynamics Laboratory. (c) The *USCGC Escanaba*, used for hull-relative inspection field trials. Image from U.S. Coast Guard at coastguard.dodlive.mil.



(a) HAUV

(b) Physical modeling basin at MHL

(c) *USCGC Escanaba*

3.4.2 Real Image Data

Here we present results using the hybrid simulation with the real imagery collected during the *SS Curtiss* inspection. Images from the dataset densely cover the entire target environment, as shown in Fig. 2.1. We use the same nominal exploration policy as the previous simulation but instead use the real camera images recorded by the robot to calculate saliency scores and attempt loop-closure registrations. The proposed algorithm is again compared to the three other active SLAM methods. Results are presented in Fig. 3.8.

This time, the deterministic revisits happen to travel portions of the environment that yield some camera registrations, but none that significantly drive down the uncertainty. PDN slightly outperforms the proposed method in overall path length. (Notice, though, that PDN is very close to triggering the execution of a fourth revisit action at the end of the mission.) Still, the proposed method results in an average uncertainty along the exploration trajectory that is 20% lower than the PDN result with less than 6% increase in path length.

3.5 Real-world Field Trials

In addition to the hybrid simulations, we tested the proposed framework in real-world field trials with the HAUV. We applied the framework to two different types of inspections with the robot: seafloor surveys of a long, narrow basin, and hull-relative surveys of a U.S. Coast Guard cutter. Details of these field trials are presented below and results are summarized in Table 3.1. Table 3.2 reports the algorithm parameters used in field testing. All parameters were selected to illustrate sufficient operation of the proposed method in the target environment.

Table 3.2 Parameters for Field Trials

	MHL	**Escanaba 1**	**Escanaba 2**
α	0.9	0.9	0.75
β	20	20	20
$\mathcal{T}$ [m-rad]	4×10^{-5}	6×10^{-5}	6×10^{-5}
$d_{\mathcal{P}}$ [m]	40	100	100

3.5.1 MHL Seafloor Surveys

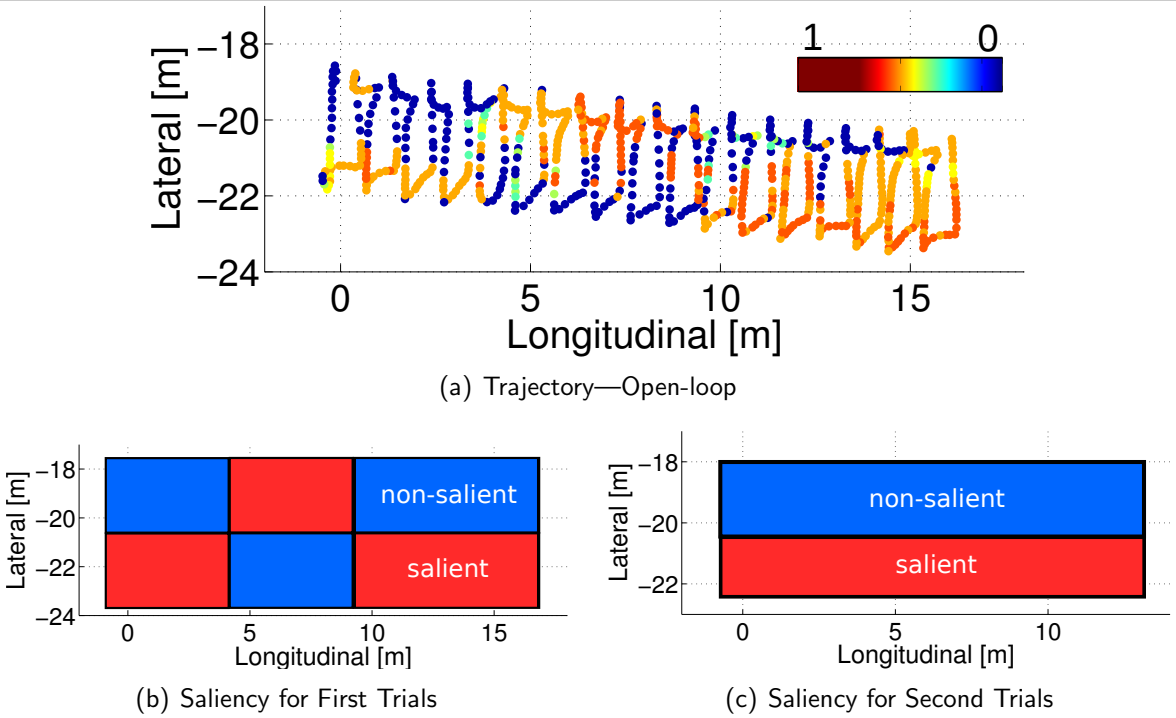
We used the HAUV and the proposed framework to survey the floor of the physical modeling basin at the Marine Hydrodynamics Laboratory (MHL) at the University of Michigan, seen in Fig. 3.9(b), in May 2014. The physical modeling basin is over 100 m long, 6 m wide, and 3 m deep. For seafloor surveys, the HAUV is configured with both the Doppler velocity log (DVL) and underwater camera facing downward and a nominal exploration policy similar to those for hull-relative surveys—to cover the target area in an open-loop fashion with back and forth tracklines. We performed these surveys with the HAUV operating just below the waterline and limited the target area to a (roughly) 15 m section of the basin.

Since the basin is a very salient environment, images acquired almost anywhere are useful for making camera registrations. To increase the difficulty of making loop-closure registrations and obtain scenarios that better illustrate the applicability of our approach, we blurred the imagery in some portions of the basin to degrade its quality. Blurring occurred in specific areas of the basin according to an independent estimate of the robot pose from an AprilTag fiducial marker [99] observed through the HAUV's periscope camera. By controlling the image quality, we uniquely tailored the saliency of the environment for two sets of field trials.

For the first set of trials at MHL, we blurred the imagery in the environment in offsetting rectangular regions. This resulted in alternating salient and non-salient areas of the basin as seen in Fig. 3.10(b). We tested the proposed framework against the open-loop nominal policy in this set of trials, with results shown in the left column of Fig. 3.11. Some small cross-track loop-closures are made during the open-loop survey, but none that prevent the navigation uncertainty from growing unbounded. The planning algorithm within the proposed framework efficiently bounds the uncertainty by selecting revisit paths that weave through the salient patches of the environment and avoid non-salient patches. The proposed method does result in an overall path length nearly twice as long as the open-loop case, however.

The environment for the second set of field trials at MHL consisted of a single salient band running lengthwise along the edge of the basin, shown in Fig. 3.10(c). For this set of trials, we compared the proposed framework against the open-loop policy and a deterministic strategy

Figure 3.10 The experimental setup for the field trials at MHL. (a) The open-loop trajectory resulting from following the nominal exploration policy. (b) The environment saliency for the first set of trials. (c) The environment saliency for the second set of trials.



(a) Trajectory—Open-loop

(b) Saliency for First Trials

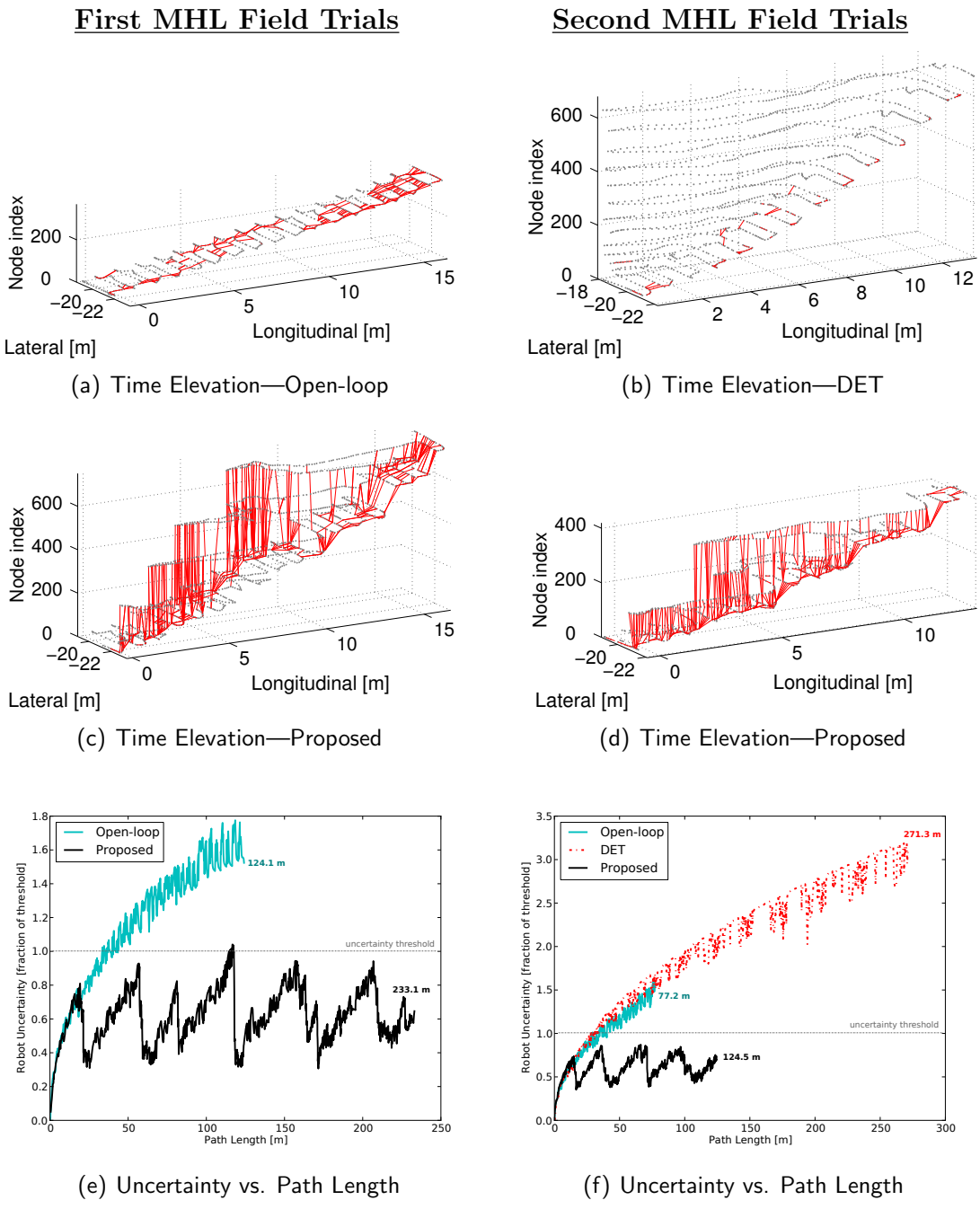
(c) Saliency for Second Trials

with preplanned revisit actions along every other trackline. Once again, the preplanned actions travel a non-salient portion of the environment and the deterministic strategy performs poorly, with large uncertainty growth and extremely long path length. The time elevation graph of Fig. 3.11(b) shows zero large loop-closing camera registrations. Contrastingly, the resulting trajectory from the proposed framework shows loop-closures spread uniformly throughout the mission, as revisit actions are performed within the salient portion of the environment. Compared to the deterministic case, the proposed method surveyed the target area with a 54% shorter path length and 70% lower average navigation uncertainty. Even if the deterministic strategy happened to travel the salient portion of the basin, the path length would be equivalent to the deterministic result shown. Refer to Table 3.1 for a summary of results.

3.5.2 *USCGC Escanaba* Hull-relative Surveys

The proposed active SLAM framework was also tested while visually inspecting the hull of the *USCGC Escanaba* in Boston, MA in October 2014. The *USCGC Escanaba* (WMEC-907) is 82 m in length with a beam of 12 m and draft of 4.4 m. For hull-relative missions, the HAUV

Figure 3.11 Results from field trials performed in the MHL physical modeling basin at the University of Michigan. Results pertaining to the first set of trials is shown in the left column and the second set of trials in the right column. In the first set of trials, small cross-track camera registrations are made during the open-loop survey (a), but none contribute to a large reduction in uncertainty. The proposed method (c) produces revisit paths that weave through the salient patches in the environment. In the second set of trials, the deterministic strategy results in zero large loop-closures as the preplanned revisit paths travel the non-salient portion of the environment. The proposed framework (d) results in visual loop-closures spread throughout the mission. The uncertainty versus path length plots are shown in (e) and (f). Visual loop-closures are represented by red links on the time elevation plots.



(a) Time Elevation—Open-loop

(b) Time Elevation—DET

(c) Time Elevation—Proposed

(d) Time Elevation—Proposed

(e) Uncertainty vs. Path Length

(f) Uncertainty vs. Path Length

travels along the underwater portion of the ship hull and actuates the DVL and underwater camera such that they always point normal to the hull surface. Two sets of experiments were performed on a section of the hull situated between the stabilizer fin and bow of the ship. Each set of trials included surveys using the open-loop, deterministic, and proposed active SLAM strategies.

Results from the first set of trials on the *USCGC Escanaba* are given in Fig. 3.12. The preplanned deterministic strategy revisits a waypoint placed on the first trackline of the mission through a salient patch along the hull. From the time elevation graph in Fig. 3.12(d), it can be seen that the deterministic case relies on large loop-closures near the beginning of the mission to bound the navigation uncertainty. The proposed method yields a trajectory with loop-closures distributed throughout the salient portions of the environment. Both strategies, however, produce favorable end results—with nearly identical average uncertainties well below the acceptable threshold and path lengths within 11 m of each other.

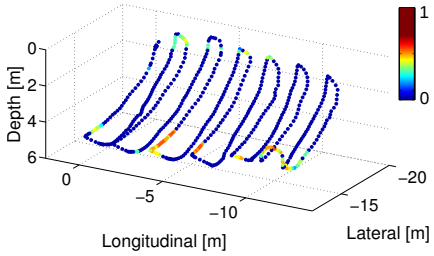
The second set of field trials on the *USCGC Escanaba* were performed at a time of day when variable sunlight penetrating the water affected the imagery collected by the HAUV, specifically relating to the salient patch located about 4 m deep. This variability is evidenced by the trajectories and time elevation plots shown in Fig. 3.13 compared with those in Fig. 3.12. Still, the deterministic and proposed methods perform favorably with respect to uncertainty and path length, shown in Fig. 3.13(j). In this set of trials, the proposed method selects revisit actions solely along the deepest portion of the environment, where the imagery is less affected by sunlight but still possesses good loop-closure utility. Here, the proposed method results in a slightly lower average uncertainty than the deterministic method.

3.6 Discussion

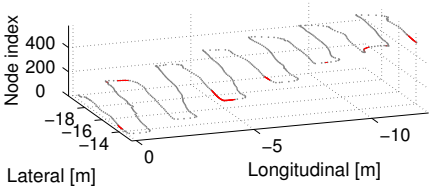
3.6.1 Performance of Planning Events

Here, we further investigate the planning algorithm within the proposed framework. For the *USCGC Escanaba* field trials, we report detailed performance statistics for each planning event in Table 3.3. The second and third columns of this table report the number of candidate plans propagated (the number of calls to `PropagatePath`() in Algorithm 3) and the number of candidates processed, or popped off the queue (calls to `Pop`(Q) in Algorithm 3), respectively. Timing statistics are reported in the fourth and fifth columns. The fourth column displays the total time spent planning for an event (triggered to exit after 20 s, upon completion of the current iteration). In the fifth column, the timing results for best path selection show that the sampling-based algorithm often returns a viable solution quickly, and can use any remaining

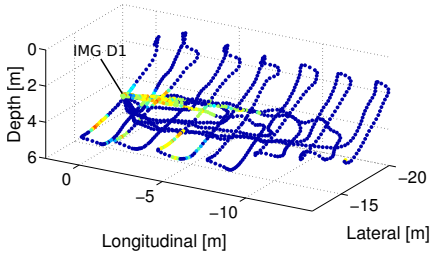
Figure 3.12 Results from the first set of field trials performed while surveying the *USCGC Escanaba* in Boston, MA. The open-loop survey is shown in (a) and (b). The deterministic strategy (c) (d) relies on large loop-closures near the beginning of the mission to bound the uncertainty. The proposed method (e) (f) yields a trajectory with loop-closures throughout the salient portions of the environment. Sample images from the deterministic and proposed strategies are displayed in (g), (h), and (i). The uncertainty versus path length plots are shown in (j). Visual loop-closures are represented by red links on the time elevation plots.



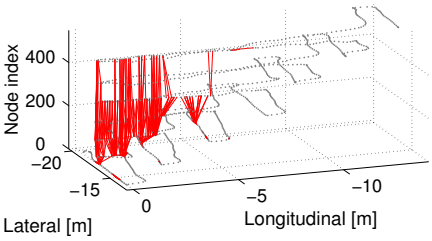
(a) Trajectory—Open-loop



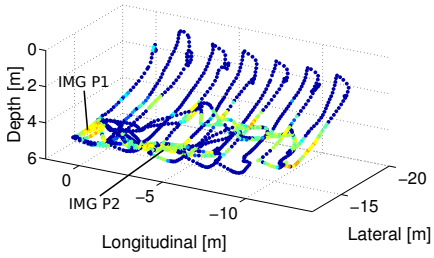
(b) Time Elevation



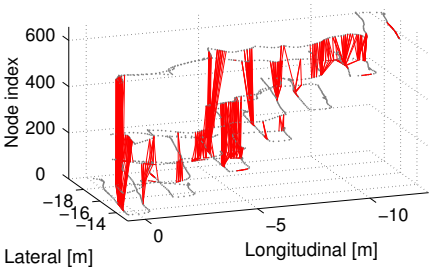
(c) Trajectory—DET



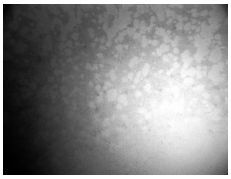
(d) Time Elevation



(e) Trajectory—Proposed

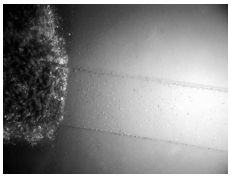


(f) Time Elevation

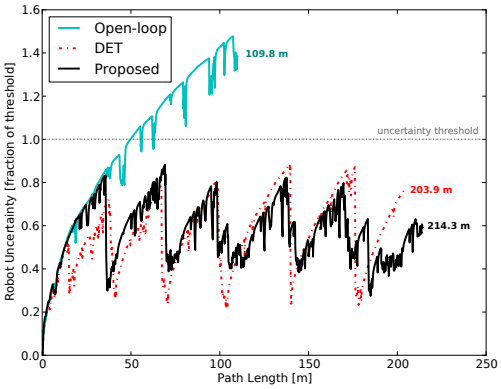


(g) IMG D1

(h) IMG P1

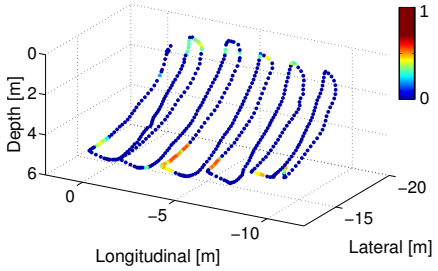


(i) IMG P2

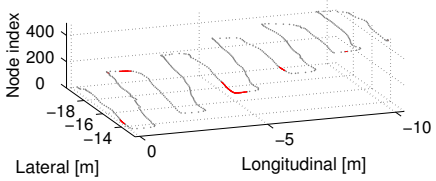


(j) Uncertainty vs. Path Length

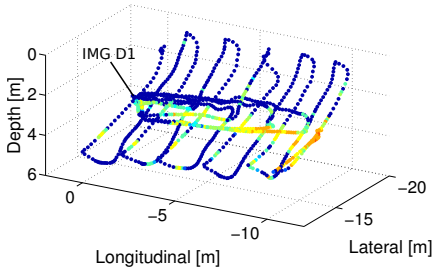
Figure 3.13 Results from the second set of field trials performed while surveying the *USCGC Escanaba* in Boston, MA. The open-loop survey is shown in (a) and (b). The deterministic strategy (c) (d) still makes loop-closures despite imagery affected by variable sunlight during the mission. The proposed method (e) (f) selects revisit paths at the deepest part of the environment where sunlight has less effect on the imagery. Sample images from the deterministic and proposed strategies are displayed in (g), (h), and (i). The uncertainty versus path length plots are shown in (j). Visual loop-closures are represented by red links on the time elevation plots.



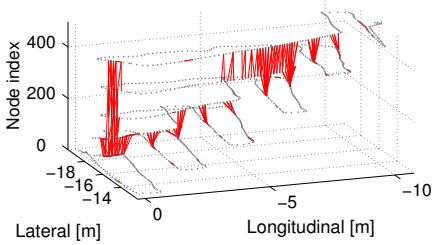
(a) Trajectory—Open-loop



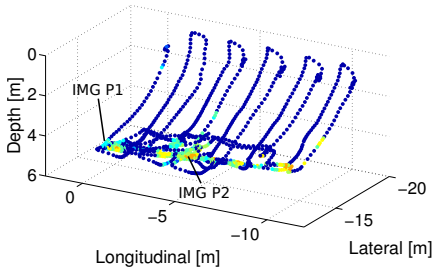
(b) Time Elevation



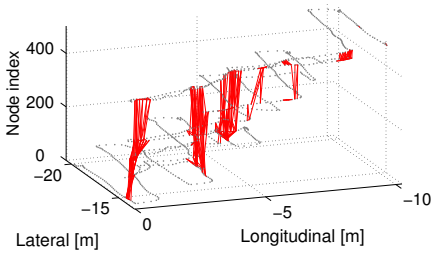
(c) Trajectory—DET



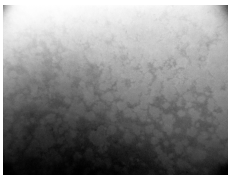
(d) Time Elevation



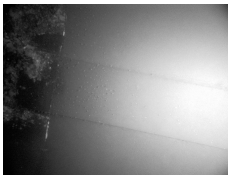
(e) Trajectory—Proposed



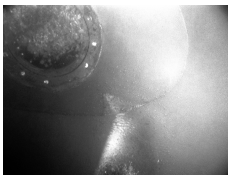
(f) Time Elevation



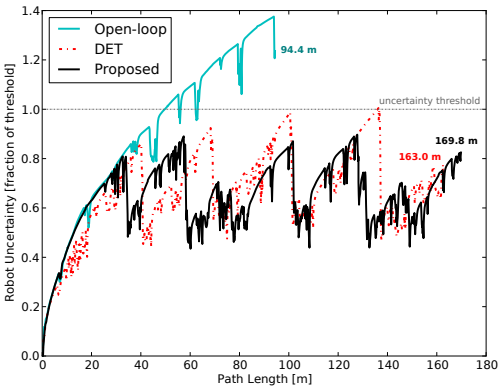
(g) IMG D1



(h) IMG P1



(i) IMG P2



(j) Uncertainty vs. Path Length

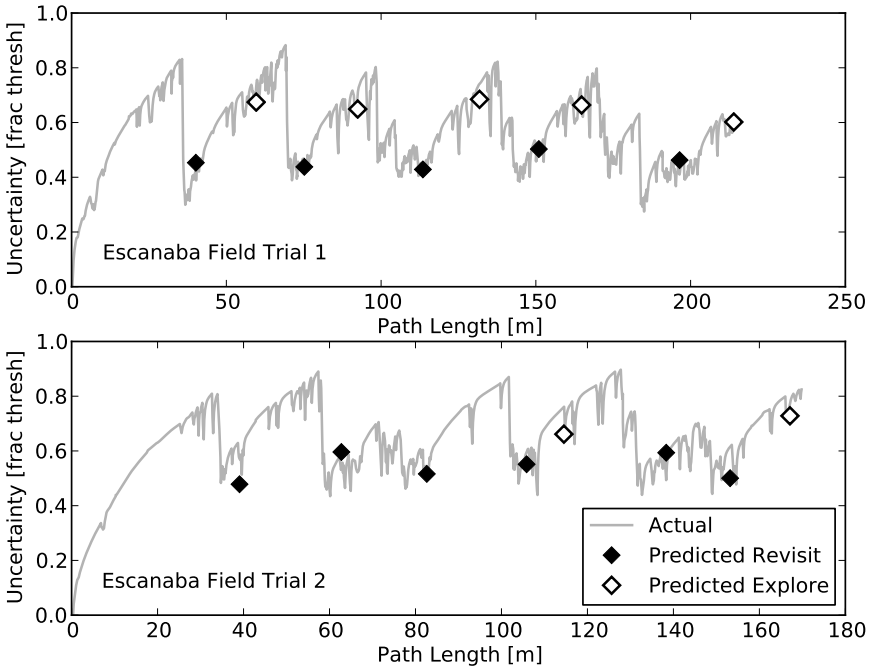
Table 3.3 *USCGC Escanaba* Planning Events

Plan No.	No. Candidates		Time Elapsed [s]		Action Selected	Predicted Uncertainty [% of $\mathcal{T}$]
	Propagated	Processed	Total	Best Path		
USCGC Escanaba Field Trial 1						
1	2524	862	4.8	0.344	Revisit	45.3
2	3528	1211	10.3	0.144	Explore	67.5
3	3445	1247	13.2	0.449	Revisit	43.8
4	2856	1238	23.0	0.262	Explore	64.9
5	3452	1185	21.3	12.526	Revisit	42.9
6	2442	991	20.6	0.270	Explore	68.5
7	2100	944	20.1	12.909	Revisit	50.3
8	1724	726	23.3	0.302	Explore	66.4
9	1866	694	20.6	0.396	Revisit	46.3
10	976	459	20.5	0.576	Explore	60.2
USCGC Escanaba Field Trial 2						
1	1602	661	2.7	1.206	Revisit	47.8
2	3074	945	8.9	6.960	Revisit	59.6
3	3442	1244	21.6	8.054	Revisit	51.6
4	2233	963	19.2	0.540	Revisit	55.1
5	1729	749	20.1	0.286	Explore	66.1
6	2377	973	23.2	4.884	Revisit	59.3
7	1031	503	20.4	3.414	Revisit	50.0
8	1361	695	20.3	0.337	Explore	72.8

time to search for improvements. The average time to find the selected best path is 2.992 s for these events. Other statistics reported include the resulting action and the predicted uncertainty of the selected best path. Regardless of the resulting decision—following the nominal exploration policy or diverting from this policy to gather loop-closure registrations—the planning process relies upon accurate prediction of the information expected along candidate paths.

Fig. 3.14 displays the results of the predictions from each plan for the hull-relative field trials. The predicted path length and uncertainty of the selected best path from each planning event is overlaid on the actual data recorded. It can be seen that the planner accurately predicts the performance of the actual robot during the survey. These results support the planning formulation of Chapter 2 and evaluation method of §3.2.2. The statistics validate the method of using GP regression for saliency prediction and scaling camera information in (3.5) to accurately capture the stochasticity of achieving registrations (when the number of link proposals is large).

Figure 3.14 The actual robot uncertainty and path length versus the robot uncertainty and path length as predicted by the active SLAM algorithm at the time of planning, for the field trials on the *USCGC Escanaba*. The predicted outcomes for each planning event align very well with the actual data collected by the robot. Black diamonds represent selected revisit actions and white diamonds represent selecting continued exploration.



3.6.2 Parameter Discussion

The proposed method includes a number of tunable parameters to provide the user with finer control and aid in transferring the system to other applications and domains. We discuss some of the more important parameters here.

The uncertainty threshold $\mathcal{T}$ is a function of the robot pose covariance. In this paper, it is set to a value that illustrates sufficient performance of the system over the duration of our experiments. In the future, however, we intend for this threshold to be set according to a measure of the resulting SLAM map quality. Note, though, that setting the threshold too low can lead to losing the opportunistic nature of the algorithm since there may be no freedom in selecting continued exploration with each planning event.

The path cost tuning parameter α controls the preference for choosing paths within each planning event. A value of α near 1 means that paths with the most beneficial drops in uncertainty are preferred over paths with shorter revisit lengths. An α near 0 means that shorter diversions are preferred even if the uncertainty benefit is minimal. If the utility of

the environment allows, setting α near 1 generally causes the robot to execute fewer, larger loop-closure actions that drop the uncertainty well below the threshold. Setting α near 0 causes the robot to gather many small loop-closures to minimally satisfy the uncertainty constraint.

The parameter β defines the degree to which the algorithm is opportunistic. Setting $\beta = 1$ provides a linear relationship to govern acceptable loop-closure actions with respect to the robot's exploration uncertainty and the uncertainty threshold. The opportunistic nature of the algorithm decreases as the value of β is increased, such that a very high β causes the robot to select revisit actions only when the uncertainty threshold is reached. For our experiments, we found that $\beta \in [10, 10000]$ reflected sensible opportunistic behavior. Refer to Fig. 3.4 for more information on β.

3.7 Chapter Summary

In this chapter, we proposed a comprehensive active SLAM framework for underwater ship hull inspection. We presented a path planning algorithm integrated with SLAM in order to find and execute loop-closure revisit actions for a robot exploring an *a priori* unknown underwater environment subject to an acceptable uncertainty threshold. We combined a sampling-based planning approach for efficiently exploring the configuration space with the planning formulation of Chapter 2 for tracking beliefs over candidate revisit paths. Finally, we developed an opportunistic approach for selecting the best revisit path that allows the robot to autonomously execute useful loop-closure actions at any point during the mission, while still exploring the target area in efficient time. The proposed method was demonstrated using a hybrid simulation with both synthetic and real camera imagery, and using real-world field trials of a robot performing seafloor and hull-relative visual inspections. The results showed that, on the whole, our framework offers many benefits over other representative active SLAM methods: principled and accurate saliency prediction, the ability to search and evaluate hundreds of candidate paths, opportunistic path selection, and consistent good performance over our experimental trials. This type of active SLAM system is useful to accomplish robust, long-term autonomy in marine environments.

Chapter 4

Efficient Planning through Compression and Structure

4.1 Introduction

Simultaneous localization and mapping (SLAM) has been successful in recent years for mobile robots operating in a wide variety of challenging environments [60]. However, solving the SLAM problem becomes increasingly expensive as the number of poses in the graph grows with space and time [102]. This computational complexity is compounded within active SLAM—a planning algorithm typically evaluates many possible candidates before selecting one to execute. Each of these evaluations entails solving a simulated SLAM system by predicting the belief of the robot given a sequence of control actions and incorporating the expected measurements along the way, as we outlined in Chapter 2. Thus, SLAM complexity can be a barrier to achieving *online* planning for autonomous robotic tasks.

In this chapter, we show that planning for active SLAM has potential in (*i*) *compressing* the representation to form an approximate distribution, and (*ii*) leveraging the *structure* of the problem, in order to evaluate candidate plans efficiently. We propose these ideas by investigating state-of-the-art algorithms from the SLAM community and applying them to the planning problem. First, we propose online graph sparsification through generic linear constraints (GLCs) [14] to compress the representation of the SLAM system in order to perform active SLAM evaluations with a smaller, approximate distribution. Second, we propose the use of the Bayes tree data structure [74] to track the SLAM system and design an active SLAM framework that takes advantage of unique characteristics of the Bayes tree when planning. These ideas are both motivated by a common goal—to improve the efficiency of expensive information-theoretic evaluations within active SLAM.

4.1.1 Related Work

Active SLAM and related belief-space planning problems consider information-theoretic objective functions to make decisions about control actions the robot should execute [67, 126, 134]. For Gaussian beliefs, evaluating this objective is very expensive however, as it generally involves inverting the associated information matrix and recovering covariance entries. Active SLAM methods in the past have have dealt with expensive evaluations by either considering a small number of candidates [80, 117, 122], approximating the problem by compressing the representation [66, 122], or aggressively pruning the search space [20, 57]. These approaches often solve SLAM as a batch operation, for example by constructing the information matrix and performing Cholesky decomposition [20, 67, 80].

Research in SLAM to reduce computational complexity is largely motivated by extending the standard formulation to better handle long-term applications that cover large environments or long durations. To address increasing computation, many recent works have sought to find efficient solutions by constructing approximate distributions of the full SLAM system. These works reduced computational overhead in the approximate distributions by enforcing sparsity in the information matrix or removing pose nodes to decrease the dimensionality of the pose SLAM system.

Thrun et al. [120] enforced sparsity in the information matrix by removing weak links (i.e., those with little information content). Vial et al. [128] enforced sparsity by performing an optimization to minimize the Kullback-Leibler divergence (KLD) between the original information matrix and an information matrix with a prescribed sparsity pattern. Their method also ensured that the approximate distribution is conservative.

Other works used pairwise measurement composition to remove nodes from the SLAM graph [70, 82, 83]. Unfortunately, measurement composition results in inconsistent estimates and is not well-defined for graphs with low-rank measurements. Instead of measurement composition, Folkesson and Christensen [45], Huang et al. [62], and Carlevaris-Bianco et al. [14] proposed methods for removing poses and replacing the corresponding measurements with linearized potentials over the elimination cliques. These methods avoid the disadvantages of measurement composition and can be augmented to produce sparsity (for instance, with a Chow-Liu tree (CLT) [14]).

In this chapter, we use a state-of-the-art graph sparsification method called GLC [14]. GLC performs marginalization to remove nodes from a SLAM graph but replaces the removed factors with an equivalent n-ary factor that induces the exact same information into the graph. To maintain sparsity, GLC can instead produce a set of unary and binary factors that form an approximate sparsified distribution. This method features many advantages over other similar sparsification methods. For example, GLC is able to handle low-rank constraints

and does not double-count information. It scales well to removing nodes in large graphs because it depends only on the size of the elimination clique and the information contained within the corresponding factors. Also, GLC allows for relinearization within standard SLAM optimizers by optionally expressing each linear factor with a local-frame linearization.

Recently, Indelman [65, 66] proposed using a conservative approximation for information-theoretic decision-making. He showed that operating on a decoupled conservative distribution is guaranteed to produce the same decision-making results as operating on the original distribution under certain circumstances, like in typical sensor deployment problems. Motivated by this concept, we show that using GLC to create a sparsified approximate distribution allows us to plan over a multi-step horizon in active SLAM while achieving computational savings and maintaining the same decision-making performance as planning with the original distribution.

Aside from research in compressing the representation, other work in solving SLAM efficiently has examined the structure of the problem. Considering the linear algebra perspective, Kaess et al. [73] used QR factorization of the measurement Jacobian instead of Cholesky factorization of the information matrix. A benefit of this approach is that the solution can be incrementally updated as new measurements arrive by applying Givens rotations directly to the square-root information matrix. From the upper triangular form, the SLAM solution is easily found by backsubstitution.

Loop-closures in SLAM directly correspond to updates in the square-root information matrix that cause significant fill-in (additional nonzero entries in the matrix). Much of the fill-in can be avoided by periodically *reordering* the variables in the problem [1]. The concept of efficient ordering with respect to graphical models was further explored by Kaess et al. [74], leading to the development of the Bayes tree structure for efficient inference. Polok et al. [107] extended the advantages of the Bayes tree to the sparse linear algebra setting. In this chapter, we leverage the use of the Bayes tree in an active SLAM framework to accomplish efficient evaluations of candidate actions.

Recently, Ila et al. [64] proposed a novel approach to efficiently recover marginal covariance matrices by performing incremental updates when measurements are added to the system and the linearization point remains unchanged. Like this approach, we are interested in investigating the structure of the problem to perform fast evaluations of information-theoretic objectives.

4.1.2 Preliminaries

We frame our problem in the context of active visual SLAM with a robot maintaining a pose graph representation of its trajectory through the environment. Visual features in the environment are used to provide pairwise constraints as loop-closing measurements between two poses in the graph with overlapping image views. The robot is tasked with performing a full-coverage survey of the environment given a nominal exploration policy that covers the target area in efficient time. However, open-loop execution of this policy leads to unbounded uncertainty growth as no large loop-closures are contributed. To reduce the uncertainty in the SLAM posterior, the robot must divert to revisit previously-observed portions of the environment in order to close loops in the pose graph. The active SLAM framework searches for candidate loop-closure paths continuously throughout the mission, evaluating each for their expected benefit in terms of efficient area coverage and uncertainty reduction.

Following the planning formulation of Chapter 2, we define the belief of the robot at a given planning step $k \in [1, K]$ as

$$\mathcal{B}_k = p(X_k | \mathcal{Z}_0, \mathcal{U}_0, Z_{1:k}, \Gamma_{1:k}, U_{0:k-1}), \tag{4.1}$$

where X_k is the state vector of interest and $\mathcal{Z}_0$ and $\mathcal{U}_0$ are the prior measurements and controls, respectively, up to the planning event. $Z_{1:k}$ are the random update measurements and $\Gamma_{1:k}$ are the corresponding random acquisition variables. The belief vector is represented by a multivariate Gaussian with mean and covariance matrix (given in terms of the inverse information matrix) found using the maximum *a posteriori* (MAP) estimate,

$$\mathcal{B}_k \sim \mathcal{N}\left({X_k}^*, {\Lambda_k}^{-1}\right). \tag{4.2}$$

We stated previously in Chapter 2 that this brings us to the nonlinear least-squares problem common to graph-based SLAM [29],

$$\begin{aligned} {X_k}^* = \arg\min_{X_k} \Big[\|X_0 - {X_0}^*\|^2_{\Lambda_0} + \sum_{i=1}^{k} \|\mathbf{x}_i - f(\mathbf{x}_{i-1}, \mathbf{u}_{i-1})\|^2_{\Omega_w} + \\ \sum_{i=1}^{k} \sum_{j=1}^{n_i} \gamma_{i,j} \left\| \mathbf{z}_{i,j} - h\left(X_i^j\right) \right\|^2_{\Omega_v^{i,j}} \Big], \end{aligned} \tag{4.3}$$

where both $\gamma_{i,j}$ and $\mathbf{z}_{i,j}$ are random. Linearizing about the nominal mean estimate, $\bar{X}_k(U_{0:k-1}) = \{{X_0}^*, \bar{\mathbf{x}}_1, \ldots, \bar{\mathbf{x}}_k\}$, found by compounding the set of controls, the problem is expressed as the

linear least-squares optimization for state update vector ΔX_k:

$$\underset{\Delta X_k}{\arg\min}\Big[\|\mathcal{A}_k(\Gamma_{1:k}, U_{0:k-1})\Delta X_k - \boldsymbol{b}_k(Z_{1:k}, \Gamma_{1:k}, U_{0:k-1})\|^2\Big], \tag{4.4}$$

where $\mathcal{A}_k$ is the weighted sparse measurement Jacobian and $\boldsymbol{b}_k$ is the weighted residual vector (both including the acquisition variables).

4.2 Sparsification for Approximate Distributions

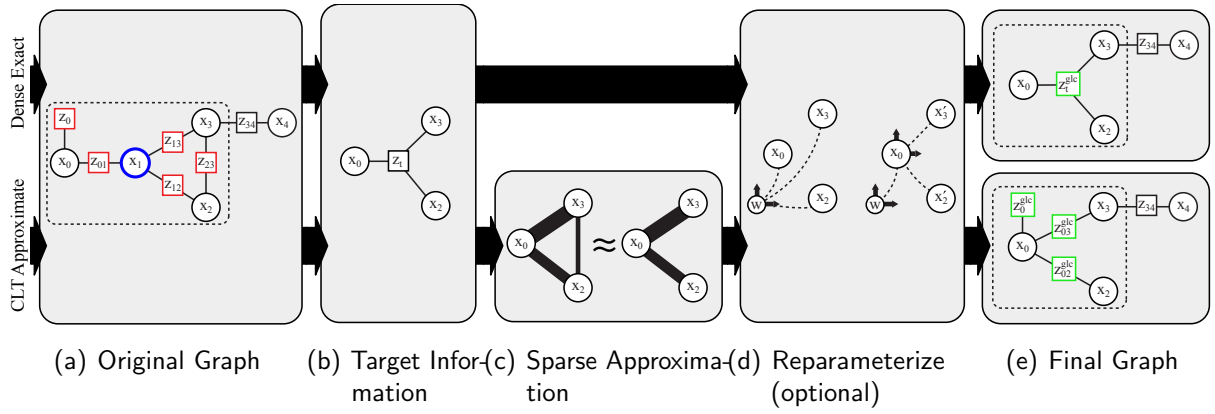
In this section, we investigate improving the efficiency of planning within active SLAM through *compression* of the representation. Our approach is to reduce the dimensionality of the underlying SLAM system by incrementally removing pose nodes in the factor graph through online graph sparsification. This process allows us to construct an approximate distribution of the full SLAM system that is useful for planning. Our proposed algorithm operates by removing and repairing nodes using GLCs [14]. The basis of this work is a concept recently introduced by Indelman [65]: we can use an approximate distribution in place of the original distribution for many information-theoretic planning applications. The benefit of using the sparsified approximate distribution is that we can achieve significant computational savings during the evaluation of candidate actions with little to no sacrifice in the ordering of decision-making outcomes. For planning purposes, the objective function values themselves do not need to be the same between the original and approximate distributions. We can make the same decisions at each planning event by only preserving the relative ordering of candidate plans.

At the end of the planning horizon, we evaluate the resulting belief $\mathcal{B}_K$ using an objective function that quantifies some uncertainty measure; here, we focus on (*i*) the determinant of the final pose covariance, (*ii*) the trace of the final pose covariance, and (*iii*) the mutual information gain [63] over the entire distribution:

$$\begin{aligned} (i) \quad & J = \left|{\Lambda_K}^{-1}\right|, \\ (ii) \quad & J = tr\left({\Lambda_K}^{-1}\right), \\ (iii) \quad & J = \frac{1}{2}\log\frac{|\Lambda_0 + \Delta\Lambda|}{|\Lambda_0|}, \end{aligned} \tag{4.5}$$

where $\Delta\Lambda$ is the information added by the candidate action to the prior information Λ_0. Each of these objectives requires evaluation of the associated information matrix, Λ_K. However, in this section we propose evaluating these objectives using a sparsified approximate distribution ${\mathcal{B}_K}'$ with information matrix ${\Lambda_K}'$. We construct ${\Lambda_K}'$ by replacing the prior SLAM information Λ_0

Figure 4.1 Graph sparsification via GLC. Figures from Carlevaris-Bianco et al. [14]. (a) Identify the node to be removed (blue) and its Markov blanket. (b) Construct the target information by marginalizing out the node to be removed. (c) If desired, use the CLT to form a sparse approximation of the target information. (d) Optionally perform a root-shift operation to reparameterize the resulting linear potential in terms of a relative frame. (e) Form the GLC factor (green) and insert into the factor graph.



in (4.3) with $\Lambda_0{}'$, an approximate prior distribution resulting from online graph sparsification.

We present our proposed method by providing a short background on GLC and then describing the online graph sparsification algorithm.

4.2.1 Generic Linear Constraints

For efficient planning in active SLAM through compression, we propose sparsifying the SLAM graph online using GLC. GLC is a framework for performing approximate marginalization in factor graphs that replaces the removed factors with one or more generic linear constraints. An overview of the method is presented in Fig. 4.1. The first step in removing a node with GLC is to identify and construct the target information, Λ_t. The target information is defined by the factors in the elimination clique—that is, the node to be removed plus the nodes in its Markov blanket. The target information is the information that results after marginalizing out the node to be removed, represented by an n-ary factor over the remaining nodes in the elimination clique. This factor induces the exact same information as the original factors at the current linearization point. It is important to note that the target information may be singular if low-rank measurements are included, as is the case with 5-degree of freedom (DOF) camera registrations.

The n-ary factor that encodes the resulting potential is called a *generic linear constraint*. Since this factor is possibly low-rank, its observation model is found using the eigendecomposition and pseudoinverse of the target information. The eigendecomposition is given

by

$$\Lambda_t = V D_t V^\top, \tag{4.6}$$

where D_t is a square diagonal matrix of the rank of Λ_t. Then, the observation model of the GLC factor is written as

$$\begin{aligned} \mathbf{z}_{\texttt{glc}} &= G\mathbf{x}_t + \mathbf{w}, \\ \mathbf{w} &\sim \mathcal{N}\left(\mathbf{0}, {\Lambda_{\texttt{glc}}}^{-1}\right), \end{aligned} \tag{4.7}$$

where $\mathbf{x}_t$ is the vector of variables remaining in the elimination clique and $G = {D_t}^{\frac{1}{2}} V^\top$. The GLC factor information is computed with the pseudoinverse of the target information:

$$\Lambda_{\texttt{glc}} = \left(G {\Lambda_t}^{+} G^\top\right)^{-1} = \left(G \left(V {D_t}^{-1} V^\top\right) G^\top\right)^{-1}. \tag{4.8}$$

The GLC factor induces the same potential into the graph as the original factors from the elimination clique. By performing the above process with optional additional root-shift operations, the factor can be transformed in order to avoid inconsistencies related to world-frame linearization.

Unfortunately, exact marginalization causes dense fill-in in the information matrix, which increases complexity. To maintain sparsity with node removal, the dense GLC factor can be sparsely approximated using a CLT [25]. In GLC, the CLT approximates the joint distribution Λ_t as the product of pairwise conditional distributions and is constructed such that the KLD is minimized. This is done by finding the maximum spanning tree over all possible mutual information pairings within the elimination clique. Then, the unary and binary potentials from the CLT are represented by GLC factors, rather than introducing one n-ary factor.

The experimental evaluation of GLC on large SLAM datasets [14] presents promising results for reducing complexity while maintaining approximate distributions, which will be useful for planning. GLC reports an average node removal time on the order of 10 ms and good KLD values even for the sparse approximate method with over 80% of nodes removed.

4.2.2 Online Graph Sparsification

Our approach is to reduce the dimensionality of the SLAM system by using GLC to remove nodes in the factor graph while preserving (as much as possible) the original information content. The method is fairly straightforward but yields large dividends in planning for active SLAM. We incrementally construct the approximate prior SLAM information ${\Lambda_0}'$ with an online sparsification algorithm. The algorithm is given in Algorithm 4 and involves two principle processes: (i) online node removal using GLC, found in the function `RemoveRecentNodes`(); and (ii) online node repair, found in `RepairNode`(n).

Algorithm 4 Online Graph Sparsification

Initialize: $\mathcal{N} = \{\}, \mathcal{R} = \{\}, \mathcal{A} = \{\}$

while (SLAM is not finished) **do**
  f_i = `ListenForIncomingFactor`()
  $\{n_i\}$ = `GetNodesFromFactor`(f_i)
  for (n_i in $\{n_i\}$) **do**
    if ($n_i \in \mathcal{N}$) **then**
      `RepairNode`(n_i)
    end if
  end for
  if (time elapsed $> \Delta t$) **then**
    `RemoveRecentNodes`()
  end if
end while

Function: `RemoveRecentNodes`()

$\{n_r\}$ = `GetRecentNodesToRemove`()
for (n_r in $\{n_r\}$) **do**
  $\{f_r\}$ = `GetFactorsToRemove`(n_r)
  $\{f_a\}$ = `CalculateGLCFactorsToAdd`($\{f_r\}$)
  `RemoveFromSLAM`($n_r, \{f_r\}$)
  `AddToSLAM`($\{f_a\}$)
  $\mathcal{N} = \mathcal{N} \cup n_r, \mathcal{R} = \mathcal{R} \cup \{f_r\}, \mathcal{A} = \mathcal{A} \cup \{f_a\}$
end for
return

Function: `RepairNode`(n)

$\{f_a\}$ = `GetPreviouslyAddedFactors`(n)
$\{f_r\}$ = `GetPreviouslyRemovedFactors`(n)
`AddToSLAM`(n)
$\mathcal{N} = \mathcal{N} \setminus n$
for (f_r in $\{f_r\}$) **do**
  $\{n_i\}$ = `GetNodesFromFactor`(f_r)
  for (n_i in $\{n_i\}$) **do**
    if ($n_i \in \mathcal{N}$) **then**
      `RepairNode`(n_i)
    end if
  end for
end for
`RemoveFromSLAM`($\{f_a\}$)
`AddToSLAM`($\{f_r\}$)
$\mathcal{R} = \mathcal{R} \setminus \{f_r\}, \mathcal{A} = \mathcal{A} \setminus \{f_a\}$
return

The proposed algorithm proceeds as follows. During SLAM, we periodically remove a large percentage of the recent nodes using sparse-approximate GLC. In our experiments with the Hovering Autonomous Underwater Vehicle (HAUV) hybrid simulation, we removed all new nodes added to the factor graph over the previous 20 s except the most recent node. In this way, the approximate prior distribution resembles a "skeleton" of the original SLAM system, with GLC factors connecting the remaining nodes in the graph.

We found this incremental removal process to be extremely efficient, on the order of tens of milliseconds to remove all recent nodes, such that these periodic operations are easily performed online and in the background throughout the duration of the mission. Here, we can see one reason why the proposed method is useful for planning: online sparsification essentially precomputes many of the necessary operations related to candidate plan evaluation. Large matrix operations (like Cholesky factorization of the corresponding information matrix—required for covariance recovery and calculating mutual information) are amortized over the length of the mission. Marginalization that is performed once during node removal is used over and over when planning with the approximate, condensed distribution.

Of course, online sparsification during SLAM has its disadvantages. One common consequence is that nodes previously removed from the graph are involved in future loop-closure measurements. Thus, we propose a simple node repair process for reinserting removed nodes back into the graph when necessary.

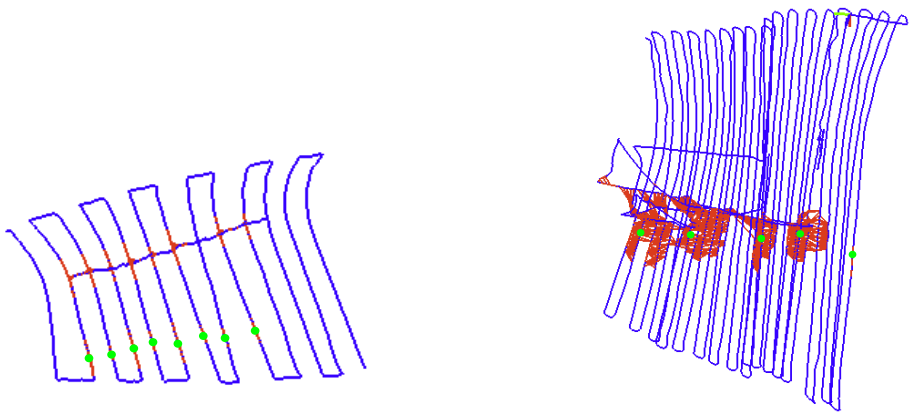
The repair process works by undoing the associated GLC node removal procedure, showcasing one benefit of using GLC for sparsification. Since GLC operates only on the local target information over the elimination clique, it is straightforward and efficient to replace the original node and factors into the graph and remove the corresponding GLC factors. Note that no calculations are required to recover the original node and factors, only storing these components upon their initial removal and looking them up upon the repair request. Indeed, by construction, swapping the GLC factors for the original factors is a near-equivalent swap of information content. By repairing the node of interest, the loop-closure measurement can be incorporated and SLAM continues as normal.

From here, we can perform decision-making at any point during SLAM by taking a snapshot of the sparsified distribution in the form of the information matrix, $\Lambda_0{}'$, and solving the simulated SLAM problem of (4.3) for each sequence of candidate control actions.

4.3 Online Graph Sparsification Results

We demonstrate our proposed method using a hybrid simulation derived from datasets collected by the HAUV performing visual SLAM in order to inspect the underwater portion

Figure 4.2 Trajectory profiles for the experiments using data collected by an underwater ship hull inspection robot. We design synthetic environments that are visually-salient only in select regions, designated by the visual loop-closures shown in red. The candidate revisit pose locations are overlaid on each graph in bright green.



(a) Sonar-spaced inspection

(b) Camera-spaced inspection

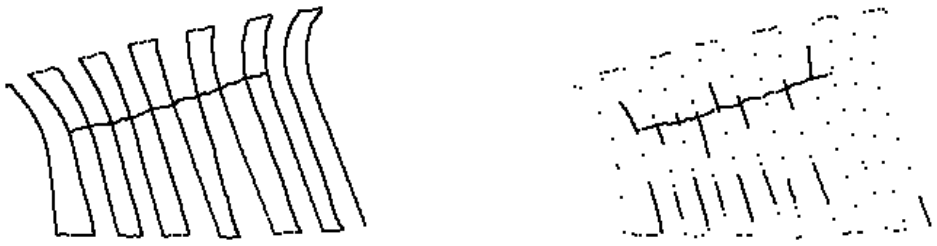
of a ship hull. We design synthetic environments that are visually-salient only in select portions of the map. More information on the underwater robot and our synthetic experimental setup can be found in Chapter 3 [20, 80].

4.3.1 Evaluation with Sonar-spaced Inspection

In the first experiment, the robot surveys the environment following the sonar-spaced trajectory profile shown in Fig. 4.2(a). Every 20 s throughout the mission, the online sparsification algorithm removes recent nodes. In between sparsification requests, the robot initiates a planning event for active SLAM with the goal of reducing its uncertainty. For each planning event (96 events in total), loop-closing revisit paths to 8 previous poses in the graph are evaluated and ranked according to their uncertainty benefit. The original and sparsified distributions for this experiment are shown in Fig. 4.3.

Fig. 4.4 summarizes all active SLAM candidate path evaluations throughout the mission. Using both the approximate and original distributions, we evaluate the determinant of the final pose covariance, trace of the final pose covariance, and mutual information gain of the distribution for each candidate, described by (4.5). Fig. 4.4(d), (e), and (f) show that, in general, evaluations with the approximate distribution found using the online sparsification algorithm follow the same trend as evaluations with the original distribution. Computational savings for recovering the final pose covariance and calculating the mutual information gain are shown in Fig. 4.4(b) and (c).

Figure 4.3 The original (a) and sparsified (b) graphs for the sonar-spaced inspection experiment.



(a) Original (b) Sparsified

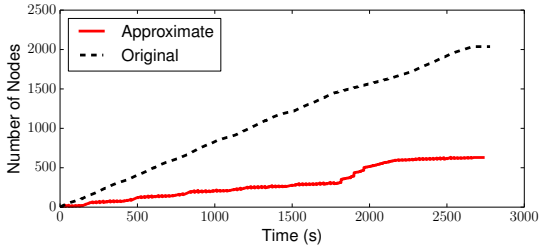
Importantly, we see in Fig. 4.5(a), (b), and (c) that the ranking of candidate plans for all 96 planning events is nearly identical regardless of whether the approximate distribution or original distribution is used for planning. For this experiment, 88.0% of all evaluations across the three objective functions maintained the same ranking. Results are improved when considering that only the top-ranked option is of interest for planning; 92.0% of top-ranked choices are preserved. On average, we achieved computational savings of 17.2 ms per covariance recovery and 117.5 ms per information gain calculation, a reduction of 55.0% and 66.5% in evaluation time, respectively. These savings are directly a result of reducing the dimensionality of the prior SLAM system with sparse-approximate GLC.

4.3.2 Evaluation with Camera-spaced Inspection

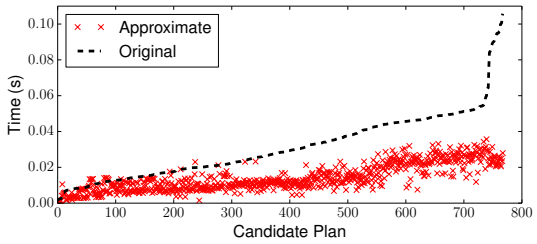
The second experiment is similar to the first except in this case the robot surveys with tracklines providing dense camera coverage of the environment. This mission profile results in a larger SLAM graph with greater connectivity compared to the first experiment, as shown in Fig. 4.2(b). An overview of the original and sparsified distributions is shown in Fig. 4.6. In this experiment, online sparsification yields a reduction in the dimension of the information matrix from $61,326$ in the original distribution to $5,142$ in the approximate distribution while maintaining sparsity. Throughout the mission, the active SLAM decision-making process considers 5 loop-closure revisit poses for each of the 143 planning events.

Fig. 4.7 provides summarized results for the camera-spaced inspection experiment. Like the first sonar-spaced experiment, the uncertainty-related objective evaluations generally follow the same trend between the original and approximate distributions. Once again, we see in Fig. 4.8(a), (b), and (c) that the ranking for candidate plans is largely preserved for all events, with 92.1% of candidate evaluations maintaining the same ordering, and 94.9% of top-ranked options. We also see in this experiment an average computation time savings of

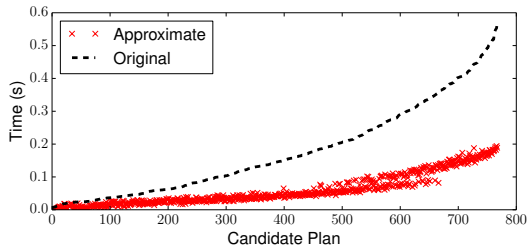
Figure 4.4 Summarized results from the sonar-spaced inspection experiment. (a) The number of nodes in the graph as a function of time. (b) Evaluation times for recovering the final pose covariance. This calculation averaged 31.3 ms and 14.1 ms with the original and approximate distributions, respectively. (c) Evaluation times for finding the mutual information gain. This calculation averaged 176.8 ms and 59.3ms with the original and approximate distributions, respectively. (d) The determinant of the final pose covariance. (e) The trace of the final pose covariance. (f) The mutual information gain over the distribution.



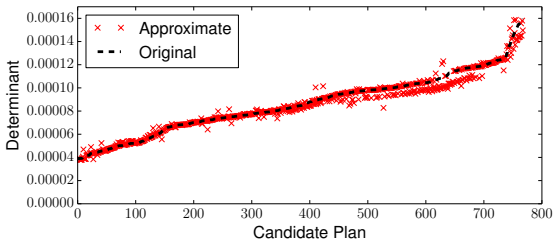
(a) Number of Nodes



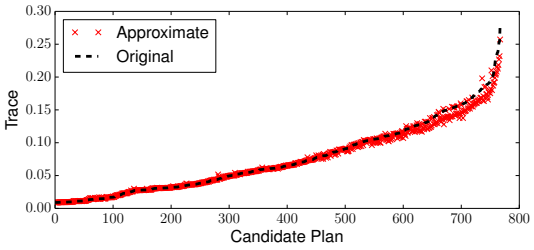
(b) Covariance Calculation Time



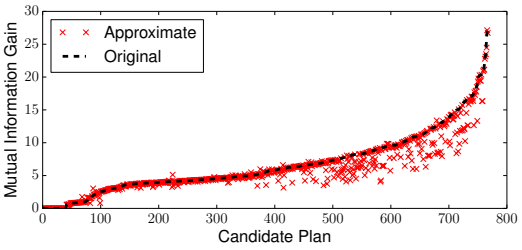
(c) Information Gain Calculation Time



(d) Determinant of Covariance

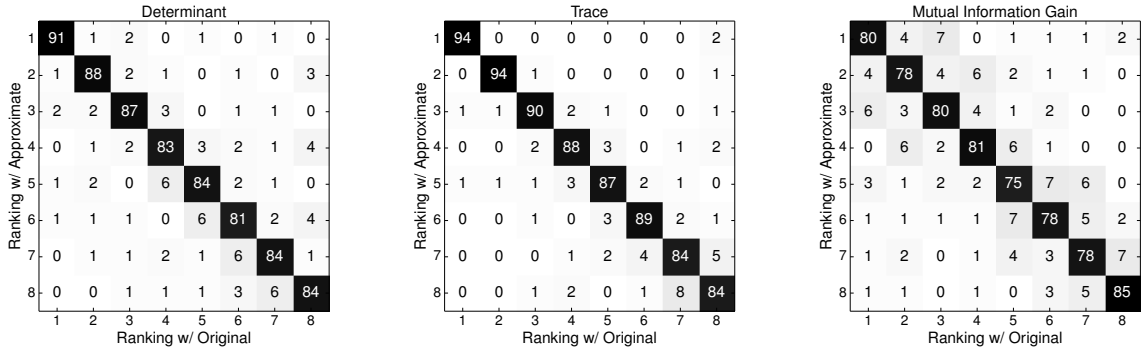


(e) Trace of Covariance



(f) Mutual Information Gain

Figure 4.5 Confusion matrices for displaying the ordering outcomes are presented in (a), (b), and (c). The 8 rows and columns correspond to the 8 candidate actions during the 96 planning events. These results show that the relative ranking of candidate plans is well-preserved when using the sparsified approximate distribution in place of the original distribution during planning.



(a) (b) (c)

42.0 ms (30.5%) for each covariance recovery and 141.3 ms (59.5%) for each information gain calculation.

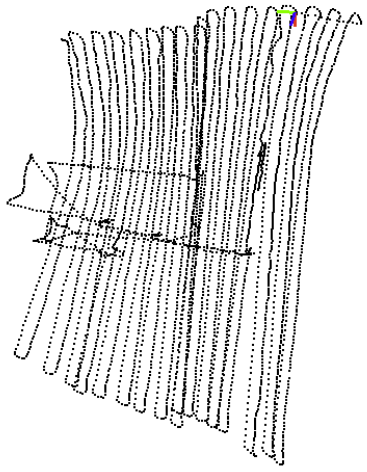
4.4 Graph Sparsification Summary

So far in this chapter, we proposed an online graph sparsification algorithm for planning with approximate distributions in active SLAM. Our method leverages sparse-approximate GLC to incrementally construct the approximate prior distribution by removing and repairing nodes online during SLAM. We demonstrated our proposed algorithm with two experiments based on an underwater robot performing visual SLAM. Planning with the approximate sparsified distribution, we showed computational savings for plan evaluations between 30% and 67% while preserving between 88% and 92% of all candidate plan rankings, and 92% and 94% of top-ranked options.

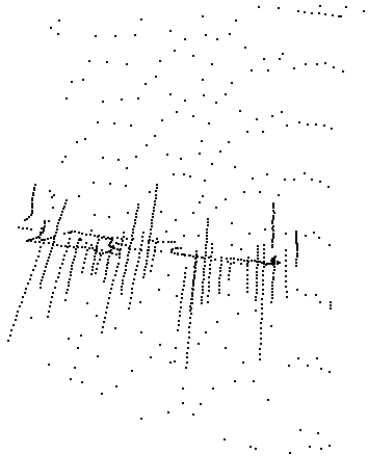
4.5 Leveraging Structure with the Bayes Tree

In this section, we transition to a new concept based on leveraging the *structure* of the planning problem to reduce computational complexity. We propose the use of the Bayes tree for planning and present an ordering of variables that facilitates fast evaluation of candidate plans that are similar. This similarity occurs, for example, when a belief is minimally changed between consecutive timesteps. The Bayes tree is a helpful structure for deciding variable orderings that are efficient when adding new measurements. We show that using

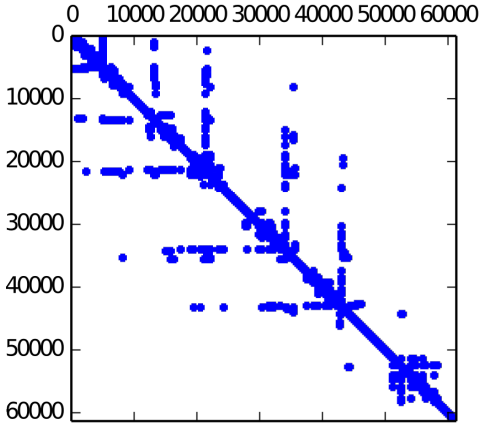
Figure 4.6 Planning with an approximate distribution found by online graph sparsification during SLAM. The original graph (a) and associated information matrix (c). The sparsified graph (b) has significant reduction in the dimension of the associated information matrix (d), and leads to computational savings during planning evaluations.



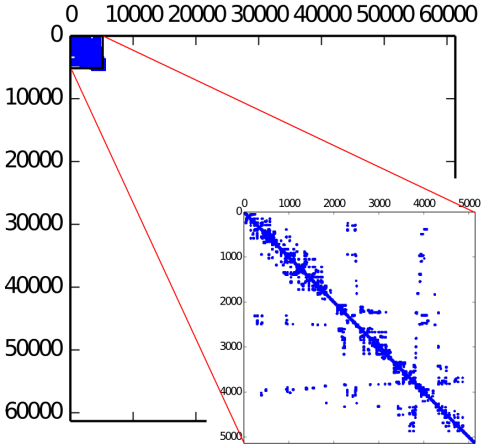
(a) Original



(b) Sparsified

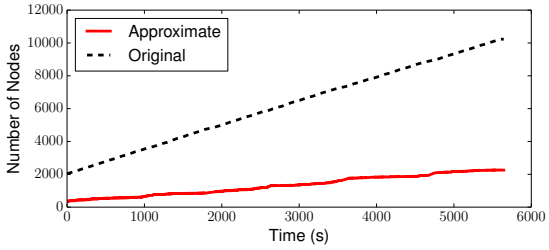


(c) dim=61326

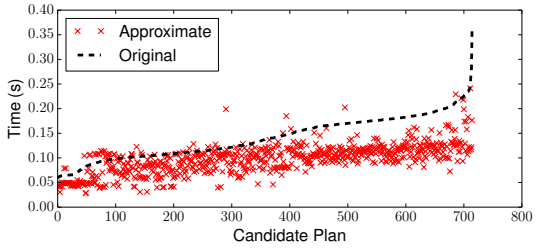


(d) dim=5142

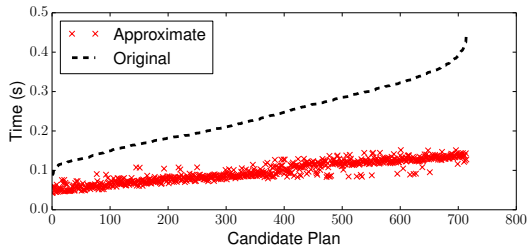
Figure 4.7 Summarized results from the camera-spaced inspection experiment. (a) The number of nodes in the graph as a function of time. (b) Evaluation times for recovering the final pose covariance. This calculation averaged 140.5 ms and 97.7 ms with the original and approximate distributions, respectively. (c) Evaluation times for finding the mutual information gain. This calculation averaged 237.6 ms and 96.3ms with the original and approximate distributions, respectively. (d) The determinant of the final pose covariance. (e) The trace of the final pose covariance. (f) The mutual information gain over the distribution.



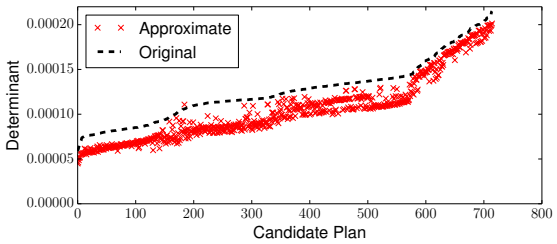
(a) Number of Nodes



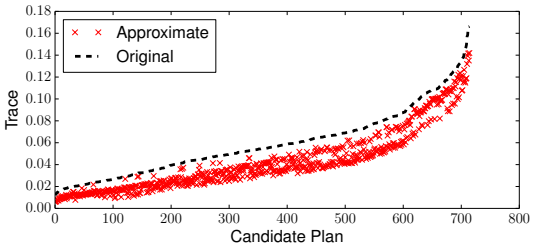
(b) Covariance Calculation Time



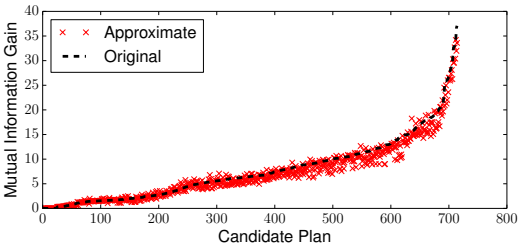
(c) Information Gain Calculation Time



(d) Determinant of Covariance

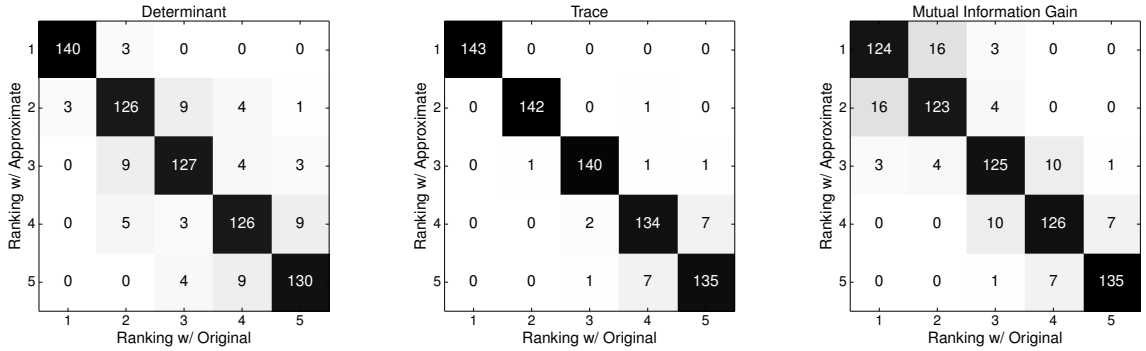


(e) Trace of Covariance



(f) Mutual Information Gain

Figure 4.8 Confusion matrices for displaying the ordering outcomes are presented in (a), (b), and (c). The 5 rows and columns correspond to the 5 candidate actions during the 143 planning events. These results show that the relative ranking of candidate plans is well-preserved when using the sparsified approximate distribution in place of the original distribution during planning.



(a) (b) (c)

this representation removes redundant computation by preserving much of the existing tree structure, allowing for fast incremental updates in order to quickly evaluate a candidate plan that is similar to a previously-evaluated candidate. Using the Bayes tree data structure for planning in active SLAM, the contributions of this section are:

1. We identify a variable ordering constraint that allows successive evaluations of a candidate to be computed efficiently between consecutive timesteps.

2. We design candidates such that they share a common Bayes tree root and present a subtree caching scheme for fast evaluations across candidates given the root.

3. We present an active visual SLAM framework for an underwater robot, building on work from Chapter 3, that leverages the ideas for planning efficiently with the Bayes tree.

Considering the problem formulation of §4.1.2, the solution to (4.4) can be calculated with sparse linear algebra by Cholesky decomposition of the information matrix $\Lambda_k = \mathcal{A}_k^\top \mathcal{A}_k$ or QR factorization of $\mathcal{A}_k$ [73]. Alternatively, Kaess et al. [74] identified that the solution can be found by interfacing directly with the corresponding graphical model using the Bayes tree data structure. We present a brief background of the Bayes tree in the next section and then show how it is used in our proposed method for efficient planning.

4.5.1 Bayes Tree

The Bayes tree encodes the relationship between the underlying graphical model and sparse matrix factorization of the SLAM problem. We can see this relationship by understanding that variable elimination on the factor graph produces a chordal Bayes net defining a product of conditional densities over the variables in the system. In the case of a Gaussian factor graph, the resulting Bayes net is equivalent to the square-root information matrix from sparse QR factorization. From this observation, Kaess et al. [74] introduced the *Bayes tree* data structure for efficient inference. Incremental updates to the SLAM system that were abstract in the matrix factorization form are instead realized in the Bayes tree as simple edits to the graphical structure.

Construction of the Bayes tree begins by first constructing the Bayes net from variable elimination on the factor graph. Eliminating a variable Θ_i from the graph results in a conditional density $p(\Theta_i|S_i)$ given the separator variables S_i. The separator variables are the variables connected to Θ_i through factors involving Θ_i. After all variables have been eliminated, the Bayes net defines the joint density of the system as a product of conditionals,

$$p(\Theta) = \prod_i p(\Theta_i|S_i). \tag{4.9}$$

The Bayes tree is constructed by discovering the cliques in the Bayes net. Each node in the tree represents a conditional density over a clique, C_j, in the chordal Bayes net, with conditional density given by $p(F_j|S_j)$. Here, F_j are the frontal variables and S_j are the separator variables, which together define the clique: $C_j = F_j \cup S_j$. The separator variables within the clique are also contained in its parent clique, Π_j, such that $S_j = C_j \cap \Pi_j$. Each directed edge in the tree points from a parent clique to one of its children and thus represents conditioning, as in the Bayes net. Together, the Bayes tree expresses the joint density of the SLAM problem as a product of the conditional densities of its nodes,

$$p(\Theta) = \prod_j p(F_j|S_j). \tag{4.10}$$

Incremental updates to the SLAM system are simple to execute within the Bayes tree. Adding a new factor is accomplished by the following steps:

1. Identify the cliques in the tree containing variables affected by the new factor.

2. Transform the paths from the root of the tree to these cliques back into a factor graph representation, and store the unaffected subtrees.

3. Insert the new factor into the factor graph.

4. Form a new Bayes net and Bayes tree from the updated factor graph using a new elimination ordering.

5. Reattach the unaffected subtrees to the new Bayes tree.

This process illustrates that only the top-most part of the tree (closest to the root) is recomputed when incorporating a new measurement, since the tree encodes the flow of information during the elimination procedure.

4.5.2 Efficient Planning with the Bayes Tree

The contributions of this section are centered around leveraging the Bayes tree as a useful data structure for planning in active SLAM.

Constrained Variable Ordering

We design each candidate loop-closure path with a fixed *entrance* pose $\mathbf{x}_e$ in the environment, such that traveling along the candidate path always proceeds through the entrance pose. In this way, the large majority of the control actions associated with a candidate do not change between consecutive evaluations. Usually, only the virtual factors between the current robot pose $\mathbf{x}_c$ and the entrance pose are modified to reflect any new measurements that were incorporated into the SLAM system between evaluations. We can achieve significant computational savings between these consecutive evaluations by intelligently constructing the Bayes tree with a variable ordering conducive to efficiently updating the candidate at sequential timesteps. A key insight is that children (subtrees) of a parent clique in the Bayes tree are conditionally independent of one another given the parent clique density. Thus, we can use the root of the tree to partition the system into conditionally independent subtrees corresponding to (i) the virtual path variables between the current SLAM node and the entrance pose of the path (X_m), and (ii) all other virtual poses on the path (X_n) and previous nodes in the SLAM graph not contained in the root (X_l). This leads to a simple rule for Bayes tree construction: we constrain the current SLAM node and entrance pose to be pushed to the root of the tree by eliminating these variables after all others in the system. In this way, the Bayes tree has the factored density

$$p(X_k) = p(X_l, X_n|X_r)p(X_m|X_r)p(X_r), \tag{4.11}$$

where $p(X_r)$ is the prior on the root and $\{\mathbf{x}_c, \mathbf{x}_e\} \in X_r$.

Conceptually, the constrained ordering heuristic is not new; for online SLAM, the iSAM2 algorithm constrains the most recently-used variables toward the top of the Bayes tree with the constrained column approximate minimum degree (COLAMD) algorithm [27, 74]. Most incremental updates to the system are therefore efficient since they typically affect only these recently-used variables. We extend this idea to the planning realm by recognizing that in planning we often know which variables will be affected in future evaluations. In addition, by designing the candidates to reuse the same control actions after the entrance pose, we assist their subsequent evaluations by allowing these portions of the simulated SLAM system to correspond to subtrees that are carried over and reused during updates. While forcing the current node and entrance pose into the root does result in some additional matrix fill-in, we found it to be computationally negligible in practice compared to the added savings.

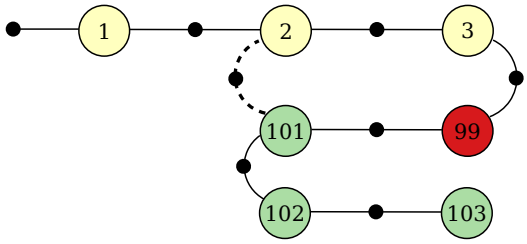
Here we examine a simple example in Fig. 4.9 to illustrate the proposed constrained ordering approach. At timestep t we have a SLAM graph consisting of three poses, $X_0 = \{1, 2, 3\}$. Beginning with the current node 3 we plan a path that travels through the entrance pose 99 and provides a loop-closure measurement between pose 2 and virtual pose 101. We solve the system at t with the constrained ordering that eliminates 3 and 99 last. The resulting Bayes tree is shown in Fig. 4.9(f) with the green and yellow subtrees together defining the density $p(X_l, X_n|X_r)$ for this candidate.

At timestep $t + 1$, measurements are added to the system and pose 4 becomes the current SLAM node. To reevaluate the path from the previous timestep, we remove the factor corresponding to the control from 3 to 99 and augment the system with factors required to transition from pose 4 to 99. Solving the new system is a simple update with the Bayes tree. The top-most part of the tree containing cliques involving affected variables is removed and converted back into a factor graph. The new factors and variables ($\{4, 5\}$) are added to the graph. We reeliminate the variables in this factor graph with a new constrained ordering forcing 4 and 99 to the end, producing the Bayes tree in Fig. 4.9(g). Note that the subtrees representing $p(X_l, X_n|X_r)$ from timestep t are carried over and attached to the newly-created cliques at $t + 1$.

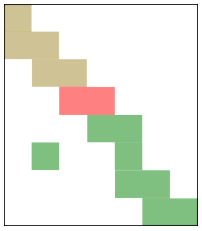
Subtree Caching

For planning, a large number of candidates are generally considered at each timestep. We presented the variable ordering constraints above in the context of efficient evaluations for a *single* candidate across *multiple* timesteps. However, we can extend the previous ideas to leverage the structure of the Bayes tree across *multiple* candidates at a *single* timestep. The key insight in this approach is that if we force the entrance poses of multiple candidates, say `A` and `B`, to be collocated and use the variable ordering constraints proposed above, then the

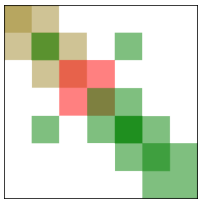
Figure 4.9 Illustrative example of the proposed constrained variable ordering. At timestep t, the robot plans a path beginning from current pose 3 and through entrance pose 99. The path contains a predicted loop-closure measurement between node 2 and virtual node 101. (a) The factor graph at time t. The yellow nodes represent variables in the SLAM graph and the green nodes represent virtual poses along the candidate path. The measurement Jacobian A, information matrix Λ, and square-root information are shown in (b), (c), and (d), respectively. The corresponding Bayes tree for this factor graph is given in (f), with variables 3 and 99 constrained to the root. At the following timestep $t+1$, the robot evaluates the candidate path once again. By design, only the virtual factors between the previous SLAM node 3 and the entrance pose 99 are modified. (e) The factor graph at $t+1$, which now includes the new current SLAM node 4 and a new virtual pose 5. The Bayes tree at $t+1$ given in (g) is minimally updated during this evaluation and reuses the subtrees calculated at time t. In this way the update is efficient due to the partitioning of the graph through the root.



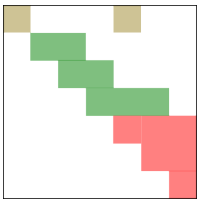
(a) Factor graph at timestep t



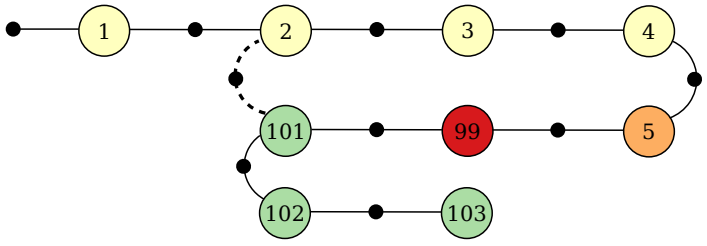
(b) A



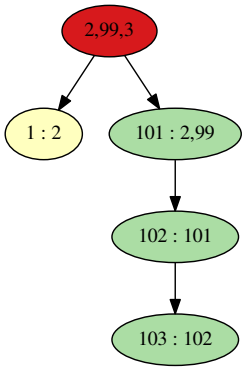
(c) Λ



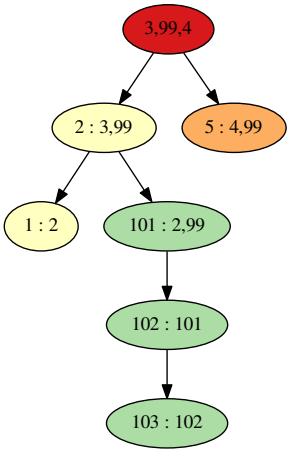
(d) Square-root information



(e) Factor graph at timestep $t+1$



(f) Bayes tree at t



(g) Bayes tree at $t+1$

factors encoded in the root clique of candidate A are the same as those encoded in the root clique of candidate B. In addition, the subtree corresponding to the virtual poses between the current SLAM node and the root is also the same:

$$\mathbf{x}_e^{\mathtt{A}} = \mathbf{x}_e^{\mathtt{B}} \rightarrow \begin{array}{l} p(X_r^{\mathtt{A}}) = p(X_r^{\mathtt{B}}) = p(X_r), \\ p(X_m^{\mathtt{A}}|X_r^{\mathtt{A}}) = p(X_m^{\mathtt{B}}|X_r^{\mathtt{B}}) = p(X_m|X_r). \end{array} \tag{4.12}$$

When B is initially evaluated, we cache the subtrees pertaining to the factors that do not change between evaluations, represented by $p(X_l, X_n^{\mathtt{B}}|X_r)$. Upon subsequent evaluations, we can recalculate the root of A and share this result with B. To evaluate B, we simply remove the subtree $p(X_l, X_n^{\mathtt{A}}|X_r)$ corresponding to candidate A and reattach the cached subtree $p(X_l, X_n^{\mathtt{B}}|X_r)$ of B to the shared root. This process is equivalent to directly assembling the square-root information matrix in the sparse linear algebra sense *without* any further calculations. All numerical entries in the matrix have already been calculated and only their positions within the matrix are left to assign. From here, the system is solved using backsubstitution beginning with the root.

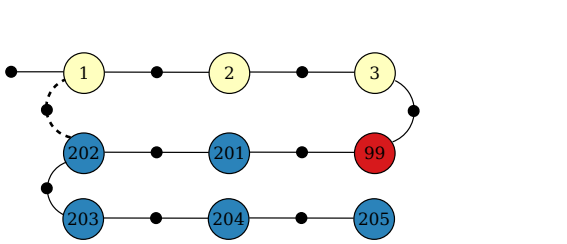
The example of Fig. 4.10 demonstrates the caching concept. We will call the previous example of Fig. 4.9 candidate A and the candidate in Fig. 4.10 candidate B. Once the Bayes tree for B has been constructed at timestep t, only the cached subtree below the root and the shared root from candidate A are necessary to construct the tree at $t + 1$.

Active SLAM Framework

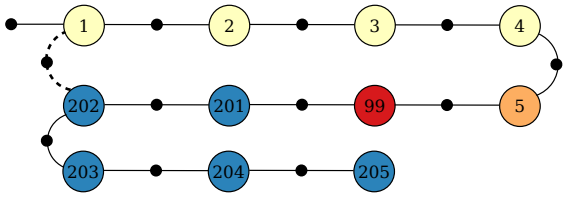
We present here an active SLAM framework that builds on the intermediate framework of §3.2 and follows the guidelines for efficient evaluation with the Bayes tree. Our application of interest is underwater inspection of a ship hull with a HAUV. The robot performs visual pose SLAM while following a nominal exploration policy that covers the target area efficiently but leads to unbounded uncertainty growth. Therefore, the goal of the active SLAM framework is to search and execute loop-closing actions throughout the mission such that the uncertainty remains below a desired threshold.

Algorithm 5 presents high-level pseudocode of our approach. At each timestep, we query the SLAM solver for the current robot pose (`GetSlamUpdate`()). If an entrance pose is not set, we select the entrance pose from a *lookahead horizon* of the next several steps of the nominal control policy (`ShiftEntrancePose`()). Given the entrance pose, we generate a bank of candidate trajectories that revisit sampled poses in the environment by traveling from the entrance pose, to the revisit pose, and back. This process is accomplished in the `PathGeneration`() function and detailed further in §3.2.1. For each candidate we compute

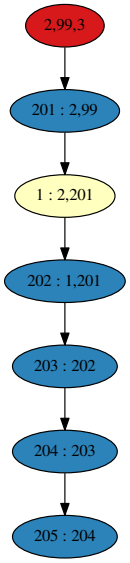
Figure 4.10 Illustrative example of the proposed subtree caching scheme. Given the candidate shown in Fig. 4.9, the robot plans a second path at timestep t. (a) The factor graph at t, where the blue variables represent the second candidate, and differ from those in Fig. 4.9(a). The corresponding Bayes tree at t is shown in (c) and contains the same root clique as the tree in Fig. 4.9(f) since the entrance pose 99 is also shared between candidates. (b) At time $t+1$, the factor graph for the second candidate is updated with current SLAM node 4 and new virtual pose 5, like that in Fig. 4.9(e). The corresponding Bayes tree in (d) is completely defined by the cached subtree from (c) and the root cliques from Fig. 4.9(g). Thus, no calculations are necessary to construct this tree, only assembling the parts.



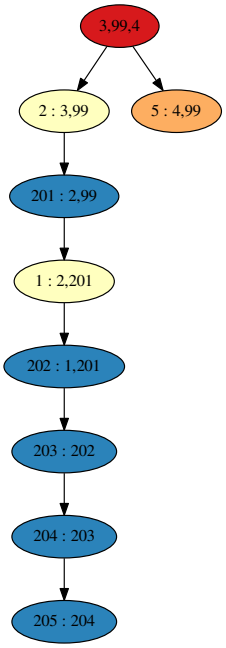
(a) Factor graph at timestep t



(b) Factor graph at timestep $t+1$



(c) Bayes tree at t



(d) Bayes tree at $t+1$

the predicted factors along the path $\{f_n^i\}$ and clear any previously-cached subtrees if they exist. The process is also executed at regular intervals or if the linearization point has changed significantly.

Also at each timestep, we retrieve any new factors $\{f_i\}$ incorporated into the SLAM system (`RetrieveNewFactors`()) and compute the factors $\{f_m\}$ that connect the current SLAM node to the entrance pose. Here, we can recalculate the shared root $\mathcal{R}$ of the Bayes tree inside the function `RecalculateRoot`() given the previous root (if it exists), the new factors $\{f_i\}$, and the current-node-to-entrance-pose factors $\{f_m\}$.

For each candidate in the candidate bank, we calculate the corresponding Bayes tree given the new shared root and the candidate-specific factors $\{f_n^i\}$. If cached subtrees exist, we can simply assemble the Bayes tree instead. Then, we can evaluate the candidate for its

Algorithm 5 Bayes Tree Active SLAM Framework

Initialize: bank of empty candidates $\{\mathcal{C}_i\}$.

```
while (SLAM is not finished) do
  x_c = GetSlamUpdate()
  if (time elapsed > Δt or LINPOINT changed) then
    x_e = ShiftEntrancePose()
    for (candidate C_i in {C_i}) do
      {f_n^i} = PathGeneration(C_i, x_e)
      ClearCachedSubtrees(C_i)
    end for
  end if
  {f_i} = RetrieveNewFactors()
  {f_m} = ComputeFactorsToEntrance(x_c, x_e)
  R = RecalculateRoot(R, {f_i}, {f_m})
  for (candidate C_i in {C_i}) do
    {S_i} = GetCachedSubtrees(C_i)
    if ({S_i} is empty) then
      T_i = CalculateBayesTree(R, {f_n^i})
      CacheSubtrees(T_i)
    else
      T_i = AssembleBayesTree(R, {S_i})
    end if
    EvaluateCandidate(T_i)
  end for
  UpdateBestCandidate()
  DecideAndExecute()
end while
```

utility in active SLAM by solving the system and recovering the final pose covariance, which we use as a measure of uncertainty. Given the set of candidate evaluations, the framework decides between continuing along the nominal trajectory or executing a candidate set of actions. Further information about the lookahead horizon and the selection criteria by which we determine the best candidate can be found in Chapter 3 [20].

4.6 Bayes Tree Results

An underwater robot is tasked with performing an inspection survey subject to a desired uncertainty threshold. The nominal control policy covers the target area efficiently but leads to unbounded uncertainty growth, so the robot relies upon the active SLAM framework to search and execute loop-closing actions. Considering this scenario, we present a brief investigation into the benefit of the proposed approach for active SLAM. The following experiments were run on a standard desktop computer with an Intel Xeon X5460 CPU. Our

algorithms were written in C++ using the Georgia Tech Smoothing and Mapping (GTSAM) library [28]. The experiments were performed within the hybrid simulation environment of Chapter 3 using data collected by the HAUV [80] and a synthetic image registration pipeline.

4.6.1 Timing and Complexity Comparison

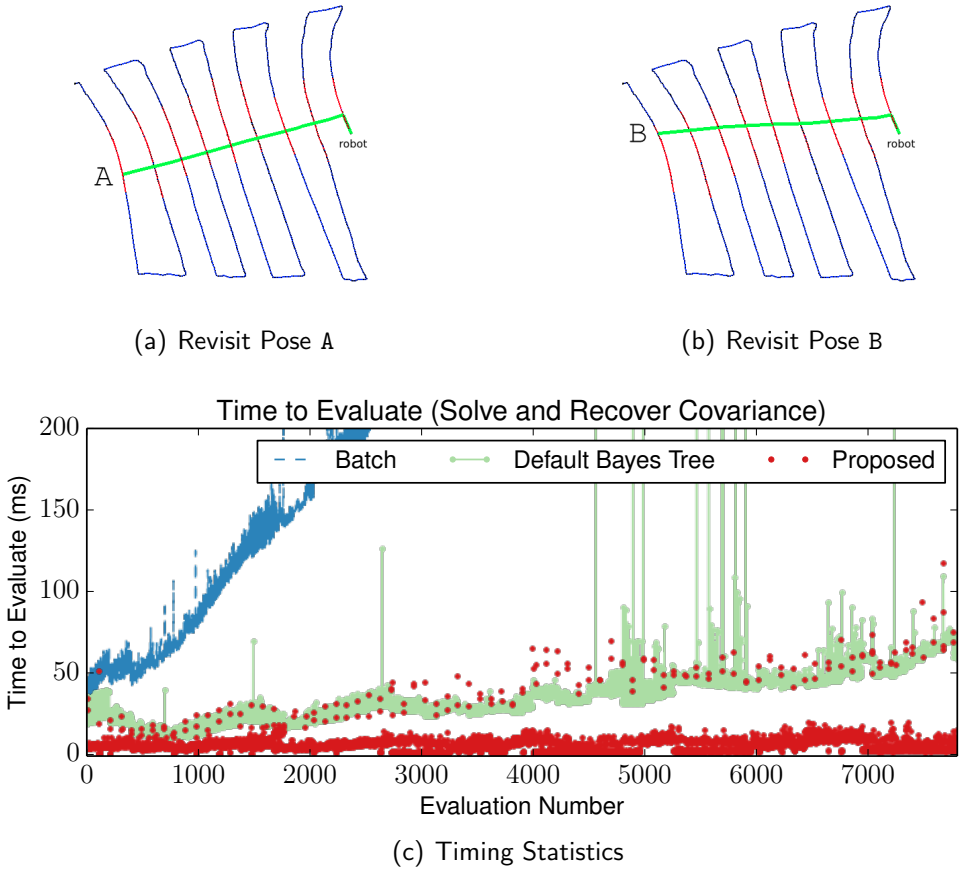
The first experiment presents a timing and complexity comparison of the proposed method in Fig. 4.11 and Fig. 4.12. At each timestep in the underwater inspection survey, the robot evaluates candidates corresponding to revisiting poses `A` and `B` in the environment. Snapshots of the trajectories at an example timestep are shown in Fig. 4.11(a) and (b). We calculate three statistics related to the evaluation of the candidates: (*i*) the time required to solve the system and recover the final pose marginal covariance, (*ii*) the number of variables reeliminated, and (*iii*) the number of factors recalculated. The proposed method with ordering constraints and subtree caching is compared against two other methods for evaluation. First, we compare against a batch solving operation where all variables and factors are recomputed during each evaluation. Second, we compare against using the Bayes tree with the default variable ordering provided by iSAM2 [74] and without reusing the virtual factors between evaluations, as if the candidates computed the control actions to `A` and `B` independently at each timestep without the entrance pose constraint.

Seen in Fig. 4.11(c), the proposed method significantly reduces computational time over the other approaches. On average, the proposed approach offers an order of magnitude reduction (9.23 x) in computation time over the default Bayes tree method and two orders of magnitude reduction (105.9 x) over the batch operation. Note that the periodic shifting of the entrance pose in the proposed algorithm is reflected in the timing plot. The proposed method aligns with the default Bayes tree method in this case. Fig. 4.12(a) and (c) show the variables reeliminated from the system at each timestep and cumulatively over the entire mission, respectively. Similarly, Fig. 4.12(b) and (d) display the same plots for the number of factors recalculated.

4.6.2 Time-constrained Active SLAM

Here we show results on the performance of the full proposed framework in an active SLAM setting. For this experiment we allowed the proposed method to search, decide, and execute loop-closing revisit actions within the hybrid simulation environment. We show results of the active SLAM performance considering three methods for evaluating candidates: (*i*) the proposed constraint ordering and subtree caching approach, (*ii*) the batch solving operation, and (*iii*) the batch solving operation with a time limitation of 200 ms. In the latter scenario,

Figure 4.11 Timing statistics for evaluating candidates traveling to revisit poses `A` and `B` over the course of the mission. The proposed method far outperforms evaluation via a batch operation and exhibits significant time savings over the default Bayes tree evaluation method.



(a) Revisit Pose `A`

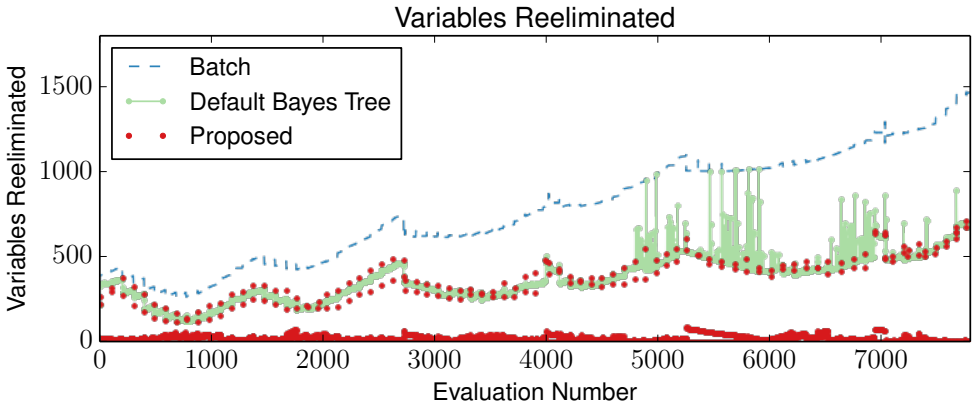
(b) Revisit Pose `B`

(c) Timing Statistics

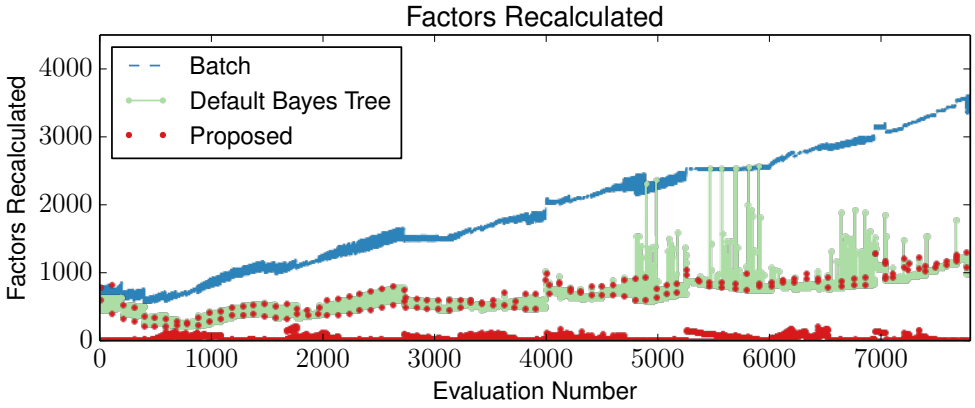
we restricted the framework to discard candidates if their evaluation required more than 200 ms.

Fig. 4.14(a) provides the evaluation times for every candidate searched throughout the mission. Once again, the computational reduction of the proposed approach is clear compared to the batch solution. We also see in this figure that the batch method (in both cases) violates the time restriction fairly early in the mission. As the robot explores and the SLAM system grows, the batch operation is unable to evaluate candidate actions in a timely manner. We can see the effect of this computational bottleneck on the resulting active SLAM performance in Fig. 4.14(b). This figure shows the performance of the framework with respect to constraining uncertainty while maintaining efficient area coverage. Here, the time-constrained batch method fails to find loop-closing actions that adequately reduce the uncertainty throughout the mission. Both the proposed and unrestricted batch approaches result in nearly identical acceptable uncertainty levels. However, the proposed method greatly

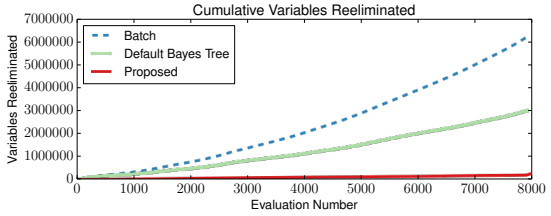
Figure 4.12 Complexity statistics for evaluating candidates traveling to revisit poses `A` and `B` over the course of the mission. The proposed method far outperforms evaluation via a batch operation and exhibits significant complexity savings over the default Bayes tree evaluation method.



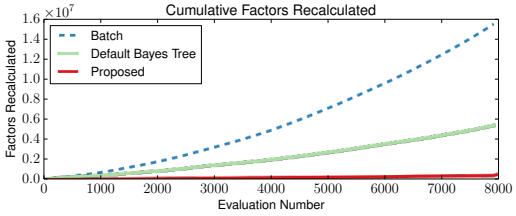
(a) Variables Reeliminated



(b) Factors Recalculated

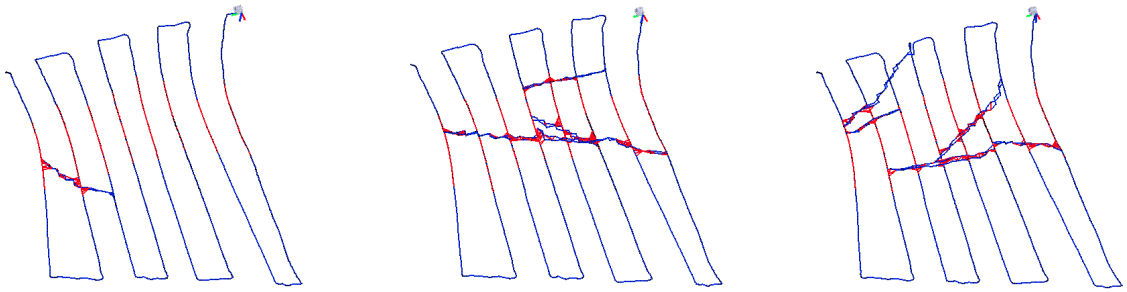


(c) Cumulative Sum



(d) Cumulative Sum

Figure 4.13 Performance results of the proposed method in an active SLAM simulation. The active SLAM framework is tested in three trials (resulting trajectories shown): (a) a time-constrained batch evaluation approach, (b) an unrestricted batch approach, and (c) the full proposed method with constrained variable ordering and subtree caching.



(a) Batch-TC (b) Batch (c) Proposed

outperforms in terms of computational efficiency.

4.7 Bayes Tree Summary

We presented contributions within active SLAM by exploiting the Bayes tree data structure for efficient planning. In particular, our proposed ideas included a constrained variable ordering and subtree caching scheme that reduce complexity in the planning problem by reusing computations between candidates. We also proposed an active SLAM framework using these concepts and showed the benefits of the method with respect to timing, complexity, and performance within active SLAM.

4.8 Discussion

Both of the concepts introduced in this chapter can be thought of as methods to precompute operations involved with solving the simulated SLAM system associated with planning. Kaess et al. [74] identified that the SLAM solution proceeds from a variable elimination procedure on the corresponding factor graph. To evaluate a candidate plan with an information-theoretic objective, all variables in the graph must be eliminated.

The online graph sparsification algorithm eliminates variables incrementally during the mission, folding the information content associated with the elimination into new factors across the connected nodes (accomplished with the GLC approximate marginalization procedure). In this way, the elimination cost for the full pose graph is amortized over the duration of the mission and eliminations performed once are reused for every planning query. There are

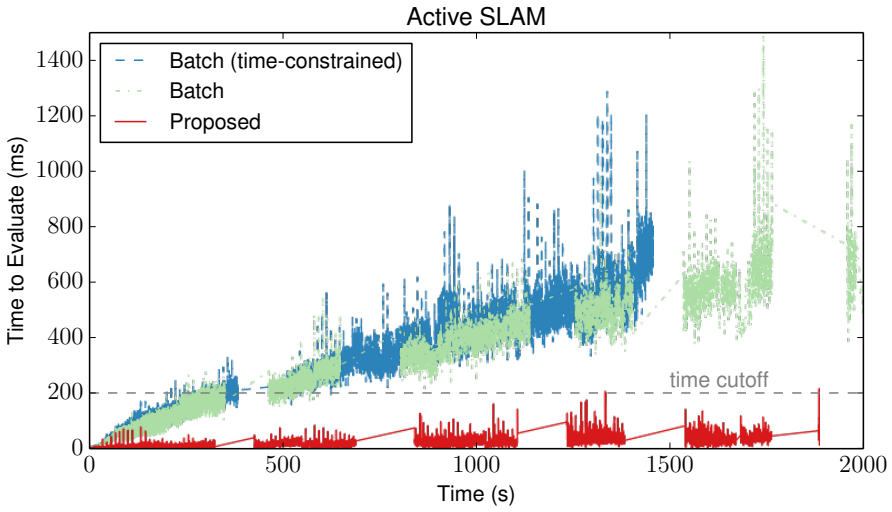
downsides to this approach, however. The sparse GLC method commits to linear factors that only approximate the information content, such that the original nonlinear factors are no longer expressed. In addition, the incremental approach may prematurely remove nodes involved in future loop-closure measurements. Our proposed online repair process partially alleviates some of these issues but a more principled approach would be beneficial.

The Bayes tree planning framework performs variable elimination in such a way that the information within the pose graph is factored and expressed as a product of conditional distributions. Nodes are not removed from the graph and the original nonlinear measurements are not replaced with approximate, linear constraints. In other words, the information is not folded into a compressed representation but rather structured into an efficient form. We proposed a method for leveraging the structure in this chapter, but our method places restrictions on the design of candidate actions and requires a knowledge as to how the factored distribution can be reused in order to gain computational savings.

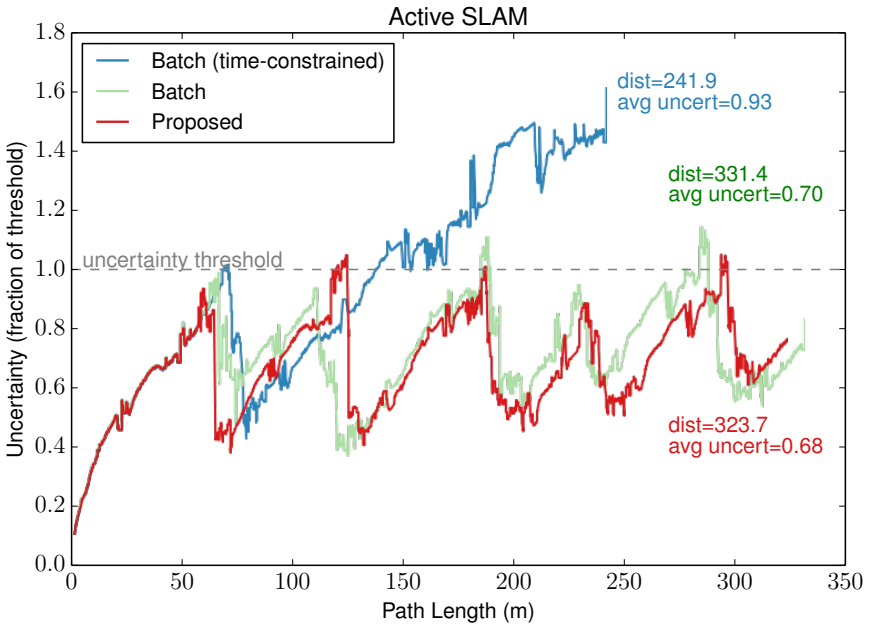
4.9 Chapter Summary

This chapter presented ideas related to improving the efficiency of active SLAM evaluations through compressing the representation and leveraging the structure of the planning problem. We investigated the concept of compression by proposing an online graph sparsification algorithm that reduces the dimensionality of the pose SLAM system in order to reduce complexity. We also examined the Bayes tree as an efficient data structure for planning, and proposed an active SLAM framework centered around its use in removing redundant computations between multiple candidate plan evaluations. Our methods show significant computational savings and provide a glimpse into how state-of-the-art techniques in the SLAM community can be beneficial when adapted to planning applications.

Figure 4.14 Performance results of the proposed method in an active SLAM simulation. Timing statistics are presented in (a) and a plot of uncertainty vs. path length for each scenario in (b). When the batch method is restricted to run in the same time as the proposed approach, it fails at finding loop-closing actions to constrain the uncertainty. The proposed method is able to sufficiently constrain the uncertainty in a coverage-efficient manner with greatly enhanced computational efficiency.



(a) Timing Statistics



(b) Uncertainty vs. Path Length

Chapter 5

Risk-averse Optimization under Uncertainty

5.1 Introduction

The underwater environment poses unique challenges for visual simultaneous localization and mapping (SLAM). Visual features are sparsely distributed throughout the environment, making it difficult to gather loop-closure measurements. Even when features are present, there is some probability that two overlapping images will not be registered. In Chapter 2, we developed a planning formulation that captures this variability in acquisition, useful for planning with underwater visual SLAM and other robotics problems. The variability in acquisition directly leads to variability in localization; the robot's belief (represented by a mean vector and information matrix) is highly dependent on the measurements received. Therefore, when predicting how the belief will evolve given a candidate sequence of control actions, there is a range of outcomes. The algorithms we proposed in Chapter 3 and Chapter 4 do not consider this belief *distribution* when selecting the best actions. Two paths with the same *expected* benefit will evaluate equivalently in the optimization, even if one has a wider range of outcomes than the other. This is because the objective functions used to this point (with respect to robot uncertainty) are *risk-neutral*. It would be beneficial to consider the risk associated with the randomness in the belief in the optimization.

This chapter proposes risk-averse objective functions for use in general belief-space planning problems, with a specific focus on minimizing localization uncertainty in active SLAM. We show that the planning formulation of Chapter 2 allows us to design these functions to account for the stochasticity in the belief. In addition, we show how risk-averse planning leads to desirable behavior within active SLAM. Our inspiration for risk-averse optimization

is drawn from research in modern portfolio theory, which we adopt and apply to the robotics domain. In order to design risk-averse objectives, we also propose an analytic method for calculating the first two moments of the posterior belief distribution. The analytic method is much more computationally efficient than approaches based on exhaustive enumeration or sampling of outcomes, and serves as a good strategy for approximating the distribution of beliefs in the planning problem.

5.1.1 Related Work

Objective function design for active SLAM was previously investigated by Carrillo et al.[15, 16], who examined the Theory of Optimal Experiment Design to construct appropriate criteria for measuring uncertainty in the system. Despite their conclusion that the determinant has desirable properties for use in active SLAM or navigation under uncertainty [17], several works use the trace of the covariance as an uncertainty metric [67, 114, 126].

Belief-space planning problems like active SLAM or navigation under uncertainty frame their decision-making process as a multiobjective optimization with costs on localization uncertainty, control effort or distance traveled, and (at times) reaching a goal pose [67, 105, 106, 126]. With the inclusion of the random measurements into the planning prediction [126], researchers examined how this randomness affects the components within the objective function concerning the state vector, X_k. In point-to-point queries, the random measurement values directly affect the probability of reaching the goal. Both van den Berg et al. [126] and Indelman et al. [67] include a quadratic cost on reaching the goal pose that encodes a certain amount of aversion to risk, as we show later. Therefore, the inclusion of this cost component leads to a conservative solution at the expense of a performance metric, like shortest distance. This conservatism was also demonstrated by van den Berg et al. in regards to gathering measurements in a light-dark world.

For those works solving planning queries in environments with obstacles, risk has also been considered under the topic of collision avoidance [34, 126, 131]. In deterministic planning, checking collisions with obstacles is a function of the robot state. Extending to the stochastic setting, where we track a robot's belief as a distribution over state, constraints in the optimization concerning collision must be formulated as *chance constraints* [18], where the failure is now described by a probability, and aversion to risk in this context is modeled by explicitly satisfying these constraints [33].

To evaluate chance constraints, there have been two typical approaches: (i) use a parameterized form of the underlying belief (e.g., Gaussian) and transform the chance constraints to constraints on the parameters [125], and (ii) use Monte Carlo simulations to

transform the stochastic problem into sampled deterministic evaluations and approximate the expected cost with the sampled set [10]. This particle-based approach does well in representing non-Gaussian distributions but the compounded computational cost of multiple objective evaluations may not be tractable for large sample sizes.

Most engineering applications formulate a dual criteria optimization to account for risk in stochastic outcomes. As in the chance-constrained case, the optimization seeks to maximize expected return subject to some limit on a *risk measure*. Markowitz [94] originally proposed variance as a measure of risk for making decisions on investments within a portfolio. Another common measure is value-at-risk (VaR), which defines a minimum level for a portfolio such that the probability of portfolio loss greater than this level does not exceed some confidence threshold [90]. A related measure known to be more consistent is the conditional value-at-risk (CVaR) [111], defined as the conditional expectation of the losses above the VaR. Ben-Tal and Teboulle [9] introduced the optimized certainty equivalent (OCE) as a risk measure, showing its parallels to expected utility theory.

Another approach to portfolio optimization, which we examine in this chapter, is *expected utility theory*, where a von Neumann-Morgenstern utility function that defines rational investor behavior is optimized [96, 132]. Different from the dual criteria optimization, this single criterion approach (the "economist" approach [112]) implicitly encodes aversion to risk in the utility function of wealth. Seck et al. [112] demonstrated an equivalence between the dual constraint optimization problem and a max-min formulation involving a certain class of utility functions. Quadratic, exponential, and power utility functions are all popular forms in the financial world [46]. The quadratic function is related to the traditional mean-variance analysis of Markowitz [71]. Exponential utility and related approaches have been widely applied to Markov decision processes (MDPs) [61, 93, 97].

5.2 Representing the Posterior Belief Distribution

The planning formulation of Chapter 2 computes a "distribution of distributions" relating to the posterior belief over a sequence of control actions. That is, the belief computed over the planning horizon considering the random measurements and random acquisitions is itself *random*. In previous chapters, we glossed over how to represent this posterior belief distribution, as the optimization problems in these chapters only required a calculation of the mean of the distribution. In this chapter, however, we will examine risk-sensitive optimization (specifically risk-averse optimization) that demands a more complete representation of the posterior belief distribution.

Recall that the posterior belief $\mathcal{B}_k \sim \mathcal{N}\left({X_k}^*, {\Lambda_k}^{-1}\right)$ is calculated over the planning

horizon in a nonlinear least-squares framework, given random acquisition variables and random measurements along a candidate sequence of control actions. Solving, the belief mean vector and information matrix are (restated here for convenience):

$$\begin{aligned} {X_k}^*(Z_{1:k}, \Gamma_{1:k}, U_{0:k-1}) &= \bar{X}_k + \left(A_k^\top \mathcal{G}_k A_k\right)^{-1} A_k^\top \mathcal{G}_k \mathbf{b}_k, \\ \Lambda_k(\Gamma_{1:k}, U_{0:k-1}) &= A_k^\top \mathcal{G}_k A_k, \end{aligned} \tag{5.1}$$

where $\mathcal{G}_k$ encompasses the random acquisition variables and the residual $\mathbf{b}_k$ contains the random measurements. This leads to the following question: how can we express the posterior belief distribution?

To represent the complete distribution, we can consider three options [55]. One option is to exhaustively enumerate every possible combination of measurements and corresponding acquisitions over the sequence of actions, as in Fig. 5.1(a). It is clear that this is an intractable endeavor, however; for each action, there are multiple measurements that might be possible, each with an associated acquisition variable, and a continuum of values that may be realized. A second option is to randomly sample a number of measurement and acquisition sequences over the planning horizon to capture the distribution as a collection of sampled outcomes (Fig. 5.1(b)) [10]. Still, this involves evaluating the deterministic belief for each sampled sequence, which is expensive for information-theoretic objectives.

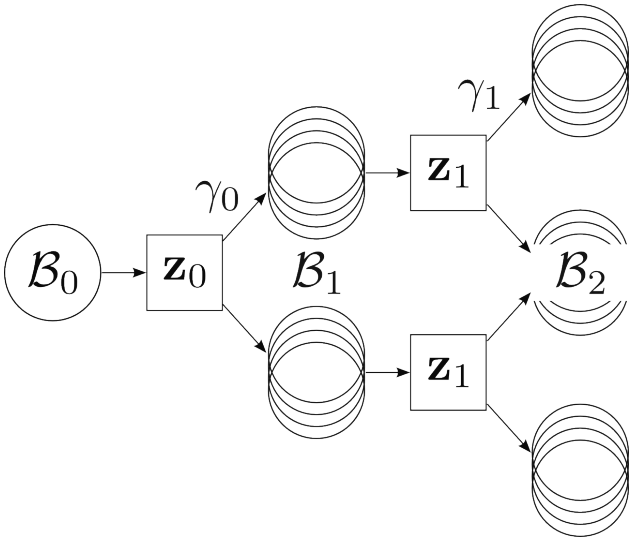
In the spirit of He et al. [55], we instead take an analytic approach, shown in Fig. 5.1(c). We choose to *approximate* the posterior belief distribution as Gaussian and analytically compute the first and second moments. In this way, we can employ the Gaussian parameterization to efficiently compute the expected reward during the risk-sensitive optimization, avoiding multiple evaluations related to exhaustive enumeration or sampling. Therefore, we approximate the first two moments of the posterior belief distribution by projecting the uncertainty from the random variables into the belief space. This projection is accomplished with the helpful equation for first-order propagation of uncertainty,

$$\mathrm{Var}[y(X)] \approx \frac{\partial y}{\partial X}\Big|_{\mathbb{E}[X]} \cdot \mathrm{Var}[X] \cdot \frac{\partial y}{\partial X}\Big|_{\mathbb{E}[X]}^{\top}. \tag{5.2}$$

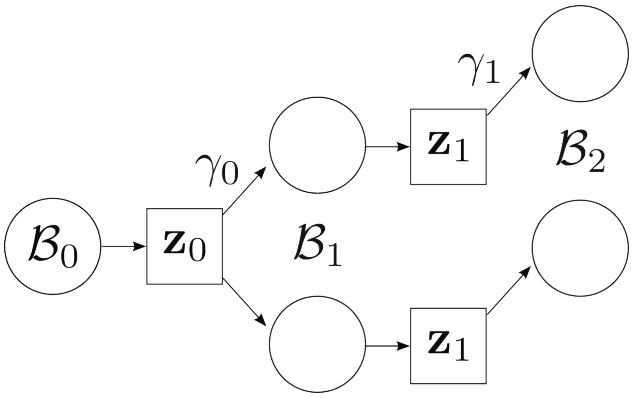
Then, the expected value and variance of the elements within the belief-space state vector are:

$$\begin{aligned} \mathbb{E}\left[{X_k}^*\right] &= \bar{X}_k, \\ \mathrm{Var}\left[{X_k}^*\right] &\approx R_k \cdot \mathrm{Var}[\mathbf{b}_k] \cdot {R_k}^\top, \\ \mathbb{E}\left[\mathrm{vec}(\Lambda_k)\right] &= \mathrm{vec}\left(A_k^\top \mathbb{E}[\mathcal{G}_k] A_k\right), \\ \mathrm{Var}\left[\mathrm{vec}(\Lambda_k)\right] &\approx P_k \cdot \mathrm{Var}[\Gamma_{1:k}] \cdot {P_k}^\top, \end{aligned} \tag{5.3}$$

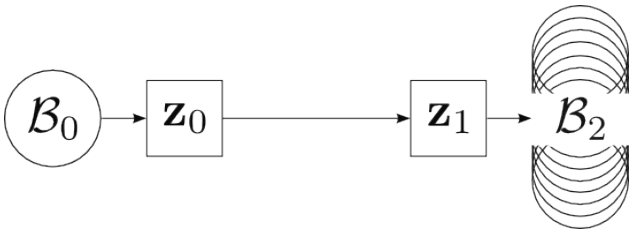
Figure 5.1 Methods for representing the posterior belief distribution. (a) Exhaustive enumeration of all possible measurement and acquisition sequences is intractable. (b) Sampling measurement and acquisition sequences to form a collection of sampled outcomes that capture the distribution. This is still expensive for information-theoretic objectives. (c) We take an analytic approach and approximate the posterior belief distribution by projecting the uncertainty from the random variables into the belief space.



(a) Exhaustive enumeration



(b) Sampling sequences



(c) Analytic computation

where

$$R_k = \left(A_k^\top \mathbb{E}[\mathcal{G}_k] A_k\right)^{-1} A_k^\top \mathbb{E}[\mathcal{G}_k], \tag{5.4}$$

and the columns of the partial derivative P_k (indexed by c) correspond to an individual $\gamma_{i,j}$,

$$P_k^{(c)} = \text{vec}\left(H_k^{i,j\top} \Omega_v^{i,j} H_k^{i,j}\right). \tag{5.5}$$

$\text{Var}[\mathbf{b}_k]$ is the residual covariance and $\text{Var}[\Gamma_{1:k}] = [\mathcal{D}(\lambda_{i,j}(1-\lambda_{i,j}))]$ is the diagonal covariance of the independent acquisition variables. The covariance of the mean vector, $\text{Var}[X_k{}^*]$, is a function of the randomness in both measurements and acquisitions. To make this calculation possible, however, we only consider the variance of the measurements (through the residual vector $\mathbf{b}_k$) and evaluate the acquisition variables at the mean [67]. The covariance of the information, $\text{Var}[\text{vec}(\Lambda_k)]$, depends on the random acquisitions and not the measurements themselves. It is worth noting that $H_k^{i,j}$ is sparse for many robotics applications, allowing us to efficiently compute the covariance of the information in (5.3) by leveraging sparsity patterns. Now that we have an analytical representation of the posterior belief distribution, we will discuss designing a risk-averse optimization next.

5.3 Risk Aversion

Given a representation of the random belief from the planning prediction, we seek to design objective functions that consider the stochasticity. Engineers often optimize with stochastic outcomes by posing a dual criteria problem: maximize expected return subject to explicit risk constraints [112]. The risk constraints are generally framed as thresholds on *risk measures*, like variance [94], VaR, CVaR [111], and OCE [9]. Economists, on the other hand, often formulate risk-sensitive optimization as a single criterion problem, maximizing the expected value of a von Neumann-Morgenstern utility function that implicitly encodes aversion to risk [132]. Inspired by portfolio selection via expected utility theory, we adopt the economist approach in this chapter.

Research from expected utility theory shows that investor behavior can be encoded in a utility function of wealth, $\mathscr{U}(W)$, such that maximizing the expected return on utility, $\mathbb{E}[\mathscr{U}(W)]$, results in more desirable decisions than directly maximizing the expected return on wealth, $\mathbb{E}[W]$. A utility function that encodes rational investor behavior holds to four main axioms [46]:

1. Investors exhibit non-satiation, $\mathscr{U}'(W) > 0$.

2. Investors exhibit risk aversion, $\mathscr{U}''(W) < 0$.

3. Investors exhibit decreasing absolute risk aversion, $\mathscr{A}'(W) < 0$.

4. Investors exhibit constant relative risk aversion, $\mathscr{R}'(W) = 0$.

The first two axioms equate to $\mathscr{U}(W)$ being monotonic and concave with respect to W. The measures of absolute risk aversion and relative risk aversion are defined as

$$\begin{aligned} \mathscr{A}(W) &= -\frac{\mathscr{U}''(W)}{\mathscr{U}'(W)}, \\ \mathscr{R}(W) &= W \cdot \mathscr{A}(W). \end{aligned} \tag{5.6}$$

Absolute risk aversion is related to the absolute amount of wealth an investor puts toward risky assets. Similarly, relative risk aversion is related to the fraction of wealth invested in risky assets.

We can consider the robot as an investor with the goal of maximizing its wealth from a number of risky investments. However, rather than maximizing wealth, the robot seeks to maximize information from a number of uncertain sensor measurements. Specifically in this chapter, we are interested in aversion to risk with respect to robot localization—that is, with respect to the error in the robot's knowledge of its state. Hence, we focus our investigation of risk aversion on the information-theoretic components of the objective function (those described by $g_\Lambda({\Lambda_k}^{-1})$).

Within the objective function, we seek to minimize uncertainty rather than maximize wealth, so we define

$$W = \mathcal{T} - tr({\Lambda_k}^{-1}) = \mathcal{T} - \mathbf{m}^\top \mathrm{vec}({\Lambda_k}^{-1}), \tag{5.7}$$

where $\mathcal{T}$ is a user-specified upper bound on the uncertainty and the trace is expanded using an element selection vector $\mathbf{m}$. The mean and variance of the "wealth" is represented by

$$\begin{aligned} \mathop{\mathbb{E}}_{\Gamma_{1:k}}[W] &= \mathcal{T} - \mathbf{m}^\top \mathbb{E}[\mathrm{vec}({\Lambda_k}^{-1})] \\ &\approx \mathcal{T} - \mathbf{m}^\top \mathrm{vec}\left(\mathbb{E}[\Lambda_k]^{-1}\right), \end{aligned} \tag{5.8}$$

and

$$\mathop{\mathrm{Var}}_{\Gamma_{1:k}}[W] = \mathbf{m}^\top \mathrm{Var}\left[\mathrm{vec}({\Lambda_k}^{-1})\right] \mathbf{m}. \tag{5.9}$$

The variance of the belief covariance matrix is written in terms of the variance of the belief information matrix:

$$\mathop{\mathrm{Var}}_{\Gamma_{1:k}}\left[\mathrm{vec}({\Lambda_k}^{-1})\right] \approx L_k \cdot \mathop{\mathrm{Var}}_{\Gamma_{1:k}}\left[\mathrm{vec}(\Lambda_k)\right] \cdot {L_k}^\top, \tag{5.10}$$

where we once again used the first-order projection of uncertainty with

$$L_k = -\left(\mathbb{E}[\Lambda_k]^{-\top} \otimes \mathbb{E}[\Lambda_k]^{-1}\right), \tag{5.11}$$

and $\otimes$ denotes the Kronecker product. Like the calculations in (5.3) and (5.5), we can directly leverage the sparsity in computing the trace from $\mathbf{m}$ in order to find only those terms in L_k that will be used in (5.9).

Also, the robotics planning problems we consider are typically formulated as minimizations. To transform the maximization of expected utility into a minimization, we write an equivalent penalty function from the utility function:

$$\mathscr{P}(W) = -\mathscr{U}(W). \tag{5.12}$$

At this point, an open question is how to design the form of the utility function $\mathscr{U}(W)$ for planning problems, specifically active SLAM. To do this in a way that facilitates multiobjective optimization, we turn to expected utility theory.

5.3.1 Expected Utility Functions

In the robotics planning literature, the localization uncertainty costs within the planning objective are typically linear in the trace (or determinant) of the belief covariance [20, 67, 114, 126]. The linear objective function is monotonic but not concave, and is therefore *risk-neutral*. Without modeling randomness in the acquisition, this cost is sensible because the belief covariance is deterministic [55]. However, our formulation leads to a random belief covariance and minimizing a linear cost in this case disregards variability in the outcome. Instead, we prefer to design objective functions that are risk-averse by replacing the linear cost with an appropriate utility function. Quadratic, exponential, and power utility functions are very common in the financial world, so we examine their usefulness in our optimization.

Quadratic Utility

The quadratic utility function is written as

$$\mathscr{U}_q(W) = W - \frac{\eta_q}{2} W^2, \tag{5.13}$$

with $\eta_q > 0$ and $W \leq 1/\eta_q$. Trivially, the equivalent penalty function is

$$\mathscr{P}_q(W) = -W + \frac{\eta_q}{2} W^2. \tag{5.14}$$

Considering the wealth W as a random variable and taking the expectation, the expected penalty becomes

$$\mathop{\mathbb{E}}_{\Gamma_{1:k}}[\mathscr{P}_q(W)] = -\mathbb{E}[W] + \frac{\eta_q}{2}\left(\mathrm{Var}[W] + (\mathbb{E}[W])^2\right). \tag{5.15}$$

Notice that the quadratic function is completely described by the first two moments, no matter the shape of the distribution. This is an appealing property in many optimization contexts when the full distribution is unknown. In addition, the quadratic utility function has parallels to traditional mean-variance optimization introduced by Markowitz [94]. However, despite being risk-averse, the quadratic function does not satisfy the third and fourth axioms for rational behavior outlined above [46]. Specifically, this function exhibits *increasing* absolute and *increasing* relative risk aversion. Inserting our definition of wealth and removing constant terms, we arrive at the following form of the quadratic penalty function:

$$\begin{aligned}\mathop{\mathbb{E}}_{\Gamma_{1:k}}[\mathscr{P}_q(W)] = (1 - \mathcal{T}\eta_q)\mathbf{m}^\top \mathrm{vec}\left(\mathbb{E}[\Lambda_k]^{-1}\right) \\ + \frac{\eta_q}{2}\left(\mathbf{m}^\top \mathrm{Var}\left[\mathrm{vec}({\Lambda_k}^{-1})\right]\mathbf{m} + \left(\mathbf{m}^\top \mathrm{vec}\left(\mathbb{E}[\Lambda_k]^{-1}\right)\right)^2\right).\end{aligned} \tag{5.16}$$

Many optimization frameworks, though, design quadratic costs in the form of $\mathbb{E}[X^\top Q X]$, where Q is a positive-definite weight matrix. This form can be factored as $\mathbb{E}[X^\top Q X] = \mathbb{E}[X]^\top Q \mathbb{E}[X] + tr(Q\mathrm{Var}[X])$. By assigning $\mathcal{T} = 1/\eta_q$, we can transform the quadratic penalty to fit this expression,

$$\mathop{\mathbb{E}}_{\Gamma_{1:k}}[\mathscr{P}_q(W)] = \frac{\eta_q}{2}\left(\mathbf{m}^\top \mathrm{Var}\left[\mathrm{vec}({\Lambda_k}^{-1})\right]\mathbf{m} + \left(\mathbf{m}^\top \mathrm{vec}\left(\mathbb{E}[\Lambda_k]^{-1}\right)\right)^2\right). \tag{5.17}$$

However, this form of the utility function fixes the level of risk aversion, such that adjusting the value of the risk parameter η_q adjusts the mean and variance terms in tandem. Therefore, we prefer to employ the more general function of (5.16) in our analysis.

Exponential Utility

Exponential utility has been widely examined for risk-averse optimization, including in MDPs [61, 93, 97]. The exponential utility function is given by

$$\mathscr{U}_e(W) = -e^{-\eta_e W}, \tag{5.18}$$

with $\eta_e > 0$. Like quadratic utility, the exponential function does not hold to the third and fourth axioms for rational behavior [46, 97]. Exponential utility exhibits constant absolute

risk aversion ($\mathscr{A}(W) = \eta_e$) and increasing relative risk aversion. When the wealth is assumed to be normally distributed, minimizing the exponential penalty function reduces to

$$\mathop{\mathbb{E}}_{\Gamma_{1:k}}\left[\mathscr{P}_e(W)\right] = \exp\left(\eta_e\left(\mathbf{m}^\top \mathrm{vec}\left(\mathbb{E}[\Lambda_k]^{-1}\right) + \frac{\eta_e}{2}\mathbf{m}^\top \mathrm{Var}\left[\mathrm{vec}({\Lambda_k}^{-1})\right]\mathbf{m}\right)\right). \tag{5.19}$$

Thus, exponential utility provides a very intuitive way of handling different levels of absolute risk aversion within the optimization—varying the value of η_e squares the weight of the variance term while linearly adjusting the effect of the mean term. In addition, we see that the exponential penalty is no longer a function of the uncertainty threshold $\mathcal{T}$, which is attractive from a design standpoint.

Power Utility

A utility function that follows rational investor behavior defined by the four axioms is the power function, given by

$$\mathscr{U}_p(W) = \begin{cases} \frac{W^{(1-\eta_p)}}{(1-\eta_p)}, & \eta_p \neq 1 \\ \log W, & \eta_p = 1 \end{cases}. \tag{5.20}$$

Here, the relative risk aversion equals the parameter η_p ($\mathscr{R}(W) = \eta_p$). We can write an equivalent penalty function to the power utility function:

$$\mathscr{P}_p(W) = \begin{cases} -\frac{W^{(1-\eta_p)}}{(1-\eta_p)}, & \eta_p \neq 1 \\ -\log W, & \eta_p = 1 \end{cases}. \tag{5.21}$$

For a random belief covariance and $\eta_p \neq 1$, the expected value of the power penalty function is approximated using a Taylor series expansion as

$$\mathop{\mathbb{E}}_{\Gamma_{1:k}}\left[\mathscr{P}_p(W)\right] \approx -\frac{\mathbb{E}[W]^{(1-\eta_p)}}{1-\eta_p} + \frac{\eta_p}{2}\mathrm{Var}\left[W\right]\mathbb{E}[W]^{(-\eta_p-1)}. \tag{5.22}$$

While this function does not reduce as nicely as the quadratic and exponential penalties, it is the only utility of the three that follows rational behavior in all scenarios.

We propose the use of the above three risk-averse penalty functions within the planning objective function. Replacing the commonly-used linear uncertainty costs with these penalty functions naturally encodes risk-averse decision-making with respect to the belief uncertainty. Each of these three functions has different characteristics that may be useful or desired for different applications.

5.4 Simulation Results

We now present simulation results that show the effect of the random acquisition variables on the belief and the benefit of risk-averse planning.

5.4.1 Analytic vs. Sampling Comparison

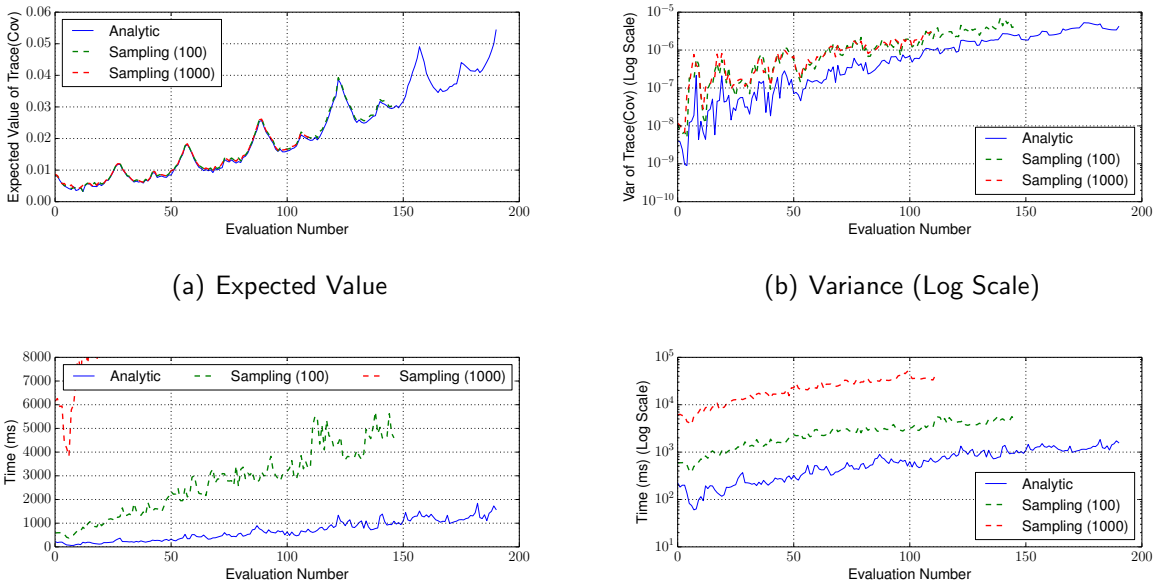
Here we present an initial comparison of the analytic and sampling strategies for computing the posterior belief distribution presented in §5.2. We perform an experiment using the hybrid simulation detailed in §3.4, with a moderately-salient region down the middle of the environment. As the robot explores following the nominal exploration policy, we continuously plan revisit paths through the salient area and predict the belief along each path using the formulation of Chapter 2.

Fig. 5.2 shows results from computing the posterior belief distribution for each evaluation during the experiment. Specifically, we compare the analytic and sampling approaches to finding the mean and variance of the trace of the final pose covariance, which are the terms of interest for our active SLAM system. Fig. 5.2(a) shows that the analytically-calculated mean of the trace exactly follows the sample mean with both 100 and 1000 samples. The variance of the trace is shown in Fig. 5.2(b). While the analytic result is not equivalent to the sample variance, the approximate calculation does follow the same trend (plotted on a log scale). For planning purposes, the variance computation does not need to be exact; we are interested in expressing the risk associated with different decisions and it is sufficient to preserve the relative risk measures between options. The benefit of the analytic approach is seen in Fig. 5.2(c) and (d). The sampling strategy is roughly an order of magnitude more computationally expensive for 100 samples and two orders of magnitude for 1000 samples. Thus, we are willing to approximate the mean and variance of the trace in order to save significant time and computational resources.

5.4.2 One-dimensional Intuition

The following example shown in Fig. 5.3 and Fig. 5.4 illustrates the intuition behind the method for risk-averse planning. Consider a sensor placed in a one-dimensional environment with prior belief information $\Lambda_0 = 1.0$. The sensor is able to receive measurements from two sources with constant information. Measurement source S_a has information Ω_a and measurement source S_b has information $\Omega_b = 0.1\Omega_a$. Each measurement source also has a binary variable describing whether it is acquired with a parameter dependent on the placement of the sensor. As such, acquisition variable γ_a reaches a peak probability of success

Figure 5.2 Comparison of analytic and sampling strategies for representing the posterior belief distribution. (a) Regarding the trace of the robot covariance, the analytic approach we propose leads to the same expected value as the sampling approach, and (b) the same trend in variance. (c) (d) The analytic approach is significantly faster to compute compared to sampling measurement and acquisition sequences (sampling shown for 100 and 1000 samples).



(a) Expected Value

(b) Variance (Log Scale)

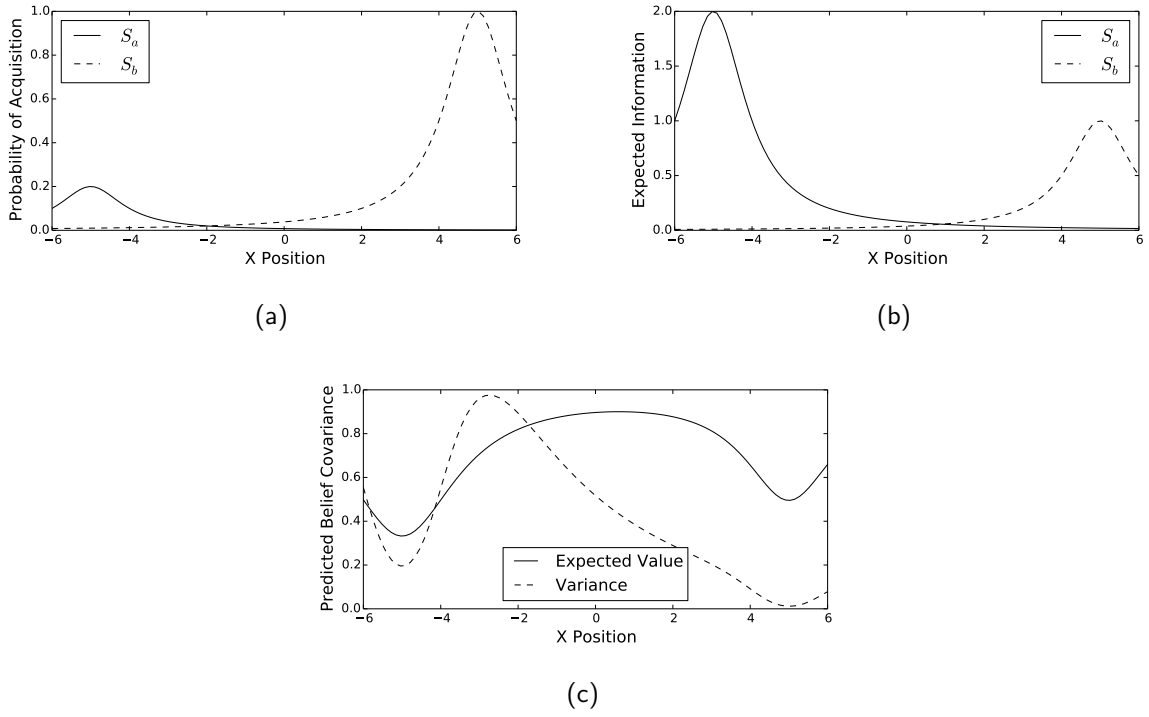
(c) Timing

(d) Timing (Log Scale)

of $\max p(\gamma_a = 1) = 0.2$ at $x = -5$. Acquisition variable γ_b reaches a peak probability of success of $\max p(\gamma_b = 1) = 1.0$ at $x = 5$. The state-dependent parameter functions for the acquisition variables are shown in Fig. 5.3(a). Each measurement source contributes expected information $\mathbb{E}[\Lambda_i] = p(\gamma_i = 1)\Omega_i = \lambda_i \Omega_i$, shown over the one-dimensional environment in Fig. 5.3(b). At their respective peak probabilities of acquisition, S_a contributes twice the expected information than source S_b, as S_a is 10 times more informative but S_b is 5 times more likely to be acquired. Using the method from Chapter 2 and the proposed analytic representation of the posterior belief distribution, Fig. 5.3(c) shows the predictions of the expected value and variance of the belief covariance for placing the sensor along the environment.

Consider the simple risk-neutral, linear objective function of $tr(\Lambda^{-1})$, graphed in Fig. 5.4(a). With an initial sensor placement of $x = 0$ and a gradient descent update framework, the sensor follows the gradient and converges to a placement of $x = -5$. Now consider minimizing the risk-averse penalty functions for quadratic ((5.16)), exponential ((5.19)), and power ((5.22)) utilities. These objective functions account for the uncertain measurement acquisition and are graphed in Fig. 5.4. In these cases, the sensor initially placed at $x = 0$ converges to the

Figure 5.3 One-dimensional intuition. (a) The functions describing the probability of acquisition for each measurement source in the one-dimensional example. (b) The expected value of information contributed to the belief by a measurement from each source. (c) The predictions for the expected value and variance of the belief covariance as a function of sensor placement.



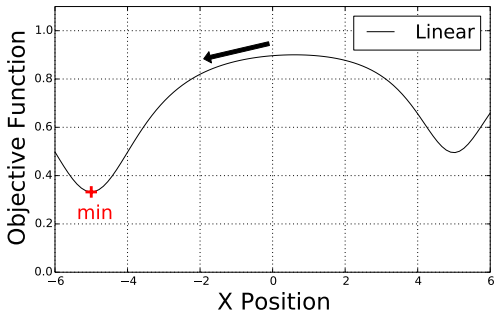
risk-averse location of $x = 5$. Monte Carlo simulations of the resulting covariance from each placement are shown in Fig. 5.4(e) and (f). While the placement at $x = -5$ often yields a very low uncertainty, it also often receives no measurements. The placement at $x = 5$ is guaranteed to receive the measurement from S_b.

Fig. 5.5 shows the change in each of the penalty functions when varying the parameters involved. The quadratic function (5.16) exhibits a wide array of risk-sensitive behaviors but is fairly restrictive in terms of valid parameter ranges. The exponential function (5.19) easily adjusts the sensitivity to variance by changing the value of η_e. The power function in Fig. 5.5(c) shows a more subtle interaction between expected value and variance when modifying the value of η_p, which directly modifies the level of relative risk aversion. This function is also dependent upon the uncertainty bound $\mathcal{T}$, seen in Fig. 5.5(d).

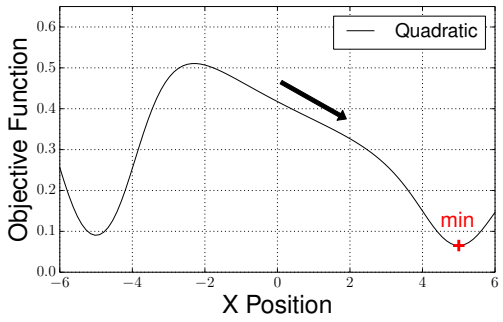
5.4.3 Planar Robot

We can apply the intuition from the previous example to a planar robot with control authority along the x and y directions. Rather than placing a sensor, we are interested in localizing

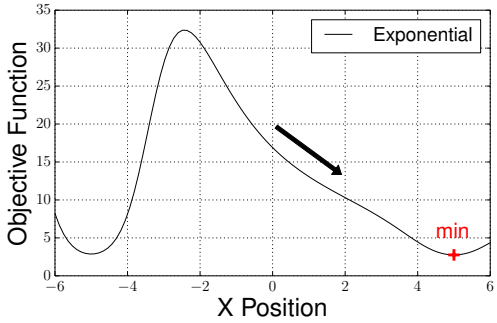
Figure 5.4 One-dimensional intuition. (a) A linear objective function of $J = \Lambda^{-1}$, which simply minimizes the expected value of the predicted belief covariance. (b) The quadratic objective function of the form of (5.16), with $\eta_q = 0.7$ and $\mathcal{T} = 1.5$. (c) The exponential objective function of the form of (5.19), with $\eta_e = 2$. (d) The power objective function of the form of (5.22), with $\eta_p = 4$ and $\mathcal{T} = 1.5$. (e)(f) The resulting covariance for 1000 trials of Monte Carlo simulations of the sensor belief, placed at $x = \{-5, 5\}$.



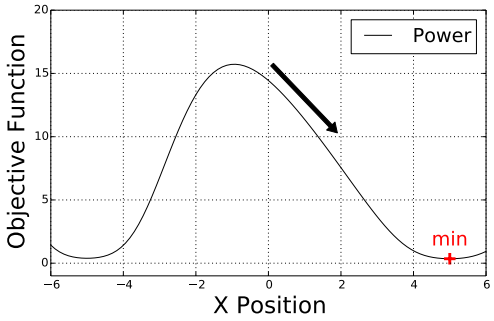
(a)



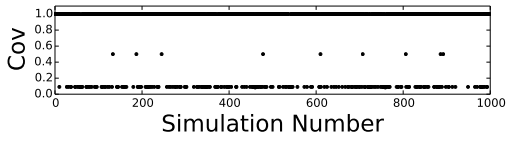
(b)



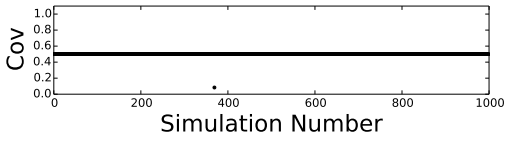
(c)



(d)

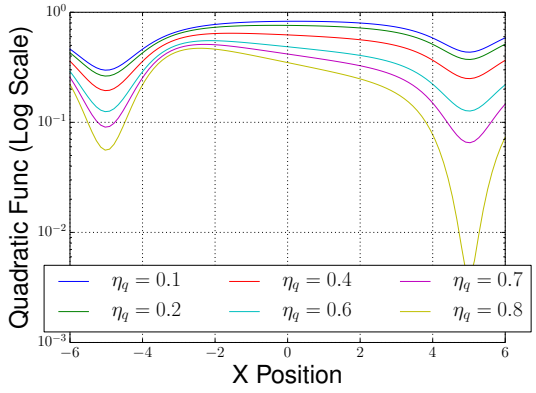


(e) $x = -5$

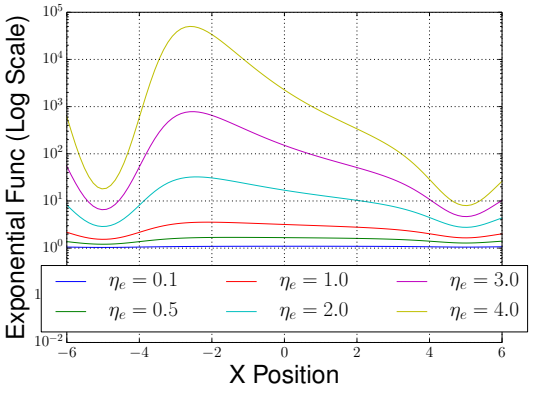


(f) $x = 5$

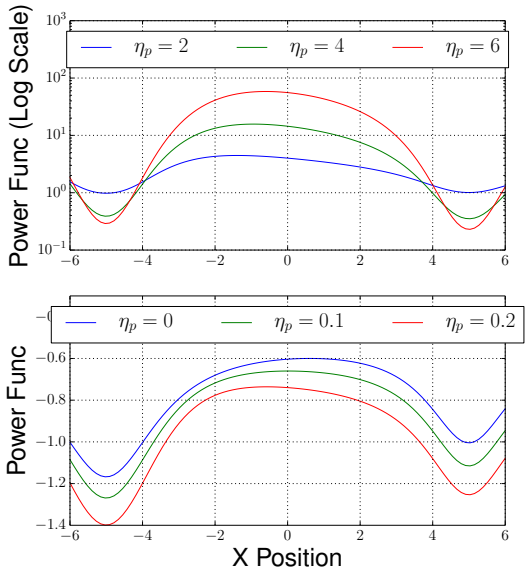
Figure 5.5 One-dimensional intuition. (a) The quadratic objective function graphed for varying values of the parameter η_q with $\mathcal{T} = 1.5$. (b) The exponential objective function graphed for varying values of the parameter η_e. (c) The power objective function graphed for varying values of the parameter η_p with $\mathcal{T} = 1.5$. (d) The power objective function graphed for varying values of the uncertainty bound $\mathcal{T}$ with $\eta_p = 4$.



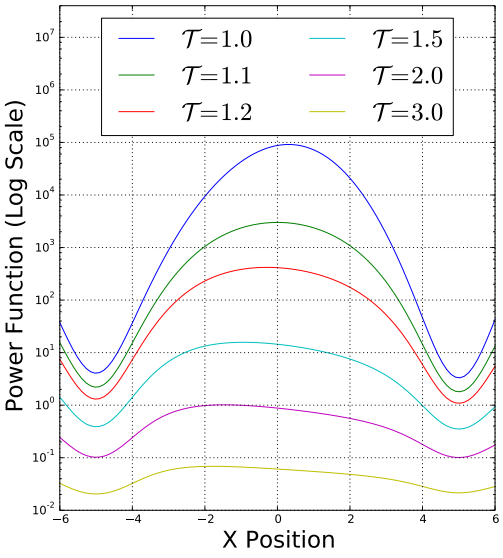
(a) Quadratic, varying η_q



(b) Exponential, varying η_e



(c) Power, varying η_p



(d) Power, varying $\mathcal{T}$

the robot along a trajectory and reaching a goal region (Fig. 5.6). Similar to the previous example, the robot receives absolute measurements in x and y from two different sources of constant information. The information and acquisition properties for these sources are the same as in the one-dimensional example (Fig. 5.3(a), (b)) but extend along the y-direction. The robot starts at pose $(x, y) = (0, 0)$ with the goal of reaching a final pose at planning step $K = 10$ with $y = 20$. We update the trajectory using a gradient descent optimization and use this example to show the applicability of our formulation to trajectory-smoothing planners.

In this example, we compare our proposed risk-averse planner to the method of Indelman et al. [67]. Their method uses the following forms of the costs in (2.25) and (2.26):

$$\begin{aligned} g_u(\mathbf{u}_k) &= \|\delta(\mathbf{u}_k)\|^2_{M_u}, \\ g_\Lambda({\Lambda_k}^{-1}) &= \mathbf{m}^\top \mathrm{vec}({\Lambda_k}^{-1}), \\ g_\mathbf{x}({\mathbf{x}_K}^* - \mathbf{x}_G) &= \|{\mathbf{x}_K}^* - \mathbf{x}_G\|^2_{M_\mathbf{x}}, \end{aligned} \tag{5.23}$$

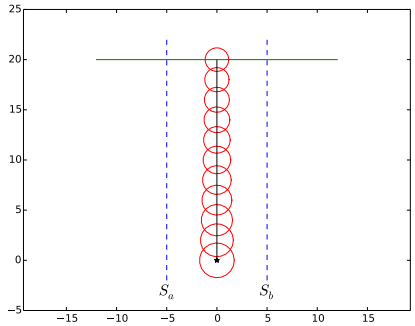
where $\delta(\,\cdot\,)$ is the diversion from the nominal control input and M represents a weight matrix. The Indelman et al. approach does not consider the randomness of the acquisition variables and is unaware of risk in the belief covariance. With this method, the robot settles on a path traveling the peak acquisition zone of S_a, shown in Fig. 5.6(c). However, our proposed framework accounts for the randomness of the acquisition in the belief covariance matrix by replacing the uncertainty cost above with $g_\Lambda({\Lambda_k}^{-1}) = \mathscr{P}(W)$, where $\mathscr{P}(W)$ represents the penalty functions of (5.16), (5.19), and (5.22). With the risk-averse optimization, the robot prefers a path traveling the peak acquisition zone of S_b, shown in Fig. 5.6(e).

The effect of the randomness of acquisition is clearly seen when we simulate 1000 runs of the robot traversing the selected paths. Fig. 5.6(b), (d), and (f) show the trace of the marginal covariance of the belief at the final planning step for the simulations. The uncertainty is often lower with the Indelman et al. method, but the proposed method path far outperforms the path from Indelman et al. with respect to worst-case uncertainties. The risk-averse path consistently receives measurements for improved localization.

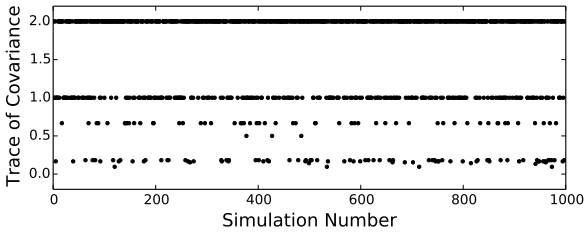
5.4.4 Active SLAM Decisions

Consider a Hovering Autonomous Underwater Vehicle (HAUV) performing visual SLAM to inspect a ship hull, as in Fig. 1.3. The robot executes a lawnmower-like trajectory over the underwater portion of the hull to collect camera images of the environment in a coverage-efficient manner. However, open-loop execution of this policy results in navigation drift, so the robot must perform loop-closing revisit actions throughout the mission to bound its uncertainty. Loop-closures in the visual SLAM formulation come from pairwise camera

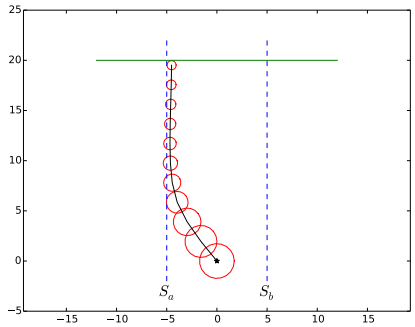
Figure 5.6 Results from the planar robot example. (a) The initial trajectory. (c) The resulting trajectory from the gradient-based optimization method of Indelman et al. [67] with an objective function that is linear in the belief covariance. (e) The resulting trajectory from the gradient-based optimization with the proposed risk-averse formulation using an objective that includes a penalty function of the belief covariance (power function results shown). (b)(d)(f) The trace of the robot's terminating covariance for 1000 trials of Monte Carlo simulations of the above trajectories. The proposed risk-averse method results in a path that consistently receives measurements for improved localization.



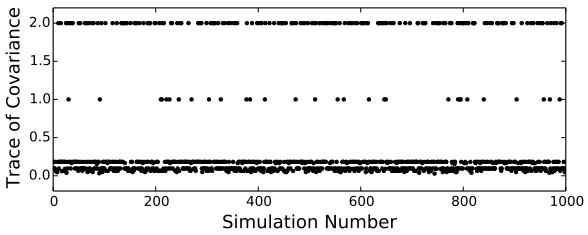
(a) Initial trajectory



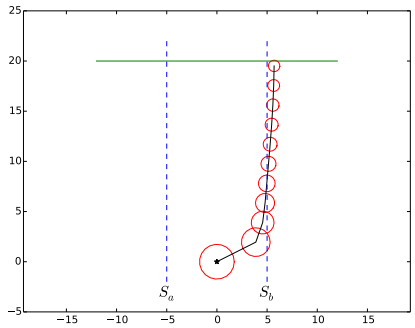
(b)



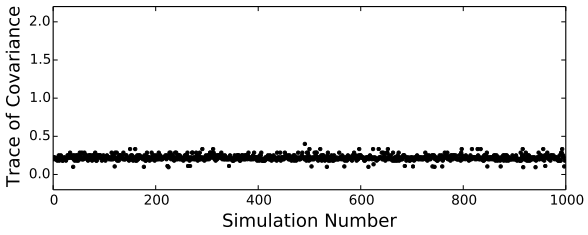
(c) Indelman et al. [67]



(d)



(e) Proposed



(f)

registrations between overlapping images.

For this simulation, we design a synthetic environment where most of the environment is feature-less and has zero registrability. We use a measure of visual saliency [80] and a Gaussian process (GP) prediction [20] to model the registration acquisition variables throughout the environment. Given a path planning algorithm for finding possible revisit paths, we evaluate a candidate path based upon its distance traveled and its final uncertainty, which we frame as a risk-averse penalty function expressing quadratic, exponential, or power utility. The objective function becomes

$$J = g_u(U_{0:K-1}) + \underset{\Gamma_{1:K}}{\mathbb{E}} \left[\mathscr{P} \left(\mathcal{T} - \mathbf{m}^\top \mathrm{vec}({\Lambda_K}^{-1}) \right) \right], \tag{5.24}$$

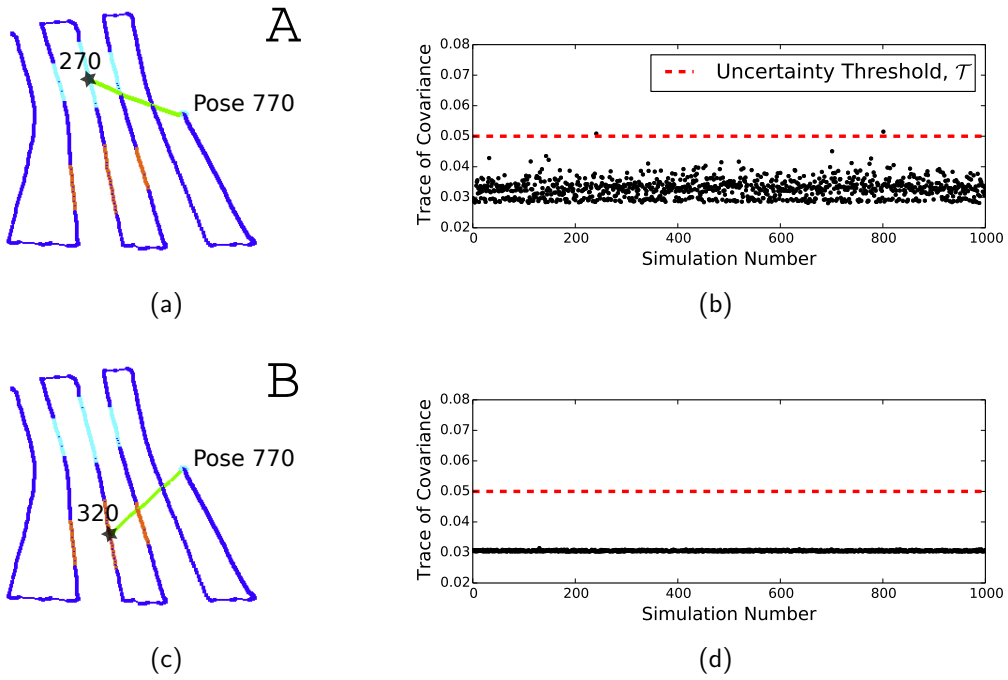
where $g_u(U_{0:K-1})$ computes the path length (scaled by a weight), the threshold is set to $\mathcal{T} = 0.05$, and $\mathbf{m}$ selects the diagonal elements of the final pose marginal covariance. We show the benefit of the proposed risk-averse framework within this type of sampling-based active SLAM system by comparing to the path evaluation method of our previous work in Chapter 3 [20].

Fig. 5.7 shows the HAUV deciding between two candidate loop-closure paths at pose number 770 of the mission. Candidate path `A` considers revisiting a moderately-salient portion of the environment centered at pose number 270 in the graph. Candidate path `B` considers revisiting a more salient area centered at pose 320 in the graph. Here we see the tradeoff illustrated throughout this chapter: path `A` has a high risk-reward ratio. Registering to poses along path `A` provides greater information gain as the resulting loop-closures are larger than loops closed via path `B`. However, the higher visual saliency for images along path `B` means that registrations to these poses are more likely to occur than those along path `A`.

Both the previous and proposed methods select path `A` in this scenario. Table 5.1 presents statistics from each method related to the selection, including evaluation times. It also presents the number of proposed camera registrations for each path, the average probability of acquisition of these hypotheses, and the expected values and variances of the uncertainties predicted using the methodology from §5.2. We overlay these predictions on the penalty function contour plots of Fig. 5.8. Despite the higher variance and longer length, path `A` has a lower expected uncertainty than path `B`. We see why preferring path `A` is sensible given the Monte Carlo simulation results for traveling each path in Fig. 5.7(b) and (d). Only 2 trials in 1000 from path `A` result in uncertainties greater than the threshold and many trials outperform the resulting uncertainties of traveling path `B`.

We investigate a second scenario later in the mission. At pose number 1145, the robot again decides between revisiting the same locations along paths `A` and `B`. This time, the

Figure 5.7 Results from the first HAUV planning scenario. (a) The trajectory of revisit path `A` from pose 770 to pose 270 and back. (b) The trace of the final pose covariance for 1000 trials of a Monte Carlo simulation of traveling path `A`. (c) The trajectory of revisit path `B` from pose 770 to pose 320 and back. (d) The trace of the final pose covariance for 1000 trials of a Monte Carlo simulation of traveling path `B`.

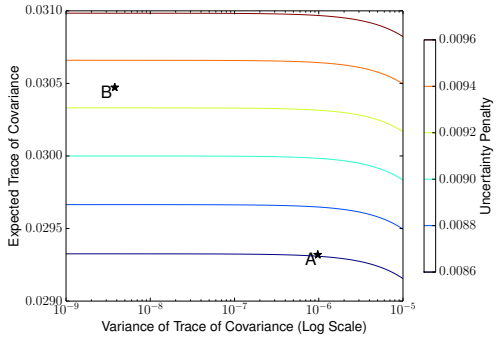


robot is farther along in the mission and must travel farther to close loops, leading to higher predicted uncertainties than in the first scenario. Here, the previous method of Chapter 3 [20] once again selects path `A`. Interestingly, the quadratic function ($\eta_q = 20$) also selects this path, as one downfall of this function is that it becomes *less* risk-averse as the wealth decreases. In contrast, the proposed exponential and power methods (with $\eta_e = 400$ and $\eta_p = 4$) prefer path `B` even though they predict a higher expected uncertainty and longer traveling distance than path `A`. Fig. 5.8 shows how the penalty function contours change given the much closer predicted proximity to the threshold. The Monte Carlo simulations of Fig. 5.9(b) and (d) show why choosing path `B` is desirable in this case. Traveling path `A` results in 147 trials of 1000 that exceed the uncertainty threshold, but the robot can confidently travel path `B` without concern.

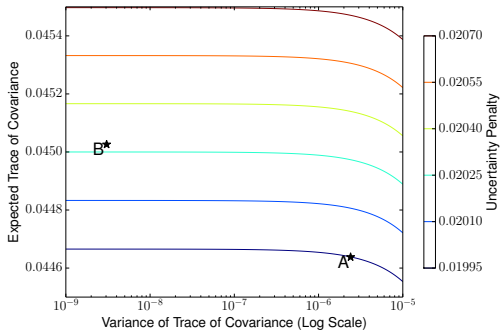
5.4.5 Hybrid Simulation

To further illustrate the proposed method, we present active SLAM results from the hybrid simulation first outlined in §3.4. In this experiment, the HAUV surveys the target environment

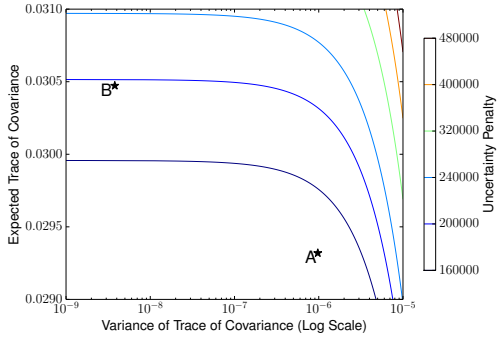
Figure 5.8 The uncertainty penalty function contours for the HAUV planning scenarios. (Left column) At pose 770. (Right column) At pose 1145.



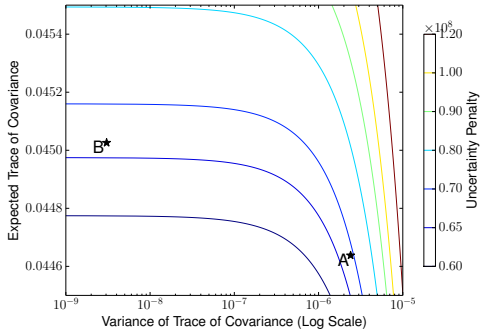
(a) Quadratic, $\eta_q = 20$



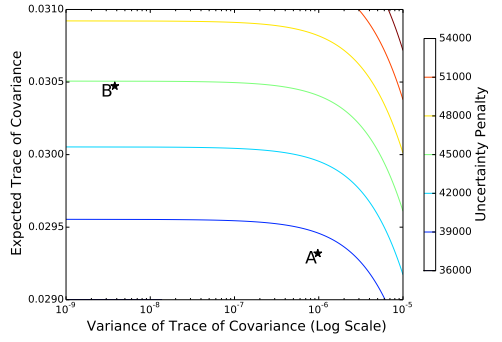
(b) Quadratic, $\eta_q = 20$



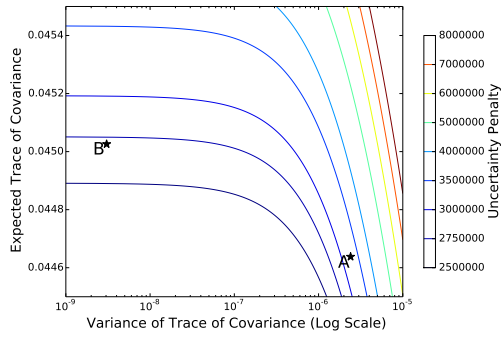
(c) Exponential, $\eta_e = 400$



(d) Exponential, $\eta_e = 400$



(e) Power, $\eta_p = 4$



(f) Power, $\eta_p = 4$

Table 5.1 Predictions & Statistics for HAUV Active SLAM Decisions

Decision at Pose 770	**Path** A	**Path** B
Distance [m]	21.23	20.45
Registration Hypotheses	39	37
Avg. $p(\gamma_{i,j} = 1)$	0.232	0.804
$\mathbb{E}[\mathbf{m}^\top \text{vec}({\Lambda_K}^{-1})]$	0.02932	0.03047
$\text{Var}[\mathbf{m}^\top \text{vec}({\Lambda_K}^{-1})]$	9.743E-07	3.791E-09
PREVIOUS METHOD[20]		
Evaluation Time [ms]	45.73	59.29
Selected Path—Linear	A	
PROPOSED METHODS		
Evaluation Time [ms]	117.75	138.66
Selected Path—Quadratic	A	
Selected Path—Exponential	A	
Selected Path—Power	A	

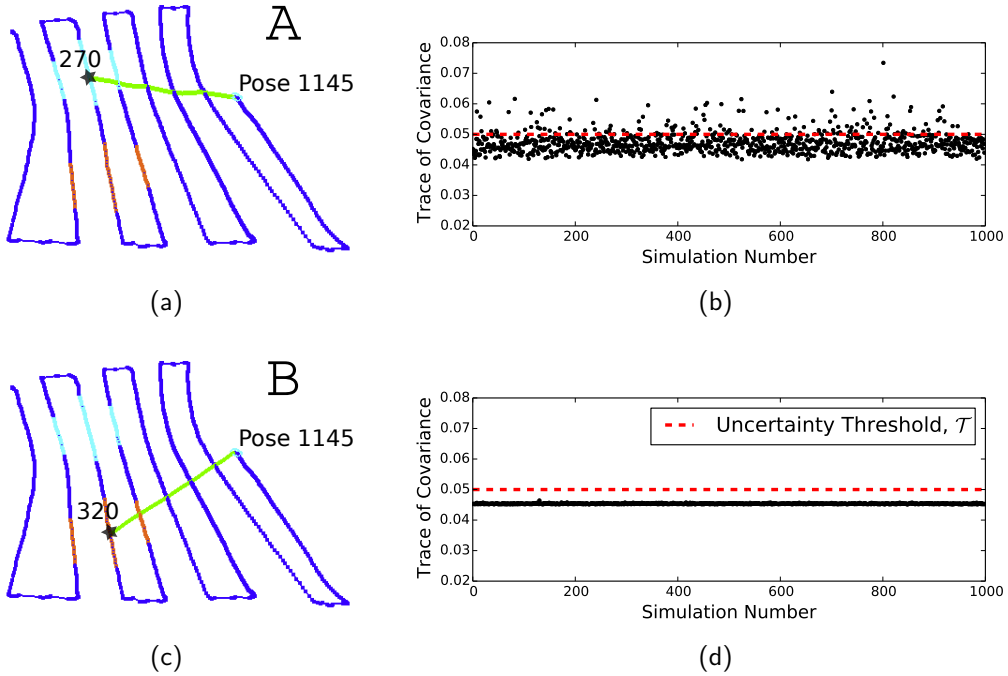
Decision at Pose 1145	**Path** A	**Path** B
Distance [m]	31.79	32.59
Registration Hypotheses	35	33
Avg. $p(\gamma_{i,j} = 1)$	0.276	0.893
$\mathbb{E}[\mathbf{m}^\top \text{vec}({\Lambda_K}^{-1})]$	0.04464	0.04503
$\text{Var}[\mathbf{m}^\top \text{vec}({\Lambda_K}^{-1})]$	2.400E-06	3.048E-09
PREVIOUS METHOD[20]		
Evaluation Time [ms]	90.03	100.46
Selected Path—Linear	A	
PROPOSED METHODS		
Evaluation Time [ms]	150.36	198.50
Selected Path—Quadratic	A	
Selected Path—Exponential		B
Selected Path—Power		B

following the nominal exploration policy with sonar-spaced tracklines. At specific points in the mission, the robot is presented with the decision to revisit to the salient regions centered at poses 270 and 320, as in the previous experiment. We evaluate each option according to the simple objective function,

$$J = \underset{\Gamma_{1:K}}{\mathbb{E}} \left[\mathscr{P} \left(\mathcal{T} - \mathbf{m}^\top \text{vec}({\Lambda_K}^{-1}) \right) \right]. \tag{5.25}$$

Here, we compare the linear function of directly minimizing the expected value of the trace of the covariance against the power penalty function of (5.22). For each objective, we run the hybrid simulation and allow the active SLAM framework to decide the trajectory. Then, we

Figure 5.9 Risk-averse planning for an underwater robot performing visual SLAM. The robot must decide between path `A` (a) and path `B` (c) for gathering loop-closure camera registrations. Previous methods select path `A`, which has a lower predicted uncertainty and shorter path length than `B`, but the proposed risk-averse planning framework selects path `B`. Monte Carlo simulations in (b) and (d) show that path `B` is indeed preferable. Path `A` exceeds the final pose uncertainty threshold in 147 of 1000 trials.



perform multiple simulations of the resulting trajectory to examine the effect of the random camera registration acquisitions on the localization uncertainty.

Results from 100 simulations using each objective function (linear and power utilities) are shown in Fig. 5.10. These graphs plot the robot navigation uncertainty versus the distance traveled throughout the mission. Fig. 5.10(b) shows the wide variability of the uncertainty resulting from the linear objective often selecting the moderately-salient revisit waypoint at pose 270. The power function characterizes the risk involved in attempting loop-closure registrations near pose 270, and thus generally prefers the revisit waypoint at pose 320. Simulations from this trajectory are shown in Fig. 5.10(d). As loop-closures along this trajectory are much more likely to be acquired, the uncertainty simulations are tightly distributed.

5.5 Discussion

The above results show how the power penalty function naturally lends itself to desirable behavior in active SLAM. While the uncertainty is low, the robot is willing to make risky decisions for possible high rewards. But as the uncertainty grows, approaching the threshold, the robot exhibits greater risk aversion, making more conservative but safer decisions. This is in contrast to the quadratic function, which exhibits decreasing aversion to risk as the wealth decreases (the uncertainty grows). In addition, the quadratic function is generally difficult to tune, as the range of valid parameter values is restricted. Outside this range, the quadratic function begins experiencing negative marginal utility and violates the first axiom of rational behavior [71].

The exponential function removes the dependence on the uncertainty threshold $\mathcal{T}$ and the risk parameter η_e directly controls the absolute risk aversion, supporting its wide investigation in previous literature. However, rational investor behavior is defined by constant relative risk aversion, where the absolute amount of risk tolerated is proportional to the amount of wealth involved, and this is not reflected in the exponential function. The active SLAM application features large gains and drops in uncertainty as the robot trades off exploration and information-gathering, so it is sensible to model constant relative risk aversion across the uncertainty spectrum. For this reason, we prefer the power utility function.

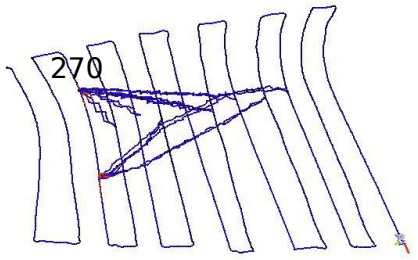
It is worthwhile to note that we are able to calculate these utility functions because we approximate the posterior belief distribution as Gaussian and select the trace of the robot's terminating covariance as the measure of uncertainty. Rather than explicitly computing the moments in (5.3), we go one step further and only compute the elements necessary for the objective functions. In the case of the trace, sparsity of the element selection vector $\mathbf{m}$ greatly reduces the computational load.

5.6 Chapter Summary

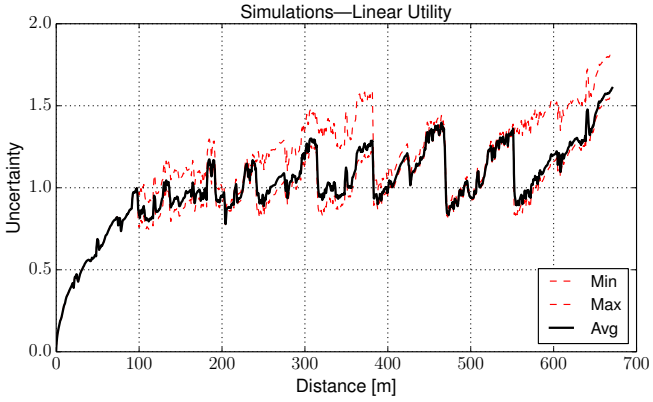
In this chapter, we proposed a risk-averse framework for planning in belief-space with random measurements and random acquisition variables. We leveraged the randomness in the belief covariance to design objective functions for the planning problem that characterize aversion to risk, inspired by expected utility theory in modern portfolio optimization. The basis of our risk-averse planning framework is an approximation to the posterior belief distribution, where we assume a Gaussian distribution and analytically compute first and second moments. We apply this method to quantify the expected value and variance of the trace of the robot's terminating covariance for a sequence of candidate control actions, and use these moments to

evaluate the risk-averse objective function. Because it encodes rational decision-making in stochastic problems, we prefer the power utility function over other common utilities like quadratic and exponential. Our simulation results show that risk-averse path planning for mobile robotics applications yields more desirable outcomes than paths found with previous approaches, using both trajectory-smoothing and sampling-based frameworks. Specifically, we show the positive performance of the power utility function in navigation under uncertainty and active SLAM scenarios.

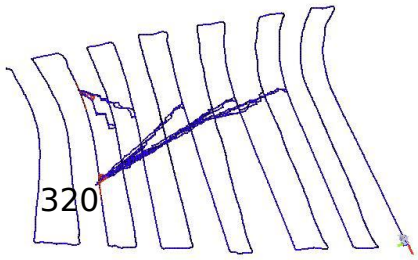
Figure 5.10 Active SLAM hybrid simulations for an underwater robot performing ship hull inspection. Decisions to revisit to two salient waypoints are made at specific points throughout the mission. (a)(b) The plot shows 100 simulations of traveling the trajectory resulting from decision-making with a linear utility objective function. The linear function minimizes the trace of the final robot covariance and often selects the revisit waypoint in a moderately-salient region centered at pose 270. Despite a low expected uncertainty, simulations show a wide variability in outcome due to the randomness in acquisition along this path. (c)(d) With the power penalty function, the robot often prefers the revisit waypoint at pose 320, where camera registrations are more likely to be acquired. 100 simulations of this resulting path show a much tighter distribution of outcomes.



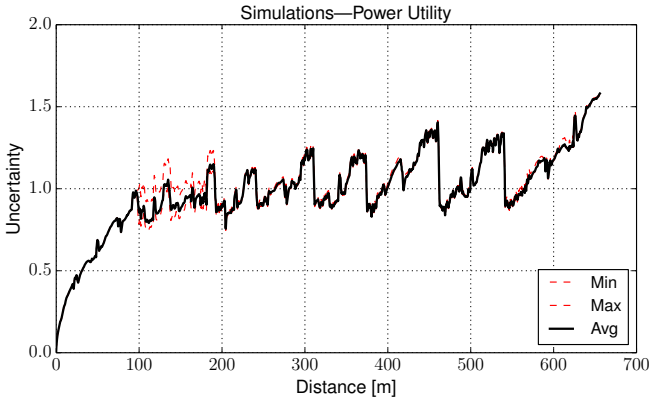
(a) Linear Utility—Trajectory



(b) Linear Utility—Simulations



(c) Power Utility—Trajectory



(d) Power Utility—Simulations

Chapter 6

Conclusions

Autonomous mobile robots operating in *a priori* unknown environments demand a comprehensive framework that integrates the core competencies of localization, mapping, and planning. This thesis proposed work in active visual simultaneous localization and mapping (SLAM) for underwater environments, with a focus on autonomous ship hull inspection with the Hovering Autonomous Underwater Vehicle (HAUV). The contribution areas of this work can be loosely characterized by the three main components of an active SLAM system: (*i*) search, (*ii*) evaluation, and (*iii*) selection. Below we outline the contributions in relation to these areas.

6.1 Contributions

We proposed a Gaussian belief-space planning formulation in Chapter 2 as an approximation to the underlying partially observable Markov decision process (POMDP), where we represent the robot belief as a Gaussian distribution and track its evolution using Bayesian inference. By predicting the belief into the future, we can perform an optimization to select a sequence of control actions for active SLAM. In the underwater environment, visual loop-closures are difficult to predict, so our formulation considers the randomness in future measurements and randomness in their acquisitions. This leads to a random posterior belief distribution. We also proposed the use of Gaussian process (GP) regression to model the probability of registrations throughout the environment, allowing us to accurately predict future measurements for planning. The developed formulation serves as the mathematical foundation for the other chapters in this thesis.

Search

In Chapter 3, we proposed a full active SLAM framework for autonomous ship hull inspection in order to efficiently survey a target environment subject to bounds on navigation uncertainty. As a main contribution, we combined sampling-based planning techniques with our formulation of Chapter 2 to search hundreds of possible revisit paths throughout the environment. We also proposed an opportunistic approach to selecting revisit actions that allows the robot to gather loop-closure measurements at any point during the mission. Our algorithm was tested in hybrid simulations and real-world field trials of seafloor and hull-relative inspections.

Evaluation

Information-theoretic objectives required for quantifying the benefit of candidate actions are expensive to evaluate. Therefore, we proposed two concepts in Chapter 4 related to reducing the complexity of planning evaluations. First, we investigated compressing the representation of the predicted belief with an online graph sparsification algorithm, and showed that planning with an approximate distribution in place of the original system can save computation time while preserving the same decision-making outcomes. Second, we proposed the use of the Bayes tree as an efficient data structure for planning in active SLAM. The Bayes tree allowed us to identify constrained variable orderings and cache subtrees in order to reuse computations between candidate plan evaluations.

Selection

Finally, Chapter 5 detailed our contributions toward designing objective functions for active SLAM that are risk-averse with respect to localization uncertainty. We proposed an analytic approach to approximating the posterior belief distribution as Gaussian. Inspired by expected utility theory, we examined rational behavior under stochastic outcomes within the optimization, and showed that our proposed methodology leads to desirable action selections for active SLAM.

6.2 Future Work

There are many potential areas of future work motivated by the methods and results in this thesis. Some of the more interesting avenues are discussed below.

Improvements to the Proposed Methods

There are a number of ways the methods proposed in this document can be improved. First, empirical evidence suggests that the HAUV belief is not always adequately represented by a Gaussian distribution, specifically in cases of failed data association or external, unmodeled disturbances. Accounting for non-Gaussian beliefs in the planning formulation is an open area of research in the belief-space planning community.

Second, the algorithmic contributions of this thesis all motivate follow-up questions that are worth pursuing. For example, the online graph sparsification algorithm prompts the question of if/when the linear constraints resulting from approximate marginalization with generic linear constraint (GLC) will be sufficient to describe the removed nonlinear factors.

Coverage Planning

One natural way to extend the research in active SLAM is to remove the dependence on the nominal exploration policy from the proposed framework. Throughout this document, the nominal exploration policy serves as the baseline for ensuring full area coverage and providing high-level guidance through the environment. The candidate actions we consider are all diversions from this policy to gather loop-closures, but all eventually return to this default behavior. Future work into full coverage planning is necessary in order to remove this dependence.

Multi-session and Multi-robot Planning

All of the inspections considered in this document are single-vehicle, single-session missions. However, there is ongoing work within the HAUV project toward multi-session SLAM capabilities [102], wherein the robot co-registers separate inspection dives to a single reference frame. In fact, the multi-session approach is a virtual necessity in order to densely survey very large environments due to finite battery life, frequent operational faults, and limited access to the ships. The active SLAM problem changes significantly considering the multi-session context: there may be prior information about the environment that can be exploited, there may be multiple uncertain reference frames to account for in the prediction, etc.

In a related concept, the multi-session capabilities can be expanded to multi-robot capabilities, where two or more vehicles simultaneously survey a single environment in a cooperative fashion. The active SLAM problem changes in this setting as well. Planning for cooperative actions can improve overall performance [134], robots may be equipped with different perceptual sensors that can provide heterogeneous information about the environment, and distribution and communication between vehicles now becomes an issue.

Appendix A

Hovering Autonomous Underwater Vehicle

This appendix provides information on the Hovering Autonomous Underwater Vehicle (HAUV).

A.1 Payload Information

The sensor payload of the HAUV is provided in Fig. A.1 and Table A.1.

Figure A.1 HAUV payload diagram.

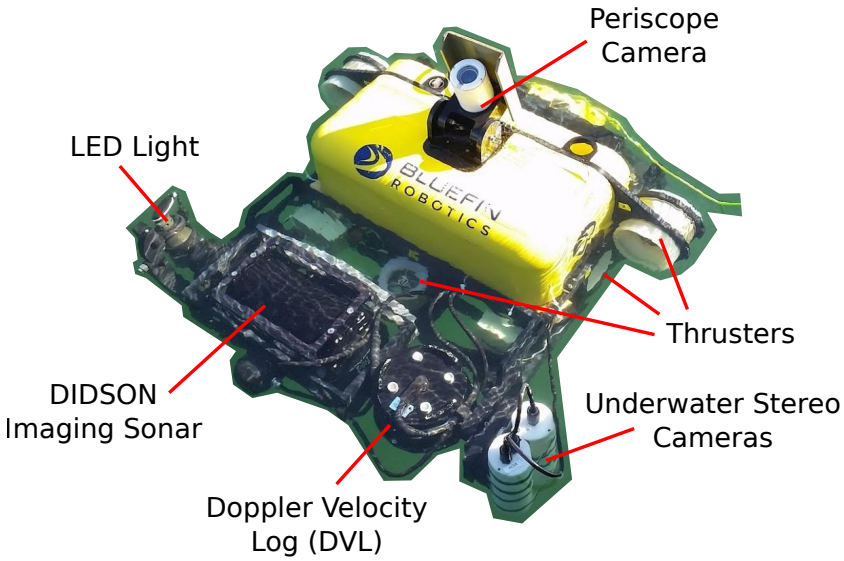


Table A.1 Payload Characteristics of the HAUV

	Description
Prosilica GC1380	12-bit digital stills, fixed-focus, monochrome, 1 Megapixel
Periscope Camera	Monocular Prosilica GC1380 in water-proof housing
Underwater Camera (monocular, pre-2013)	Monocular Prosilica GC1380 in water-proof housing
Underwater Camera (stereo, post-2013)	Two Prosilica GC1380s in separate water-proof bottles, linked via Fast Ethernet
Lighting	520 nm (green) LED
IMU	Honeywell HG1700
Depth	Keller pressure, 1-σ noise at 10 cm
Imaging Sonar	Sound Metrics 1.8 MHz DIDSON
DVL	RDI 1200 kHz Workhorse; also provides four range beams
Thrusting	Five rotor-wound thrusters
Battery	1.5 kWh lithium-ion
Dry Weight	79 kg
Dimensions	1 m $\times$ 1 m $\times$ 0.45 m

A.2 Waypoint Controller

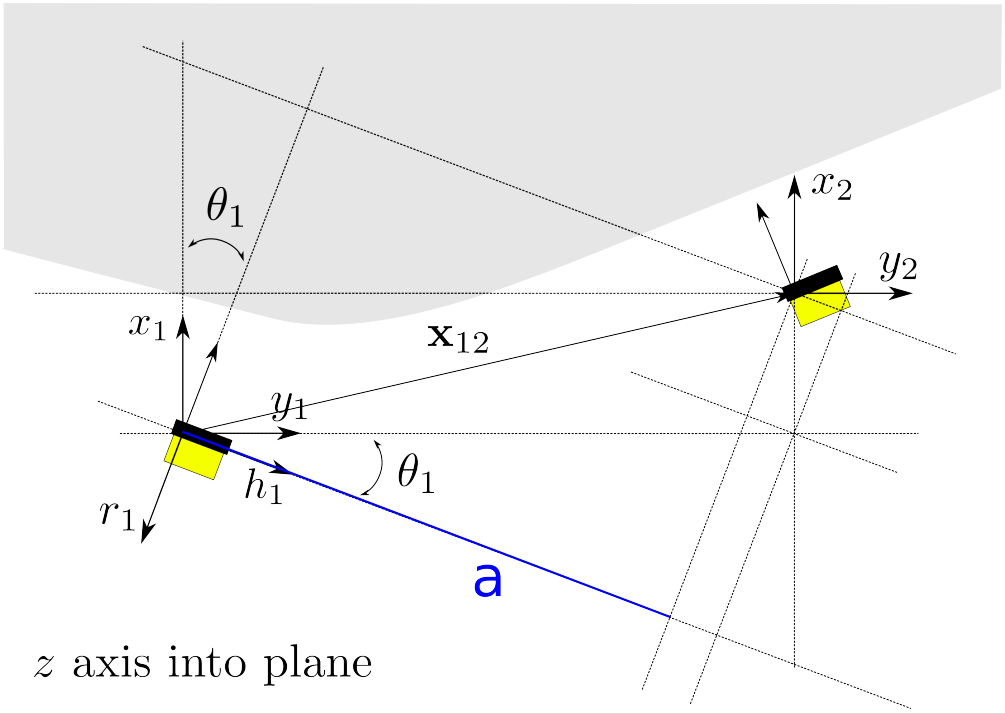
The simultaneous localization and mapping (SLAM) system reports poses in the 3D Cartesian coordinate frame $\{x, y, z\}$. The HAUV internally reports poses in the hull-relative frame $\{h, v, s\}$, where h is the horizontal along-hull distance over the hull manifold, v is the vertical along-hull distance over the hull manifold, and s is the distance from the hull (normal to the manifold). In order to send waypoint commands to the vehicle, we must transform the relevant poses from the SLAM coordinate frame to the HAUV hull-relative frame. This is a complex computation, so we transform between these frames with a series of approximations that are continuously recomputed (5 Hz or 10 Hz) by the waypoint controller software. These approximations are detailed in Fig. A.2 and Fig. A.3. Each approximation consists of computing the straight-line vector from the current pose to the desired pose and projecting this vector into the local hull-relative coordinate frame, done in two steps.

The first step in the transformation from $\{x, y, z\}$ to $\{h, v, s\}$ is the rotation about the z-axis to the intermediate frame $\{h, r, z\}$, shown in Fig. A.2:

$$\begin{aligned} h &= -x \sin\theta + y \cos\theta, \\ r &= -x \cos\theta - y \sin\theta, \\ z &= z, \end{aligned} \tag{A.1}$$

where θ is the heading of the current vehicle pose. The second step is the rotation about the

Figure A.2 The vector $\mathbf{x}_{12}$ represents the offset from the current pose $\mathbf{x}_1$ to the desired pose $\mathbf{x}_2$, expressed in the SLAM frame. We must express this vector in the vehicle's hull-relative frame before sending control commands. The first step to transform from SLAM frame $\{x, y, z\}$ to HAUV hull-relative frame $\{h, v, s\}$ is to transform to the intermediate frame $\{h, r, z\}$ by rotating about the z-axis by the vehicle's current heading, θ. The distance 'a' approaches the true horizontal error between current pose $\mathbf{x}_1$ and desired waypoint $\mathbf{x}_2$.



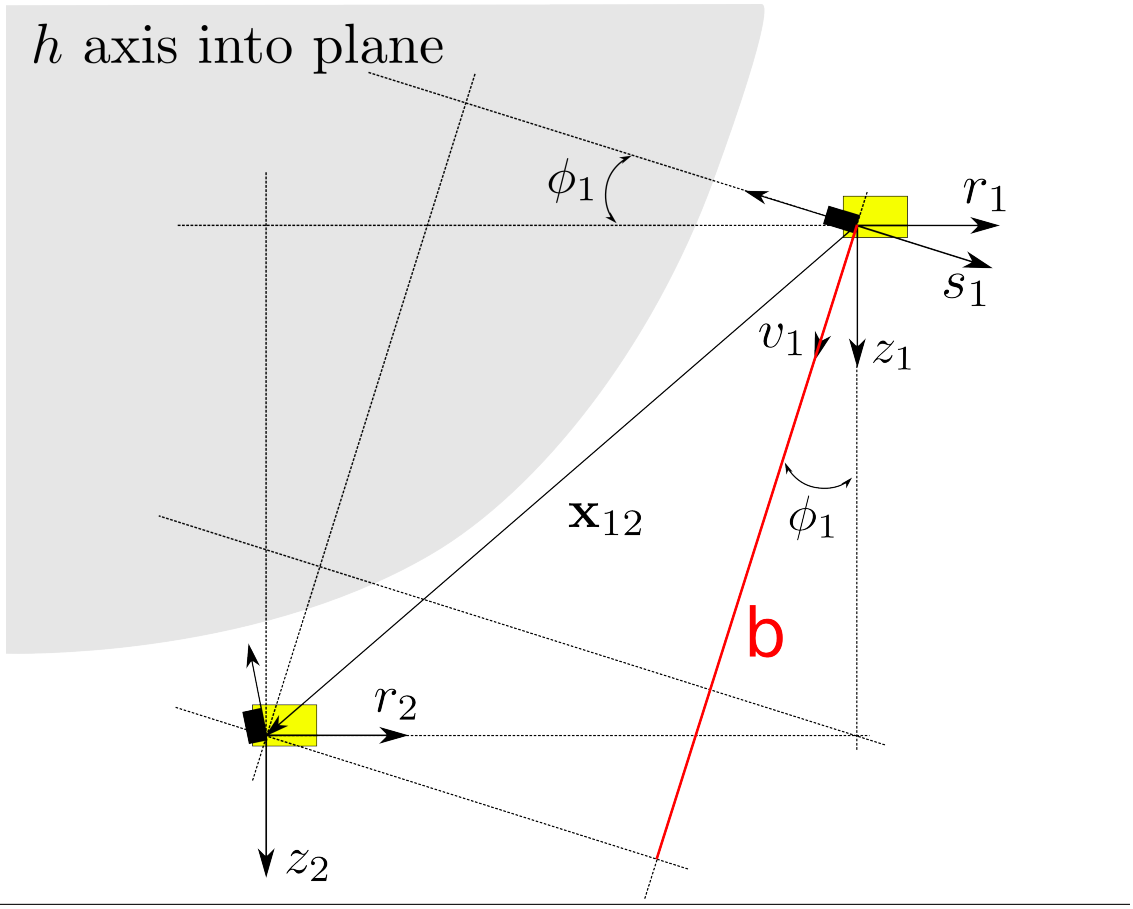
h-axis to the final $\{h, v, s\}$ frame, shown in Fig. A.3:

$$\begin{aligned} h &= h, \\ v &= -r \sin\phi + z \cos\phi, \\ s &= r \cos\phi + z \sin\phi, \end{aligned} \tag{A.2}$$

where ϕ is the current Doppler velocity log (DVL) angle.

As the vehicle travels from current pose $\mathbf{x}_1$ to desired waypoint $\mathbf{x}_2$, the distance 'a' in Fig. A.2 represents the error in h. This local approximation of the horizontal error approaches the true horizontal error as the vehicle controls to the desired waypoint. Similarly, the distance 'b' in Fig. A.3 represents the local approximation to the vertical error, which converges to the true vertical error as the vehicle follows the curvature of the hull. The setpoint for distance from the hull (s_2) is set to the distance from the hull at the desired waypoint.

Figure A.3 The final step to transform from $\{x, y, z\}$ to $\{h, v, s\}$ is to rotate about the h-axis by the current DVL angle, ϕ. The distance '`b`' approaches the true vertical error as the vehicle converges to the desired waypoint.



Bibliography

[1] P. Agarwal and E. Olson. Variable reordering strategies for SLAM. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3844–3850, Vilamoura, Portugal, Oct. 2012.

[2] A. Agha-mohammadi, S. Chakravorty, and N. M. Amato. FIRM: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements. *International Journal of Robotics Research*, 33(2):268–304, 2014.

[3] A. Bachrach, S. Prentice, R. He, P. Henry, A. S. Huang, M. Krainin, D. Maturana, D. Fox, and N. Roy. Estimation, planning, and mapping for autonomous flight using an RGB-D camera in GPS-denied environments. *International Journal of Robotics Research*, 31(11):1320–1343, 2012.

[4] H. Bai, D. Hsu, and W. S. Lee. Integrated perception and planning in the continuous space: A POMDP approach. *International Journal of Robotics Research*, 33(9):1288–1302, 2014.

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

[6] R. Bajcsy. Active perception. *Proceedings of the IEEE*, 76(8):996–1005, 1988.

[7] S. Barkby, S. B. Williams, O. Pizarro, and M. V. Jakuba. Bathymetric SLAM with no map overlap using Gaussian processes. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1242–1248, San Francisco, CA, USA, Sept. 2011.

[8] 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.

[9] A. Ben-Tal and M. Teboulle. An old-new concept of convex risk measures: The optimized certainty equivalent. *Mathematical Finance*, 17(3):449–476, 2007.

[10] L. Blackmore, M. Ono, A. Bektassov, and B. C. Williams. A probabilistic particle-control approximation of chance-constrained stochastic predictive control. *IEEE Transactions on Robotics*, 26(3):502–517, 2010.

[11] S. D. Bopardikar, B. Englot, and A. Speranzon. Robust belief roadmap: Planning under uncertain and intermittent sensing. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 6122–6129, Hong Kong, China, May 2014.

[12] F. Bourgault, A. A. Makarenko, S. B. Williams, B. Grocholsky, and H. F. Durrant-Whyte. Information based adaptive robotic exploration. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 540–545, Lausanne, Switzerland, Sept. 2002.

[13] A. Bry and N. Roy. Rapidly-exploring random belief trees for motion planning under uncertainty. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 723–730, Shanghai, China, May 2011.

[14] 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.

[15] H. Carrillo and J. A. Castellanos. *Navigation Under Uncertainty Based on Active SLAM Concepts*, volume 42 of *Studies in Systems, Decision and Control*, pages 209–235. Springer International Publishing, Switzerland, 2015.

[16] H. Carrillo, I. Reid, and J. Castellanos. On the comparison of uncertainty criteria for active SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2080–2087, St. Paul, MN, USA, May 2012.

[17] H. Carrillo, Y. Latif, M. L. Rodriguez-Arevalo, J. Neira, and J. A. Castellanos. On the monotonicity of optimality criteria during exploration in active SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1476–1483, Seattle, WA, USA, May 2015.

[18] A. Charnes and W. W. Cooper. Chance-constrained programming. *Management Science*, 6(1):73–79, 1959.

[19] S. M. Chaves and R. M. Eustice. Efficient planning with the Bayes tree for active SLAM. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Daejeon, Korea, Oct. 2016. Accepted, to Appear.

[20] 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, Sept. 2014.

[21] S. M. Chaves, J. M. Walls, E. Galceran, and R. M. Eustice. Risk aversion in belief-space planning under measurement acquisition uncertainty. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2079–2086, Hamburg, Germany, Sept. 2015.

[22] S. M. Chaves, R. W. Wolcott, and R. M. Eustice. NEEC research: Toward GPS-denied landing of unmanned aerial vehicles on ships at sea. *Naval Engineers Journal*, 127(1): 23–35, 2015.

[23] S. M. Chaves, A. Kim, E. Galceran, and R. M. Eustice. Opportunistic sampling-based active visual SLAM for underwater inspection. *Autonomous Robots*, 2016. In Press.

[24] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. The MIT Press, Cambridge, MA, 2005.

[25] C. K. Chow and C. N. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, IT-14(3):462–467, 1968.

[26] C. Connolly. The determination of next best views. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 432–435, St. Louis, MO, USA, Mar. 1985.

[27] T. Davis, J. Gilbert, S. Larimore, and E. Ng. A column approximate minimum degree ordering algorithm. *ACM Transactions on Mathematical Software*, 30(3):353–376, 2004.

[28] F. Dellaert. Factor graphs and GTSAM: A hands-on introduction. Technical Report GT-RIM-CP&R-2012-002, Georgia Institute of Technology, Sept. 2012.

[29] 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.

[30] 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.

[31] D. Devaurs, T. Simeon, and J. Cortes. Efficient sampling-based approaches to optimal path planning in complex cost spaces. In *Proceedings of the International Workshop on the Algorithmic Foundations of Robotics*, Istanbul, Turkey, Aug. 2014.

[32] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, 1959.

[33] N. E. Du Toit and J. W. Burdick. Probabilistic collision checking with chance constraints. *IEEE Transactions on Robotics*, 27(4):809–815, 2011.

[34] N. E. Du Toit and J. W. Burdick. Robot motion planning in dynamic, uncertain environments. *IEEE Transactions on Robotics*, 28(1):101–115, 2012.

[35] T. Duckett, S. Marsland, and J. Shapiro. Fast, on-line learning of globally consistent maps. *Autonomous Robots*, 12(3):287–300, 2002.

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

[37] B. Englot and F. S. Hover. Three-dimensional coverage planning for an underwater inspection robot. *International Journal of Robotics Research*, 32(9–10):1048–1073, 2013.

[38] T. Erez and W. D. Smart. A scalable method for solving high-dimensional continuous POMDPs using local approximation. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*, pages 160–167, Catalina Island, CA, USA, July 2010.

[39] R. Eustice, O. Pizarro, H. Singh, and J. Howland. UWIT: Underwater image toolbox for optical image processing and mosaicking in Matlab. In *Proceedings of the International Symposium on Underwater Technology*, pages 141–145, Tokyo, Japan, Apr. 2002.

[40] 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.

[41] 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.

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

[43] H. J. S. Feder, J. J. Leonard, and C. M. Smith. Adaptive mobile robot navigation and mapping. *International Journal of Robotics Research*, 18(7):650–668, 1999.

[44] 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.

[45] 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, Apr. 2004.

[46] J. C. Francis and D. Kim. *Modern Portfolio Theory: Foundations, Analysis, and New Developments*. John Wiley & Sons, Hoboken, NJ, 2013.

[47] E. Galceran and M. Carreras. Planning coverage paths on bathymetric maps for in-detail inspection of the ocean floor. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4159–4164, Karlsruhe, Germany, May 2013.

[48] E. Galceran, S. Nagappa, M. Carreras, P. Ridao, and A. Palomer. Uncertainty-driven survey path planning for bathymetric mapping. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 6006–6012, Tokyo, Japan, Nov. 2013.

[49] H. H. Gonzalez-Banos and J.-C. Latombe. Navigation strategies for exploring indoor environments. *International Journal of Robotics Research*, 21(10–11):829–848, 2002.

[50] R. L. Gordon. Acoustic doppler current profiler: Principles of operation, a practical primer. Technical report, RD Instruments, San Diego, CA, USA, Jan. 1996.

[51] D. J. Hand. *Statistics: A Very Short Introduction*. Oxford University Press, Oxford, 2008.

[52] R. M. Haralick. Propagating covariance in computer vision. In *Proceedings of the International Conference on Pattern Recognition*, volume 1, pages 493–498, Jerusalem, Israel, Oct. 1994.

[53] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2): 100–107, 1968.

[54] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[55] R. He, E. Brunskill, and N. Roy. Efficient planning under uncertainty with macro-actions. *Journal of Artificial Intelligence Research*, 40(1):523–570, 2011.

[56] G. Hollinger and G. Sukhatme. Sampling-based motion planning for robotic information gathering. In *Proceedings of the Robotics: Science & Systems Conference*, Berlin, Germany, June 2013.

[57] G. A. Hollinger and G. S. Sukhatme. Sampling-based robotic information gathering algorithms. *International Journal of Robotics Research*, 33(9):1271–1287, 2014.

[58] G. A. Hollinger, B. Englot, F. S. Hover, U. Mitra, and G. S. Sukhatme. Active planning for underwater inspection and the benefit of adaptivity. *International Journal of Robotics Research*, 32(1):3–18, 2013.

[59] F. S. Hover, J. Vaganay, M. Elkins, S. Willcox, V. Polidoro, J. Morash, R. Damus, and S. Desset. A vehicle system for autonomous relative survey of in-water ships. *Marine Technology Society Journal*, 41(2):44–55, 2007.

[60] 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.

[61] R. A. Howard and J. E. Matheson. Risk-sensitive markov decision processes. *Management Science*, 18(7):356–369, 1972.

[62] G. Huang, M. Kaess, and J. J. Leonard. Consistent sparsification for graph optimization. In *Proceedings of the European Conference on Mobile Robots*, pages 150–157, Barcelona, Spain, Sept. 2013.

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

[64] V. Ila, L. Polok, M. Solony, P. Smrz, and P. Zemcik. Fast covariance recovery in incremental nonlinear least square solvers. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 4636–4643, Seattle, WA, USA, May 2015.

[65] V. Indelman. Towards information-theoretic decision making in a conservative information space. In *Proceedings of the American Control Conference*, pages 2420–2426, Chicago, IL, USA, July 2015.

[66] V. Indelman. No correlations involved: Decision making under uncertainty in a conservative sparse information space. *IEEE Robotics and Automation Letters*, 1(1): 407–414, 2016.

[67] V. Indelman, L. Carlone, and F. Dellaert. Planning in the continuous domain: A generalized belief space approach for autonomous navigation in unknown environments. *International Journal of Robotics Research*, 34(7):849–882, 2015.

[68] L. Jaillet, J. Cortes, and T. Simeon. Sampling-based path planning on configuration-space costmaps. *IEEE Transactions on Robotics*, 26(4):635–646, 2010.

[69] M. V. Jakuba, C. N. Roman, H. Singh, C. Murphy, C. Kunz, C. Willis, T. Sato, and R. A. Sohn. Field report: Long-baseline acoustic navigation for under-ice AUV operations. *Journal of Field Robotics*, 25(11–12):861–879, 2008.

[70] 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.

[71] D. Johnstone and D. Lindley. Mean-variance and expected utility: The borch paradox. *Statistical Science*, 28(2):223–237, 2013.

[72] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2):99–134, 1998.

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

[74] 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, 2012.

[75] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research*, 30(7):846–894, 2011.

[76] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[77] A. Kim and R. M. Eustice. Pose-graph visual SLAM with geometric model selection for autonomous underwater ship hull inspection. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1559–1565, St. Louis, MO, USA, Oct. 2009.

[78] A. Kim and R. M. Eustice. Combined visually and geometrically informative link hypothesis for pose-graph visual SLAM using bag-of-words. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1647–1654, San Francisco, CA, USA, Sept. 2011.

[79] 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.

[80] A. Kim and R. M. Eustice. Active visual SLAM for robotic area coverage: Theory and experiment. *International Journal of Robotics Research*, 34(4–5):457–475, 2015.

[81] J. C. Kinsey, R. M. Eustice, and L. L. Whitcomb. Underwater vehicle navigation: Recent advances and new challenges. In *IFAC Conference on Manoeuvring and Control of Marine Craft*, Lisbon, Portugal, Sept. 2006.

[82] 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.

[83] H. Kretzschmar and C. Stachniss. Information-theoretic compression of pose-graphs for laser-based SLAM. *International Journal of Robotics Research*, 31(11):1219–1230, 2012.

[84] H. Kurniawati, D. Hsu, and W. S. Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Proceedings of the Robotics: Science & Systems Conference*, pages 65–72, Zurich, Switzerland, June 2008.

[85] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006.

[86] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, 2001.

[87] J. J. Leonard and H. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1442–1447, Osaka, Japan, Nov. 1991.

[88] C. Leung, S. Huang, N. Kwok, and G. Dissanayake. Planning under uncertainty using model predictive control for information gathering. *Robotics and Autonomous Systems*, 54(11):898–910, 2006.

[89] J. Li and M. Johnson-Roberson. Multi-altitude multi-sensor fusion framework for AUV exploration and survey. In *Proceedings of the IEEE/MTS OCEANS Conference and Exhibition*, pages 1–8, St. John's, NL, Canada, Sept. 2014.

[90] T. J. Linsmeier and N. D. Pearson. Value at risk. *Financial Analysts Journal*, 56(2): 47–67, 2000.

[91] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

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

[93] S. I. Marcus, E. Fernandez-Gaucherand, D. Hernandez-Hernandez, S. Coraluppi, and P. Fard. Risk sensitive Markov decision processes. *Systems and Control in the Twenty-First Century*, 22:263–281, 1997.

[94] H. Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952.

[95] A. Melkumyan and F. Ramos. A sparse covariance function for exact Gaussian process inference in large datasets. In *Proceedings of the International Joint Conference on Artifical Intelligence*, pages 1936–1942, Pasadena, CA, USA, July 2009.

[96] R. C. Merton. *Continuous-Time Finance*. Wiley-Blackwell, Cambridge, MA, 1992.

[97] T. M. Moldovan and P. Abbeel. Risk aversion in Markov decision processes via near-optimal Chernoff bounds. In *Proceedings of the Advances in Neural Information Processing Systems Conference*, pages 3140–3148, Lake Tahoe, NV, USA, Dec. 2012.

[98] S. T. O'Callaghan and F. T. Ramos. Gaussian process occupancy maps. *International Journal of Robotics Research*, 31(1):42–62, 2012.

[99] E. Olson. AprilTag: A robust and flexible visual fiducial system. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3400–3407, Shanghai, China, May 2011.

[100] P. Ozog. *Advances in simultaneous localization and mapping in confined underwater environments using sonar and optical imaging*. PhD thesis, Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI, USA, Jan. 2016.

[101] P. Ozog and R. M. Eustice. Identifying structural anomalies in image reconstructions of underwater ship hulls. In *Proceedings of the IEEE/MTS OCEANS Conference and Exhibition*, pages 1–7, Washington, D.C., USA, Oct. 2015.

[102] 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, Special Issue on Safety, Security and Rescue Robotics*, 33(3):265–289, 2016.

[103] C. H. Papadimitriou and J. N. Tsitsiklis. The complexity of Markov decision processes. *Mathematics of Operation Research*, 12(3):441–450, 1987.

[104] S. Patil, Y. Duan, J. Schulman, K. Goldberg, and P. Abbeel. Gaussian belief space planning with discontinuities in sensing domains. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 6483–6490, Hong Kong, China, May 2014.

[105] S. Patil, G. Kahn, M. Laskey, J. Schulman, K. Goldberg, and P. Abbeel. Scaling up Gaussian belief space planning through covariance-free trajectory optimization and automatic differentiation. In *Proceedings of the International Workshop on the Algorithmic Foundations of Robotics*, Istanbul, Turkey, Aug. 2014.

[106] R. Platt, R. Tedrake, L. Kaelbling, and T. Lozano-Perez. Belief space planning assuming maximum likelihood observations. In *Proceedings of the Robotics: Science & Systems Conference*, pages 587–593, Zaragoza, Spain, June 2010.

[107] 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*, Berlin, Germany, June 2013.

[108] J. M. Porta, N. Vlassis, M. T. J. Spaan, and P. Poupart. Point-based value iteration for continuous POMDPs. *Journal of Machine Learning Research*, 7:2329–2367, 2006.

[109] S. Prentice and N. Roy. The belief roadmap: Efficient planning in belief space by factoring the covariance. *International Journal of Robotics Research*, 28(11–12):1448–1465, 2009.

[110] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge, MA, 2006.

[111] R. T. Rockafellar and S. Uryasev. Optimization of conditional value-at-risk. *Journal of Risk*, 2:21–42, 2000.

[112] B. Seck, L. Andrieu, and M. D. Lara. Parametric multi-attribute utility functions for optimal profit under risk constraints. *Theory and Decision*, 72(2):257–271, 2012.

[113] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza. *Introduction to Autonomous Mobile Robots*. MIT Press, Cambridge, MA, USA, 2nd edition, 2011.

[114] R. Sim and N. Roy. Global A-optimal robot exploration in SLAM. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 661–666, Barcelona, Spain, Apr. 2005.

[115] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. I. Jordan, and S. S. Sastry. Kalman filtering with intermittent observations. *IEEE Transactions on Automatic Control*, 49(9):1453–1464, 2004.

[116] 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.

[117] C. Stachniss, G. Grisetti, and W. Burgard. Information gain-based exploration using Rao-Blackwellized particle filters. In *Proceedings of the Robotics: Science & Systems Conference*, Cambridge, MA, USA, June 2005.

[118] S. Thrun. Monte carlo POMDPs. In S. Solla, T. Leen, and K.-R. Müller, editors, *Proceedings of the Advances in Neural Information Processing Systems Conference*, pages 1064–1070. MIT Press, 2000.

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

[120] S. Thrun, Y. Liu, D. Koller, A. Y. 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.

[121] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. The MIT Press, Cambridge, MA, 2006.

[122] R. Valencia, J. Miro, G. Dissanayake, and J. Andrade-Cetto. Active pose SLAM. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1885–1891, Vilamoura, Portugal, Oct. 2012.

[123] R. Valencia, M. Morta, J. Andrade-Cetto, and J. Porta. Planning reliable paths with pose SLAM. *IEEE Transactions on Robotics*, 29(4):1050–1059, 2013.

[124] J. van den Berg, P. Abbeel, and K. Goldberg. LQG-MP: Optimized path planning for robots with motion uncertainty and imperfect state information. *International Journal of Robotics Research*, 30(7):895–913, 2011.

[125] J. van den Berg, S. Patil, and R. Alterovitz. Motion planning under uncertainty using differential dynamic programming in belief space. In *Proceedings of the International Symposium on Robotics Research*, Flagstaff, AZ, USA, Aug. 2011.

[126] J. van den Berg, S. Patil, and R. Alterovitz. Motion planning under uncertainty using iterative local optimization in belief space. *International Journal of Robotics Research*, 31(11):1263–1278, 2012.

[127] S. Vasudevan, F. Ramos, E. Nettleton, H. Durrant-Whyte, and A. Blair. Gaussian process modeling of large scale terrain. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1047–1053, Kobe, Japan, May 2009.

[128] 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.

[129] K. Vickery. Acoustic positioning systems: A practical overview of current systems. In *Proceedings of the IEEE/OES Autonomous Underwater Vehicles Conference*, pages 5–17, Cambridge, MA, USA, Aug. 1998.

[130] M. P. Vitus and C. J. Tomlin. Sensor placement for improved robotic navigation. In *Proceedings of the Robotics: Science & Systems Conference*, pages 217–224, Zaragoza, Spain, June 2010.

[131] M. P. Vitus and C. J. Tomlin. Closed-loop belief space planning for linear, Gaussian systems. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2152–2159, Shanghai, China, May 2011.

[132] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, Princeton, NJ, USA, 2nd edition, 1947.

[133] 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):1191–1208, 2014.

[134] J. M. Walls, S. M. Chaves, E. Galceran, and R. M. Eustice. Belief space planning for underwater cooperative localization. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2264–2271, Hamburg, Germany, Sept. 2015.

[135] P. Whaite and F. Ferrie. Autonomous exploration: Driven by uncertainty. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3):193–205, 1997.

[136] L. Whitcomb, M. Jakuba, J. Kinsey, S. Martin, S. Webster, J. Howland, C. Taylor, D. Gomez-Ibanez, and D. Yoerger. Navigation and control of the nereus hybrid underunder vehicle for global ocean science to 10,903 m depth: Preliminary results. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 594–600, Anchorage, AK, USA, May 2010.

[137] S. Williams and I. Mahon. Simultaneous localisation and mapping on the great barrier reef. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1771–1776, New Orleans, LA, USA, Apr. 2004.

[138] 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.

[139] R. W. Wolcott and R. M. Eustice. Fast LIDAR localization using multiresolution gaussian mixture maps. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2814–2821, Seattle, WA, USA, May 2015.