MULTI-SLAM SYSTEMS FOR FAULT-TOLERANT SIMULTANEOUS LOCALIZATION AND MAPPING

A Dissertation Presented

by

SAMER B. NASHED

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February 2024

Manning College of Information and Computer Sciences

MULTI-SLAM SYSTEMS FOR FAULT-TOLERANT SIMULTANEOUS LOCALIZATION AND MAPPING

A Dissertation Presented

by

SAMER B. NASHED

Approved as to style and content by:

DEDICATION

To all my friends, family, peers, and professors

who have encouraged my love of science and engineering

ACKNOWLEDGMENTS

Although no path to a PhD is without adversity, I consider myself to have been exceedingly lucky when it comes to the people with whom I have been able to share this journey. First and foremost, I am incredibly grateful to my family, especially my parents and my wife Su Lin, who have supported me so steadfastly throughout this process.

I have also been able to create a family of sorts here at UMass, from some of the great office and custodial staff, especially Leeanne Leclerc and Rob, to my friends, many of them also housemates or labmates, including Dirk Ruiken, Kyle Wray, Mitch Hubert, Shanu Vashishtha, Ameya Gadbole, Zeal Shah, Jay Wong, Jarrett Holtz, Sadegh Rabiee, Alyx Burns, Kyle Vedder, Tiffany Liu, Khoshrav Doctor, Scott Jordan, Takeshi Takahashi, Mike Lanigan, Shuwa Miura, Moumita Choudhury, Kevin Winner, and Myungha Jang. You have made my time outside the lab so much more meaningful and memorable.

The ideas, techniques, and much of the behind-the-scenes work for the projects in this thesis would not have happened without valuable input and late nights from many of my collaborators, both at UMass—Justin Svegliato, Matteo ”Bruaco” Brucato, Connor Basich, Abhinav Bhatia, Saad Mahmud, and Mason Nakamura—and elsewhere—Dave Ilstrup, Roger Webster, Joey Durham, Claudia Goldman. Our work together has taught me many valuable lessons and influenced my thinking, and it is easy to enjoy research with such wonderful collaborators.

It has also been a great pleasure to have such a friendly, supportive, and invested thesis committee. Meghan Huber, Donghyun Kim, and Jong Jin Park, you have provided some of the most interesting, inspiring, and exciting perspective on the work in this thesis, and it is profoundly better for it.

Most importantly, I would like to thank my advisors, Rod Grupen and Shlomo Zilberstein. Rod, your generosity, grace, and kindness when I was without a lab will always be deeply appreciated, and without your support I would have certainly never finished. It has been a sincere pleasure to learn from you. And Shlomo, I would like to you for your patience, guidance, enthusiasm, and willingness to include me in your lab as I finished the PhD program. You have both been important role models for me not only in research, but also as mentors, managers, and leaders. I will always be grateful for my time in the LPR and RBR Laboratory.

ABSTRACT

MULTI-SLAM SYSTEMS FOR FAULT-TOLERANT SIMULTANEOUS LOCALIZATION AND MAPPING

FEBRUARY 2024

SAMER B. NASHED

B.A., SWARTHMORE COLLEGE

M.Sc., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Roderic Grupen and Professor Shlomo Zilberstein

Mobile robots need accurate, high fidelity models of their operating environments in order to complete their tasks safely and efficiently. Generating these models is most often done via Simultaneous Localization and Mapping (SLAM), a paradigm where the robot alternatively estimates the most up-to-date model of the environment and its position relative to this model as it acquires new information from its sensors over time. Because robots operate in many different environments with different compute, memory, sensing, and form constraints, the nature and quality of information available to individual instances of different SLAM systems varies substantially. ‘One-size-fits-all’ solutions are thus exceedingly difficult to engineer, and highly specialized systems, which represent the state-of-the-art for most types of deployments, are not robust to operating conditions in which their assumptions are not met. This thesis seeks to investigate an alternative approach to these robustness and universality problems by incorporating existing SLAM solutions within a larger framework supported by planning and learning. The central idea is to combine learned models that estimate SLAM algorithm performance under a variety of sensory conditions, in this case neural networks, with planners designed for planning under uncertainty and partial observability, in this case partially observable Markov decision problems (POMDPs). Models of existing SLAM algorithms can be learned, and these models can then be used online to estimate the performance of a range of solutions to the SLAM problem at hand. The POMDP policy then selects the appropriate algorithm, given the estimated performance, cost of switching methods, and other information. This general approach may also be applicable to many other robotics problems that rely on data-fusion, such as grasp planning, motion planning, or object identification.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Robots spark our imagination and curiosity, while also providing the potential to help people with significant challenges in their daily lives. Robots could handle many hazardous or monotonous jobs or be employed in jobs for which there is a shortage of dedicated human workers, such as eldercare. Many industries are already using robots or preparing to augment their work forces with robots, including agriculture, transportation, security, rescue, resource extraction such as mining and logging, space exploration, maintenance and custodial services, and construction. Although different on the surface, these tasks require robots to possess a common set of capabilities. For mobile robots — robots that can move around using wheels, legs, wings, fins, rotors, or other forms of locomotion — one such essential ability is the creation of models of the robot's operating environment that support safe and efficient navigation. A corollary to this requirement is that robots must be able to localize relative to this model. We call these two problems *mapping* and *localization*, respectively, and they are often executed simultaneously as a robot explores a new environment or updates an existing model. This process is known by the acronym SLAM, for Simultaneous Localization And Mapping. Although state-of-the-art SLAM systems have evolved considerably since their inception in the 1980s, new approaches to situated decision making like SLAM remain one of the most challenging obstacles to long-term deployment of robust mobile robotic systems.

In general, there are three primary challenges SLAM systems must overcome. The first is computational complexity. Computing the exact maximum likelihood estimate for all of a robot's prior locations (known as trajectory estimation) is $O(n^3)$ since it involves a matrix inversion. Here, n represents the number of points in time for which we want location estimates and is typically in the thousands for deployments longer than a few minutes. Approximate solutions are, therefore, the state-of-the-art and approximation always involves marginalizing some variables – the choice of which describes the variation in approaches.

The second is robustness to sensor noise. Sensors, actuators, and the robot's prior beliefs about the world are all imperfect. Signals from modern sensors often contain random noise, perturbations, or corruption with respect to the ground truth signal. These defects in the signal can be caused by small imperfections in the hardware, by quantization error as analog signals are converted to digital ones, or by the internal model of the sensor which does not perfectly describe the physics of the interaction between the sensor and the environment it is sensing. Sometimes this noise follows a known distribution, but often it deviates in extreme, unpredictable ways, which can make estimating the probability of sensing a particular value difficult. Imperfect models and noisy sensing, combined with the inability to accurately model even the distribution from which a noisy signal may be drawn, make higher-level inference problems such as robot localization extremely challenging.

The third is partial observability. Sensor information is typically incomplete. The field of view and resolution of cameras and depth sensors is limited and important features of an environment may be out of view due to these limitations or due to environmental dynamics such as occlusion, lighting, or weather. Moreover, data from sensors is temporally constrained to represent a single moment in time, whereas the models built by SLAM must represent the world at any time. Learning and applying higher level concepts to form prior beliefs or resolve ambiguities is one of the most widely adopted methods for addressing partial observability. Many potentially useful priors about the world — for example, the vast majority of buildings have a continuous, closed outer wall — are difficult to represent explicitly and are challenging to learn. This limitation also affects lower-level processes. For instance, a robot may view an open doorway, leave the area, and then later return to view the same doorway with the door closed. The change in appearance may be substantial depending on the sensor, which can cause the robot to calculate a low probability that the two sensor readings correspond to the same physical location. Humans may use knowledge about how doorways operate and the fact that they may exist in a range of closed and open configurations to help us solve this problem. They may also use additional information available in the scene or as prior knowledge to help reduce the set of reasonable hypotheses. Robots, however, often lack this kind of background knowledge in the form of prior beliefs about the dynamics of their operating environment and the objects within it, since these concepts are challenging to define mathematically.

Throughout this thesis, a distinction will be made between SLAM *algorithms*, which are the individual building blocks represented in Figure 1.1 b), and SLAM *systems*, which are collections of individual algorithms, run in parallel or in sequence, that together allow state estimation, represented by Figure 1.1 b) in its entirety. Within a SLAM system there may be many SLAM algorithms that perform different functions. Modern SLAM research almost always addresses one or more of the above fundamental limitations (complexity, robustness, partial observability), and typically focuses on improving a specific component (algorithm) of an integrated SLAM system that may be swapped in or out in place of other algorithms. Figures 1.1 a) and 1.1 b) contextualize the role of SLAM systems within a typical mobile robot's operating stack and expand on the notion of SLAM systems as compositions of many SLAM algorithms, respectively. These sub-systems are often highly specialized to a robot's particular intended operating environment, sensor payload, and computation and memory constraints, and the choice of which algorithms to use is made pre-deployment. However, slight perturbations along these axes can cause catastrophic failure and indeed predicting such failures is a challenging open research question. Thus, even as research on each sub-system incrementally relaxes some of the context specificity, it is clear that universally applicable algorithms, even for narrowly focused sub-systems, are not on the horizon owing to the diversity of potential operating environments and the challenges posed therein.

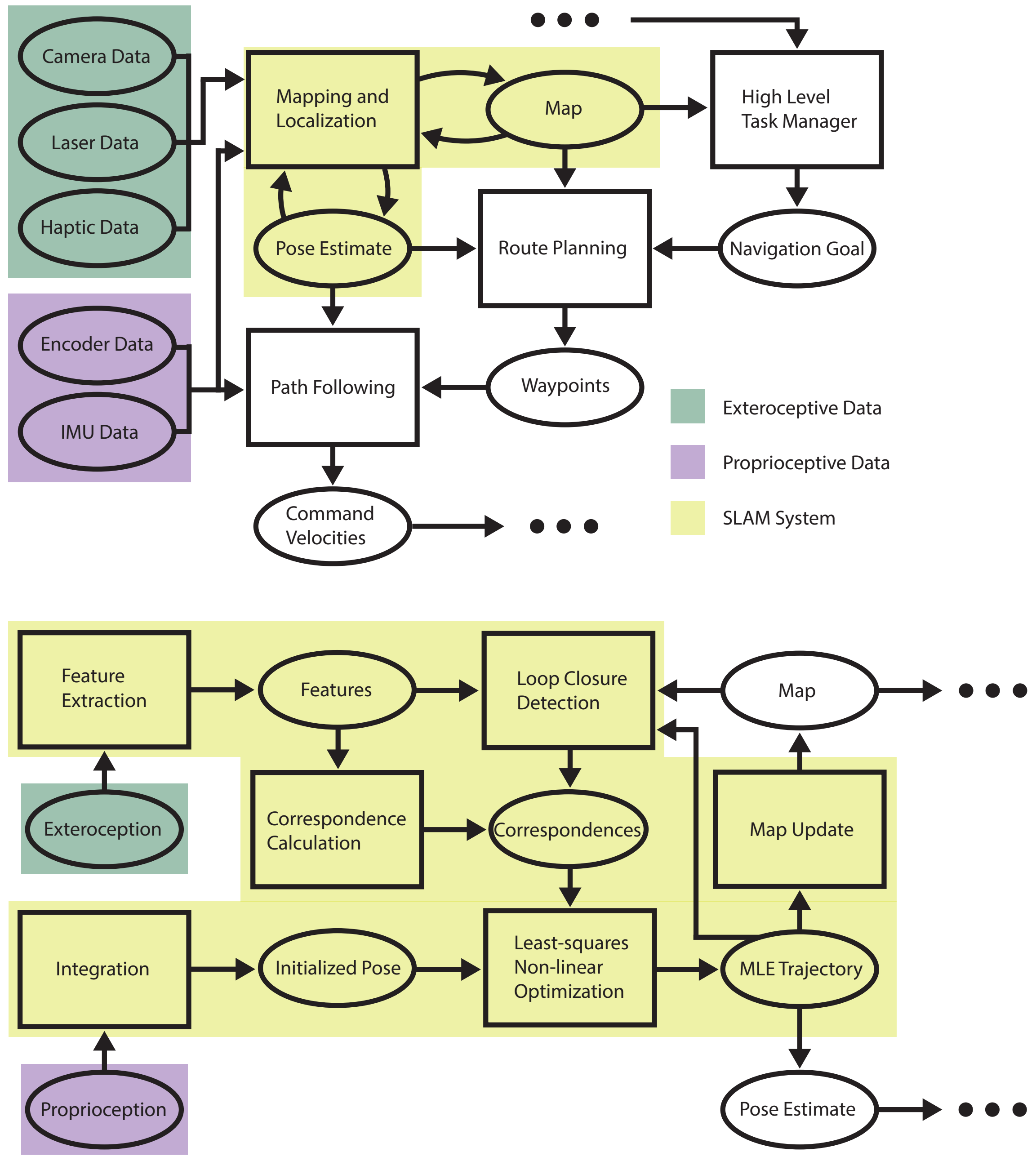


Figure 1.1: Top: A diagram of a simplified mobile robotics architecture and the role of the SLAM system within that architecture. Bottom: A detailed diagram of a representative SLAM system, illustrating how different sub-systems are combined to generate pose estimates and maps. Exteroceptive data is generated from the robot sensing the environment, while proprioceptive data is generated by the robot sensing itself. The color coding across both figures is consistent. Ellipses represent further processing of data or components of the architecture not depicted.

Specialization is not strictly a negative phenomenon, as performance improvements are often won through specialization where additional assumptions about the quality and availability of data and the nature of the operating environment are made. Many specialized SLAM algorithms have begun to work reliably in limited domains and while these domains individually represent a tiny fraction of all contexts where we want robot competence, collectively they cover a significant space of possible scenarios. A SLAM system that operates reliably in all such scenarios could be considered ‘universal’. **I hypothesize that choosing *online* from a portfolio of SLAM algorithms is a realistic alternative to developing a single, universal SLAM system.** This leads naturally to several more specific questions that are the focus of this thesis.

1. Can we accurately predict the performance of a specific SLAM algorithm given information about current sensor data and the robot’s internal model of the environment?

2. Could we use these predictions to decide which SLAM algorithm would produce the most accurate localization estimate at a particular time given that there are costs to switching methods?

3. How do we manage multiple SLAM algorithms simultaneously from both a data perspective and an inference perspective? Are there benefits to long-term robot learning or data-labeling that this system confers?

4. Does such a ‘Multi-SLAM’ system extend the capabilities of robots in terms of the types of deployments that may be reliably executed without localization failure?

As a high-level example to motivate the spirit of this proposal, consider the following scenario. A professor is carrying their large dog down a set of stairs in their home. When the lights are on or there is enough sunlight, the environment affords the professor accurate estimates for several quantities through their sense of vision. These include the position of their feet relative to the stairs, their whole body position relative to the top and bottom of the stairs, and an estimator of their acceleration (augmenting the cochlear fluid in their inner ear). These data streams afford accurate enough localization for the professor to navigate the stairs safely and efficiently. However, given the same task in the same environment but without the benefit of a well-lit staircase, the professor needs to employ a *fundamentally different algorithm* that involves heavy reliance on haptic feedback from their limbs as they slide their feet to sense the discontinuities between stairs. In low light, they may use some visual feedback, but they may interpret the images they receive in a completely different manner. In summary, the task is the same and the agent (professor) is the same, but the environment has changed such that certain localization affordances are differentially present across deployments. These differences are substantial enough to require different behavior (a different algorithm) in order to complete the task safely and efficiently.

The design and implementation of Multi-SLAM systems advanced in this thesis is just one possible instance of such a system, and leaves many design choices open for future experimentation. However, the distinction between a mode-switching system like the one proposed and a more continuous version where different algorithms are always active but are up-or down-weighted, both of which could be considered Multi-SLAM systems, is rather minor compared to the difference between SLAM systems that provide the ability to adapt to sensing conditions online and those that do not. There are, however, several compelling reasons why mode-switching is likely to be the most performant option for Multi-SLAM systems in the long run. First, this setup allows existing code to be used with the least modification. Second, it simplifies the determination of relative data quality to the ordinal domain rather than the numerical domain. Third, predictive models may be trained in isolation since they are not coupled to other front-end algorithms. Fourth, development of specialized SLAM systems can continue in parallel without requiring any extra experimentation or development prior to use in a Multi-SLAM system.

During the course of the work presented in the earlier chapters, it became very clear that when SLAM systems succeed, they do so in large part due to specialized assumptions about their operating conditions, and thus are very likely to fail when these conditions are not met. Moreover, there is incredible variety in the SLAM literature, matching the variety of operating conditions that may be present in the large number of aspirational mobile robotics applications. Together, these two insights have been the primary motivators for the development and study of Multi-SLAM systems and their constituent parts.

1.1 Contributions

This thesis presents several contributions to the SLAM literature, which I break down into two groups. The first group (Chapters 3-6) focuses on SLAM *algorithms* for robust perception that aim to mitigate the effects of imperfect information and noisy sensor readings by constructing mathematical models that are more robust to outliers (Chapters 3 and 4), are more efficient at using partial information (Chapters 4 and 6), and can incorporate top-down guidance in the form of priors or constraints for more accurate and efficient inference (Chapters 5 and 6). Specifically, this group makes the following contributions:

- **Curating Long-Term Vector Maps:** Given a reconstruction of an environment, long-term vector mapping filters out observations from movable entities and curates a maximum likelihood map of line segments extracted from the remaining observations, essentially creating a blue print of the environment. This method has a small memory footprint that scales with the number of features in the environment rather than the area explored or duration of the deployment. (Published in IROS 2016)
- **Localization under Topological Uncertainty for Lane Identification of Autonomous Vehicles:** This algorithm uses observations of the environment (lane markings) and observations of other vehicles on the road to estimate the lane membership of an autonomous vehicle. This research also does not assume the given map is perfect, and deals with uncertainty in the topology of the map by employing a

population of models to reason about whether its current observations contradict the topological structure given by the map. (Published in ICRA 2018)

- **Human-in-the-Loop SLAM:** This research proposes an algorithm for correcting outputs of SLAM systems using minimal human input. SLAM algorithms often produce imperfect results, and even non-experts can identify errors in the resultant maps. This system takes a small amount of human input, creates new, potentially rank-deficient constraints, adds them to the original optimization problem, and re-solves the problem to generate a more accurate map without the need to re-collect data or increase the solver's compute budget. (Published in AAAI 2018)

- **Robust Rank-Deficient SLAM:** Many features that are easily and reliably detected do not fully constrain a robot's relative motion from frame to frame. This research improves the state-of-the-art in such systems in both 2D and 3D by proposing a number of improvements including correspondence matching techniques, rank-deficient constraint detection and formulation within an optimization framework, and efficient storage and manipulation of map items online. (Published in IROS 2021)

The second group (Chapters 7-9), motivated by insights won during the development of the systems presented in the preceding chapters, introduces predictive components (Chapter 7) and planning components (Chapter 8) of 'Multi-SLAM' systems (Chapter 9), which represent a novel and effective strategy for robust SLAM. These chapters and the design of Multi-SLAM systems as a whole draw on several concepts explored in other contexts in the preceding chapters, though the technical details often differ. These include filtering visual stimuli in order to reject or exclude them from further inference (Chapter 3), managing portfolios of models in order to adaptively apply them to the appropriate inference problem (Chapter 4), modifying and augmenting the graphical model describing pose-graph SLAM (Chapter 5), and understanding how specialization improves performance within a restricted domain (Chapter 6). Specifically, this group makes the following contributions:

- **Learning Performance Models for SLAM Algorithms:** Even as SLAM algorithms become more robust, there are still no practicable theories for predicting when they will produce a poor result. This research develops a learning framework for predicting SLAM system performance using deep neural networks. We show certain convolutional neural network architectures that reliably learn useful predictive performance models of SLAM systems operating using different modalities. (In preparation)
- **A Belief-Space Planner for Adaptive Fault Tolerant SLAM:** This research formulates and solves a novel decision-making problem that captures the relevant trade offs when choosing between SLAM algorithms online. We model the problem as a partially observable Markov decision process and implement a belief-space planner over SLAM algorithms that operates on noisy estimates of component effectiveness and reliability. We demonstrate in simulation that this system is more robust to changes in sensing conditions than singular SLAM systems. (Submitted ICRA 2024)
- **An Integrated Architecture for Multi-SLAM Systems:** While trajectory estimate error may be reduced via Multi-SLAM systems, this technique creates several complex modeling and inference problems. This research describes new variations of dynamic Bayesian networks needed to model the process of switching between SLAM algorithms and new techniques for stitching together maps made using different modalities. This research also offers an overview of some of the support systems required to make Multi-SLAM systems functional in practice. (In preparation)

The contributions in this thesis work towards mobile robotic systems that work reliably in many contexts. They achieve this by increasing robustness of SLAM systems through several different mechanisms, including filtering observations, maintaining multiple models of the world, incorporation of external constraints, and specialization to specific types of data. Multi-SLAM systems are the culmination of these efforts into an adaptable, extensible framework that benefits passively from advances in specialized SLAM algorithms.

CHAPTER 2

RELATED LITERATURE

2.1 What Makes SLAM Challenging?

Questions involving SLAM have been in the crosshairs of researchers for decades, and although modern SLAM systems perform considerably better than their predecessors, there are still significant obstacles to deploying SLAM systems that ‘just work’. These challenges arise primarily from three sources: constraints due to complexity and resource bounded robots; the incredible variety of environments and hardware configurations on which SLAM systems may be deployed; and the need for robust behavior in the presence of noisy data. All of the research in this thesis, and most of the research in the greater SLAM community, is aimed at addressing one or more of these challenges. The goal of this section is to present these core challenges so that the practical benefits of research discussed in the remainder of this thesis are clear.

2.1.1 Computational Challenges

SLAM systems can be broken down into two large components: the front-end and the back-end. The front-end extracts features from one or more sensor streams and matches those features with previous sensor readings to produce an initial estimate of the robot’s motion during the time elapsed between readings. It also attempts to detect when the robot revisits a location by matching the current sensor readings to previous sensor readings. These procedures can be expensive, especially for high-resolution sensors, but their compute time is bounded in practice by taking advantage of space-partitioning data structures and parallelization. The most significant computational bottleneck usually occurs in the back-end.

The back-end is responsible for computing the maximum likelihood estimate (MLE) or maximum a posteriori (MAP) of the pose of the robot at a series of times $(t_0, \ldots, t_n)$. This series grows without bound as the robot operates. In many cases the number of times we want to estimate robot state, n, grows by one every time an image is captured or a laser scans. Typical rates are between 1Hz and 40Hz. Computing the exact solution to these state estimation problems over the complete history of the robot's deployment, for a robot with d position and orientation variables, is $O((dn)^3)$ since it involves a matrix inversion. Moreover, to maintain the most accurate estimate, this problem must be re-solved every time there is new information from a sensor. Approximate solutions are therefore required in most cases, and these approximations either reduce the number of variables (reduce n) or use a method of inference with better asymptotic complexity. We will discuss various methods for localization in Section 2.2.

Memory constraints do not typically affect state estimation, but they can affect how maps are stored and updated. Depending on the representation, some maps may require more memory in order represent the environment at a higher fidelity. Maps naturally grow over time as the robot explores its environment and models new entities, and there are many open questions as to the most compact, informative, and easily updated representations of the world. Moreover, different representations may encode different types of information and support different methods of localization. We will discuss various choices for representing the environment in Section 2.3.

2.1.2 'Curse of Ubiquity' Challenges

The fact that every mobile robot needs some version of a SLAM system to operate effectively means that the problem has received significant attention, and many aspects of the problem are now well-understood thanks to previous research. However, this also means that any *general* SLAM solution must address the unique challenges of every combination of robot, sensor suite, and environment. This combinatorial increase in domains of consideration explodes a single state estimation problem into hundreds or thousands of related, but distinct, problems. This problem diversity can be ignored in some applications where operating conditions can be controlled. For example, robots on the floors of Amazon warehouses operate in a relatively tightly controlled environment where assumptions about the nature and availability of different visual features are relatively easy to meet. However, in other applications, such as robots assisting in search and rescue missions after natural or industrial disasters, different operating environments may vary substantially.

Moreover, each sensor type has unique characteristics, including under what conditions it provides useful signals. Even the choice of sensor position on a robot can affect the applicability of different techniques. The number of possible sensors on modern robots is large, including lasers, sonar, gyroscopes, accelerometers, joint encoders, mono-cameras, stereo cameras, depth cameras, GPS, infrared sensors, touch and torque sensors, wifi antennae, and other, specialized hardware. Thus, the combination of possible sensors and environments for which some type of SLAM algorithm could be implemented is truly staggering, and contemporary research faces significant challenges in unifying such algorithms.

Although some applications permit specialization, SLAM researchers are also interested in the basic science of developing ever more general SLAM algorithms for several reasons. More powerful and adaptable SLAM systems not only take us closer to more general purpose robots, but they also provide increased capability and flexibility to existing robots in the form of robustness to catastrophic failures, such as the loss of a sensor, that effectively change the set of information available to the robot. General SLAM systems are also of interest for their ability to help us understand and potentially model some of the cognitive processes humans and other animals use to form models of their world.

2.1.3 Robustness Challenges

All SLAM algorithms, regardless of the sensors they use or the environment in which they are deployed, must also deal with challenges posed by noisy data. By noisy data, we mean that measurements about the state of the world generated by the robot's sensors generally contain some error. For instance, if the true distance from a robot to a wall is 1.0 meter, a sensor which measures this distance will often report values near 1.0 meter, such as 0.95 meters or 1.05 meters, but rarely do they report the correct value (1.0 meter) exactly. The same problem occurs, for different physical reasons, in virtually all sensing modalities. If the aggregate behavior of sensor readings can be accurately described by starting with the true values and adding noise terms drawn from a known distribution, this problem becomes easily solvable. Unfortunately, this is generally not true for robotic sensors. In most cases, not only is sensor data noisy, but the noise term is also drawn from a partially unknown distribution. We say partially unknown because under some conditions we can characterize the distribution of sensor noise fairly accurately. However, there are also many conditions in which we cannot. The need to deal with potentially large and unpredictable errors in sensor data has motivated a substantial subset of SLAM research.

SLAM systems also face higher accuracy requirements than many other estimation problems. Since SLAM systems estimate robot location as frequently as 20-30Hz, accuracy within a safe tolerance must be well above 99.99% in order to achieve a reasonable mean time between failures. Moreover, there are significant cascading effects produced by poor location estimates, such as incorrect maps, infeasible plans, and instantiation of bad prior beliefs for subsequent tasks, including future localization. All of these failure cases can have significant consequences for safety and efficacy. Combined, the relatively strict requirements for performance and our inability to universally describe sensor noise using simple distributions create significant, unique challenges for SLAM research.

2.2 Methods for State Estimation (Localization)

2.2.1 Mathematical Foundations

For the remainder of this thesis, bold script denotes vectors (e.g. **x**), capital bold script denotes matrices (e.g. **X**), lower case italics denotes scalars (e.g. x), and capital italics denotes sets (e.g. X). Notation deviating from these conventions will be explained locally.

All SLAM systems estimate the current positional state of a robot. In addition, they may also estimate past positional states or the location of various entities in their environment. We call these positional states *poses*. In 2D applications, such as a robot rolling along an office floor, we denote the pose of the robot at time t as $\mathbf{x}_t = [\begin{array}{ccc} x_t & y_t & \theta_t \end{array}]^T$, where x_t and y_t represent the location of the robot along some x- and y-axes at time t, and θ_t represents the robot's orientation, or yaw, at time t. For robots operating in 3D, such as those that fly or dive, we define the pose as $\mathbf{x}_t = [\begin{array}{cccccc} x_t & y_t & z_t & \phi_t & \theta_t & \psi_t \end{array}]^T$, where x_t, y_t, and z_t, denote the robot's latitudinal, longitudinal, and vertical location, and ϕ_t, θ_t, and ψ_t, denote its roll, pitch, and yaw, respectively.

Although robots may have many configurable joints, for the purpose of localization we can usually consider them abstractly as rigid bodies with coordinate systems fixed to some reference point. A robot's position in its environment can be defined by the rotation and translation of its own coordinate system relative to some global coordinate system. In Euclidean space, the set of all finite linear combinations of translations and rotations form algebraic groups. Specifically, they form the special Euclidean groups $SE(2)$ (2D) and $SE(3)$ (3D), which are both topological groups as well as Lie groups. $SE(2)$ and $SE(3)$ are also subgroups of the group of affine transformations. For rigid bodies, transformations can always be described by combinations of rotation and translation. These two operations themselves form groups. The special orthogonal groups for rotations, $SO(2)$ (2D) and $SO(3)$ (3D), and the translation group for translations.

Using homogeneous coordinates, rotations and translations can be combined into a single matrix operation. This operation represents the transformation from one robot pose or other rigid body, denoted by the vector $\mathbf{x}$, to a new pose $\mathbf{x}'$. We will use $\mathbf{R}$ to denote rotation matrices and $\mathbf{T}$ to denote translation vectors. Since the special Euclidean groups are subsets of the group of affine transformations, we will use $\mathbf{A}$ for transformation matrices which combine both operations. Thus, we can express the process of existing at pose $\mathbf{x}$, undergoing some translation $\mathbf{T}$ and rotation $\mathbf{R}$, and ending up in pose $\mathbf{x}'$, as

$$\mathbf{x}' = \mathbf{R}\mathbf{x} + \mathbf{T} \quad \text{or} \quad \mathbf{x}' = \mathbf{A}\mathbf{x}.$$

Rotations in three dimensions can also be represented more compactly using quaternions [30], and affine transformations using dual quaternions [77], but the general SLAM problem statement and solution techniques remain largely the same in either representation.

Virtually all SLAM algorithms estimate the current most likely pose, $\mathbf{x}_t$. Many also estimate the most likely trajectory of the robot $X = \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_t\}$. That is, the most likely set of all poses since the beginning of the deployment. Estimating only the current pose is often called *filtering*, while estimating the entire trajectory is often referred to as *smoothing*. In general, solving this problem exactly involves solving the following linear system of equations, representing the relative measurements made by the robot between its previous and subsequent positions and between itself and the world, and the relative certainty of these measurements.

$$\mathbf{\Lambda}\mathbf{x} = \mathbf{b}, \quad \mathbf{x} = \mathbf{\Lambda}^{-1}\mathbf{b} \tag{2.1}$$

Here, $\mathbf{x}$ is the vector of variables representing all robot poses $[x_0, y_0, \theta_0, \ldots, x_n, y_n, \theta_n]^T$ we would like to solve for, $\mathbf{\Lambda}$ represents the constraints derived from measurements, and $\mathbf{b}$ represents the estimated uncertainty of each measurement. Thus, if $\mathbf{x}$ is a $1 \times n$ vector, $\mathbf{\Lambda}$ will be an $n \times n$ matrix. We should note that while most of the relationships between variables represented in this problem are in fact non-linear, assuming linearity, or at least the ability to linearize at a given point, is not such a strong assumption that it invalidates this formulation.

Many algorithms also construct a map of the world that estimates the pose of different objects in the environment. These object, key point, or landmark poses may be computed as functions of the robot pose estimate, or may be jointly optimized for during pose estimation. In the latter case, the problem to be solved takes the same form as equation (2.1), but where now $\mathbf{x} = [x_0, \ldots, \theta_n, l_0^x, \ldots, l_m^\theta]^T$, and $\mathbf{\Lambda}$ is an $(n + m) \times n$ matrix. In both cases, solving for $\mathbf{x}$ exactly involves computing the inverse or Moore-Penrose inverse of $\mathbf{\Lambda}$, an operation which is extremely inefficient, especially as n grows large and given the inverse must be computed online every time new information is available.

In many cases, in addition to the maximum likelihood pose or trajectory, we would also like to estimate the probability density function (PDF) over the parameters of $\mathbf{x}_t$ or possibly even the entire trajectory. The methods for doing so are specific to different solution techniques and we will touch on them briefly in the following subsections.

The vast majority of SLAM systems will either estimate the posterior over the most recent pose, $p(x_t|x_{0:t-1}, z_{0:t})$, or the posterior over the entire trajectory $p(x_{0:t}|z_{0:t})$. In general, there are many ways to engage with the computational complexity of equation (2.1). To-date, there are three primary classes of techniques that represent alternatives to solving (2.1) directly: 1) Kalman and information filters; 2) particle filters; and 3) non-linear optimization. In each class there has been significant research effort investigating different methods for avoiding computations which are unlikely to affect the parameters of interest, leveraging empirically validated structures commonly found in robotics applications, and using domain-specific assumptions regarding the availability or reliability of additional measurements or the relationship between measurements.

Of course, all of these methods make trade offs between different formal guarantees and aspects of empirical performance. In particular, computational cost is frequently traded for the ability to 1) represent non-Gaussian posterior distributions, 2) represent a history of poses rather than just the most recent pose, and 3) retain the ability to correct poor location estimates given future observations should the current estimate be inaccurate. In the following sub-sections, we will cover these methods and their trade offs more thoroughly.

2.2.2 Kalman Filtering

The Kalman filter, sometimes called the Kalman-Bucy filter, was first published in the early 1960s. Kalman filters are general tools for state estimation and have been used for many, many real-world applications including guidance and control of ships, aircraft, and spacecraft. Kalman filters were also the first widely adopted method for robot localization.

For roboticists, they offer a principled way to combine information from multiple sensors over time in order to estimate robot location. Kalman filters operate in two steps. First, a model for how the robot moves, called a dynamic model, forward model, or process model, is used along with control inputs to the robot's motors to predict how the robot should move if it executed the given control input. This is called the predict step. It then uses external sensor readings in combination with an observation model, the output of which likely differ slightly from the process model prediction, and combines both streams of information (prediction and observation) via a weighted average into a single estimate for how the robot has moved since its last location estimate. This is called the update step. The degree to which the process model or observation model is favored during the weighted averaging is based on the quality or uncertainty of the information source. We will now formalise these concepts and processes.

Kalman filters have two primary requirements in order to perform reliably. The first is that the forward model must be a discrete time linear dynamic system. That is, we must be able to write the forward model as

$$\mathbf{x}_{t+1} = \mathbf{F}\mathbf{x}_t. \tag{2.2}$$

Now, Eqn. (2.2) only encodes what would happen to our state $\mathbf{x}$ if we let it evolve passively according to the laws of physics and did not send any control inputs, such as commanding a certain torque from some motor. We will use the variable $\mathbf{u}$ to denote the control inputs we might send, and we will represent by the matrix $\mathbf{B}$ the effect control inputs will have on our state.

$$\mathbf{x}_{t+1} = \mathbf{F}\mathbf{x}_t + \mathbf{B}\mathbf{u}_t. \tag{2.3}$$

In many robotics settings we have access to proprioceptive sensors such as joint encoders, odometers, accelerometers, and gyroscopes. These sensors operate at much higher rates than exteroceptive sensors, typically greater than 100Hz. Thus, whenever we receive new exteroceptive data and wish to estimate state, we can usually numerically integrate readings from these sensors and use the result of this integration in place of the control vector, in place of the product $\mathbf{B}\mathbf{u}_t$, or even in place of the process model, depending on the dynamics of the system. Hereafter, we will refer to control inputs and integrated proprioceptive sensor data interchangeably. In a world of perfect models, this would be our complete state update equation. However, our process models (or proprioceptive sensors, or both) are imperfect. Kalman filters deal with this imperfection by modeling the error $\mathbf{w}$, or "noise, between the estimate for $\mathbf{x}_{t+1}$ given by Eqn. (2.3) and its true value:

$$\mathbf{x}_{t+1} = \mathbf{F}\mathbf{x}_t + \mathbf{B}\mathbf{u}_t + \mathbf{w}_t. \tag{2.4}$$

The second big assumption necessary for performant Kalman filters is that this noise term, and others we will introduce later, are sampled from zero-mean Gaussian distributions with known variance. Now, Eqn. (2.4) is clearly a recurrence relation, and if we were to apply this estimator repeatedly, it is also clear that the estimated value of $\mathbf{x}$ would quickly diverge from its actual value due to the repeated addition of $\mathbf{w}$ terms.

Exteroceptive data $\mathbf{z}$ allows us to avoid this divergence via an observation model, $\mathbf{H}$, which, together with $\mathbf{z}$, offers an *independent* estimate of state.

$$\mathbf{z}_t = \mathbf{H}\mathbf{x}_t + \mathbf{v}_t. \tag{2.5}$$

Of course this model is also imperfect, and the Kalman filter framework again assumes its errors may be characterized by a vector $\mathbf{v}$ drawn from a zero-mean Gaussian distributions with known variance. Thus, in summary we have

$$\mathbf{x}_t = \mathbf{F}\mathbf{x}_{t-1} + \mathbf{B}\mathbf{u}_t + \mathbf{w}_t \quad \text{and} \quad \mathbf{z}_t = \mathbf{H}\mathbf{x}_t + \mathbf{v}_t, \tag{2.6}$$

where the covariance of the process, $\mathbf{\Sigma}_F$, and observations, $\mathbf{\Sigma}_H$, are such that

$$\mathbf{w} \sim \mathcal{N}(0, \mathbf{\Sigma}_F) \quad \text{and} \quad \mathbf{v} \sim \mathcal{N}(0, \mathbf{\Sigma}_H). \tag{2.7}$$

In many cases, especially when $\mathbf{u}$ represents proprioceptive data or the robot remains static in the absence of control inputs, $\mathbf{F}$ or $\mathbf{B}$, or both, may be the identity matrix.

Now, we present the update algorithm for the vanilla Kalman filter. To begin, at time t, we have the state estimate $\mathbf{x}_t$ and our uncertainty (covariance) about our state estimate, $\mathbf{\Sigma}_{X_t}$. We then calculate $\mathbf{x}_{t+1}$ and $\mathbf{\Sigma}_{X_{t+1}}$ as

$$\mathbf{x}_{t+1} = (\mathbf{F}\mathbf{x}_t + \mathbf{B}\mathbf{u}_t) + \mathbf{K}_t(\mathbf{z}_t - \mathbf{H}(\mathbf{F}\mathbf{x}_t + \mathbf{B}\mathbf{u}_t)) \tag{2.8}$$

and

$$\mathbf{\Sigma}_{X_{t+1}} = (\mathbf{I} - \mathbf{K}_t\mathbf{H})(\mathbf{F}\mathbf{\Sigma}_{X_t}\mathbf{F}^T + \mathbf{\Sigma}_F), \tag{2.9}$$

where

$$\mathbf{K}_t = (\mathbf{F}\mathbf{\Sigma}_{X_t}\mathbf{F}^T + \mathbf{\Sigma}_F)\mathbf{H}^T(\mathbf{H}(\mathbf{F}\mathbf{\Sigma}_{X_t}\mathbf{F}^T + \mathbf{\Sigma}_F)\mathbf{H}^T + \mathbf{\Sigma}_H)^{-1}. \tag{2.10}$$

The $(\mathbf{z}_t - \mathbf{H}(\mathbf{F}\mathbf{x}_t + \mathbf{B}\mathbf{u}_t))$ term is often called the innovation and roughly represents the amount of agreement between the predicted state coming from the process model and the measured state estimated by the observation model. The expression we highlight as $\mathbf{K}_t$ is called the Kalman gain and represents how heavily the weighted average is skewed towards trusting the process model versus observation model. Kalman filters may be described as ‘high-gain’ where they rely more on observation (exteroceptive) data, or ‘low-gain’, in which they rely more on the process model.

Before introducing several extensions to the vanilla Kalman filter, we highlight some benefits and characteristics common to all Kalman filters. First, they use a recurrence relation, or operate recursively, meaning that only the most recent state estimate, rather than its entire history, is required to estimate the next state. This makes computation extremely fast. Second, if the (admittedly, strong) assumptions of linearity and zero-mean Gaussian noise are met, then the Kalman filter is an *optimal* estimator, minimizing the mean squared error of the estimated state parameters. If the Gaussian assumption is broken, then the Kalman filter remains the best linear estimator, although non-linear estimators may be better.

Despite these benefits, and the deployment of vanilla Kalman filters in many domains [91, 232, 380, 100], there remain serious drawbacks when using them for estimating robot location.

2.2.2.1 Kalman Filter Variants

Here, we briefly cover some of the most popular variants of the Kalman filter, designed to overcome the vanilla version’s chief flaws.

2.2.2.1.1 Extended Kalman Filter The extended Kalman filter (EKF) relaxes the linearity assumption on the process and observation models in the basic Kalman filter. This is done by instead assuming that these models are simply differentiable:

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_t) + \mathbf{w}_t, \quad \mathbf{z}_t = h(\mathbf{x}_t) + \mathbf{v}_t \tag{2.11}$$

and then using the matrix of partial derivatives of these new models, their Jacobians,

$$\mathbf{F}_t = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\mathbf{x}_{t-1},\mathbf{u}_t} \quad \text{and} \quad \mathbf{H}_t = \left.\frac{\partial h}{\partial \mathbf{x}}\right|_{\mathbf{x}_t} \tag{2.12}$$

to replace the original, linear models. These change are then propagated to the covariance estimates as well during the update process. Second-order EKFs refine their process model estimate by computing the second derivative, represented by the Hessian matrix, in addition to the Jacobian. Generally, EKF-SLAM has had some is one of the most popular choices for KF-based SLAM systems [276, 370].

However, because the linearization operation results in an approximation of the underlying model, the filter is no longer optimal. Moreover, EKFs may still diverge if the models are highly non-linear [145], and often produce over confident covariance estimates [153].

2.2.2.1.2 Unscented Kalman Filter The unscented Kalman filter (UKF) [153] further extends the ability of the Kalman filter framework to handle models that are non-linear to an even greater degree. It does so by propagating a set of sample points through the non-linear process and observation models and then estimating the mean and covariance using these transformed samples. A valid set of (weighted) sample points for an state vector $\mathbf{X}$ is any set of points and weights which have the same weighted first and second moments as the *a priori* distribution.

After evaluating the non-linear process model at each of the sample points, the predicted state and its covariance are estimated by again weighting the resulting points from the discrete samples during computation. The observation mean and covariance are computed in the same way, and the update step resembles the basic Kalman update. This process skips the linearization process entirely, avoiding the need to compute Jacobians and relaxing the assumption of differentiability of the models. However, it raises a new question, which us how to best pick the sample points. For example, a recent extension of the UKF, the Cubature Kalman Filter (CKF) [15], samples points according to the Cubature rule, which applies Gaussian quadrature to state vectors after they have been transformed from Cartesian space to spherical coordinates. Overall, UKF-SLAM systems are the predominant choice for KF-SLAM and have been studied extensively [340, 140, 146].

In fact, both UKFs and CKFs are special cases of a more general type of filter called a Gaussian filter, based on the general idea of matching the moments of the underlying real-valued distribution and the discrete distribution represented by the samples. One other popular type of Gaussian filter is the Gauss-Hermite Kalman Filter (GHKF), which uses a multi-dimensional Gauss-Hermite quadrature to numerically solve the Gaussian integral. While effective in many applications, the UKF has a few disadvantages. First, it is more computationally demanding, since samples need to be selected and evaluated at every time step. Moreover, EKFs, UKFs, and CKFs all still require Gaussian error in process and observation models.

2.2.2.1.3 Non-Gaussian Kalman Filters To deal with non-Gaussian error, a number of techniques have been developed, primarily based on tools for approximating non-Gaussian distributions with parameterizations that are still tractable for computation. One such example is the Gram–Charlier or Edgeworth expansion [336], which uses a series of Hermite polynomials to represent different density functions.

Another approach is the so-called Gaussian-sum approximation, where non-Gaussian distributions are approximated by a finite sum of Gaussian distributions. This has produced different forms of Gaussian-sum filters (GSF)[16, 11]. Both predicted and posterior densities are represented as sums of Gaussians, and the moments each Gaussian are found using a method of choice. This may be an EKF [11] or a more complex method. Thus, the GSF is essentially a set of Kalman filters running in parallel, where each individual filter provides a weighted state estimate, and the final estimate is computed by combining the individual estimates according to their weights. These methods are more accurate, but maintaining an ensemble of filters large enough to effectively approximate the non-linear distribution can become computationally untenable.

2.2.2.1.4 Adaptive Kalman Filter The above Kalman filters assume a constant, correct process and observation models and constant, correct uncertainty in process and observation models. However, in many cases this is not possible. The adaptive Kalman filter [225, 226] is able to identify incorrect models and fix them on-line, using the data generated during the first part of the deployment. This is done by analyzing the innovation term (Eqn. 2.8) and its covariance, which correspond to the error between the predicted state and observed state. When the filter is operating correctly and all assumptions are met, the sequence of error terms (innovation terms) will form a zero-mean Gaussian distribution. When this is not the case, the covariance matrices can be updated via fixed-window averaging [234], recency weighting [4], or other techniques so that the distribution of innovation terms becomes Gaussian.

One other notable form of adaptive Kalman filter is called multiple-model adaptive estimation [212, 223]. These methods use an ensemble of Kalman filters with different parameters for noise and or process models, the estimates of which, over time, are up- or down-weighted according to their accuracy. Ultimately, they model whose parameters most closely match the data will have the highest weight and will be most influential in the final state estimate. All forms of adaptive Kalman filters may be combined with other Kalman filter extensions, since they tackle the relaxation of different assumptions.

2.2.2.1.5 Indirect Kalman Filter Unlike the previous variants, indirect Kalman filters (IFK) [222] do not directly estimate state. Instead, systems that use IKFs assume that a reasonably accurate estimate of state can be attained through direct methods such as odometry, dead-reckoning, or perhaps another Kalman filter [113]. Meanwhile the IKF runs on different information, perhaps provided at a lower frequency, such as GPS, and estimates the error associated with the direct state estimate. This error estimate can then be used to intermittently correct the state estimate. IKFs also somewhat alleviate the need for a direct filter to supply a highly accurate and perhaps computationally expensive state estimate at a high frequency, allowing the main control loop of the system to operate with less accurate direct state estimation.

In practice, IKFs have enjoyed some success in robotics applications [113, 381]. However, they too suffer from the same set of shortcomings that limit other forms of Kalman filters.

2.2.2.2 Summary

Overall, Kalman filters and their variants are an attractive way to localize a robot, so long as their assumptions are met. They are generally computationally cheap, fairly simple to program, and do not require storing a large amount of data. Moreover, their various extensions make them viable under a surprisingly large number of conditions. These include unmanned aerial vehicles [132], autonomous submarines [383], autonomous vehicles [417], and many other sub-problems within visual SLAM [59].

However, since in most applications the linearity and Gaussian noise assumptions do not hold, Kalman filters require one or more extensions in order to produce reasonable results. Moreover, in these cases, not only are Kalman filters not optimal, but they can produce state estimates that are very far from the true values. In addition, once the true state has diverged from the filter's estimate, recovering the correct state, even after observing additional data, is nearly impossible. This is primarily because the distribution of state estimates is constrained to be a Gaussian distribution, which is uni-modal, and the Kalman filter is a filter, meaning it only computes state estimates based on the previous estimate and the current data. It does not have the ability to continually re-examine or reason about passed sequences of data. Additionally, as the notion of state becomes more extensive, perhaps to include changing parts of the environment beyond the control of the robot, the Kalman filter framework becomes harder and harder to use effectively for SLAM systems.

Nonetheless, although this thesis does not highlight further results on Kalman filtering specifically, they remain an active area of research and a very important tool for many other types of control systems. There are many other variants of Kalman filters not presented here that have been designed for domains beyond robotic localization and SLAM.

2.2.3 Particle Filtering

Particle filters, introduced in their current form in 1993 [114], were developed in response to some of the shortcomings in Kalman filters. Most notably, particle filters are multi-modal, do not require strictly Gaussian error models or linear systems, may represent arbitrary posterior distributions, and can recover the correct pose estimate even after periods of incorrect estimation, known as 'divergence' in Kalman parlance. Their chief difference from Kalman filters is that they use random sampling to approximate both *a priori* and *a posteriori* probability density functions on the state variables. This empirical support for the distributions allows them to relax the Gaussian assumptions. In fact, in a linear system with Gaussian noise, a vanilla Kalman filter and vanilla particle filter will produce the same results as the number of particles tends to infinity. Furthermore, because particle filters do not assume Gaussian distributions, they are no longer constrained to the space of linear process or observation models. Much of the Bayesian theory we rely on today to understand these filters was first introduced in the late 1990s [82, 83].

Algorithms based on random sampling and simulation are often called "Monte Carlo" methods, a moniker inspired by the Monte Carlo Casino in Monaco. Virtually all such algorithms are approximations, and thus not optimal, instead relying on empirical validation. The most basic versions of particle filters are in fact related to an extension of the Kalman filter, called the Ensemble (or Monte Carlo Multiple-Model) Kalman filter (EnKF), where the primary difference is that all probability distributions in the EnKF are assumed to be Gaussian. These methods have applications both within and beyond robotics, and there is ongoing research on which estimation method is best for different applications [78, 275, 203, 390].

The estimation problem remains the same, we want the probability of state $\mathbf{x}_t$ given the the previous state $\mathbf{x}_{t-1}$ and some observation $\mathbf{z}_t$: $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{z}_t)$. As with the Kalman filter, we have process and observation models, although they need not be linear or differentiable.

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_t) + \mathbf{w}_t, \quad \mathbf{z}_t = h(\mathbf{x}_t) + \mathbf{v}_t. \tag{2.13}$$

Here, $f()$ and $h()$ have no restrictions other than that they are reasonable representations of $p(x_t|x_{t-1}, u_t)$ and $p(z_t|x_t)$, respectively, and that they can be evaluated efficiently.

The basic idea behind a particle filter for state estimation is to first sample several potential state values (particles) from the prior and propagate those samples according to any proprioceptive measurements made by the robot. At this point the set of sampled particles is analogous to the belief represented in a Kalman filter after the process update step. Here, the propagated samples of the robot's belief about its state are then re-weighted according to the support they lend to the exteroceptive measurements. The distribution represented by the particles after re-weighting is analogous to the final belief after applying the Kalman innovation.

Particle filters suffer from several disadvantages, all of which manifest as computational burdens due to various forms of particle inefficiency. That is, in practice vanilla particle filters often require a very large number of particles to work robustly. This is due to several reasons. First, as the state space grows, providing coverage via sampling becomes exponentially more expensive. Second, if the dimension of the state is high, then there is the possibility of so-called 'weight collapse', where all weights for all particles approach unity [31]. This of course destroys most of the information captured in the distribution quite quickly.

Third, during the re-sampling step, the particle representing the true closest estimate can be eliminated. This is known as particle depletion. Without more advanced sampling strategies, the primary method to combat particle depletion is simply to add more particles. A significant body of research has been done on various ways to mitigate these problems, with encouraging results. Below, we highlight a few of the most important insights.

2.2.3.1 Particle Filter Variants

Here, we briefly cover some of the most important extensions to vanilla particle filters.

2.2.3.1.1 Sequential Importance Sampling Sequential importance sampling (SIS), originally developed in statistical physics [122, 309], is employed when particles are sampled from a high-dimensional, possibly non-Euclidean space [364]. This is extremely common in localization applications, as particles frequently represent both positional and rotational components as well as potentially aspects of the map. The main idea of SIS is to approximate the high-dimensional, non-parametric distribution from which we wish to sample by a sequence lower-dimensional, parametric distributions that are thus easier to sample from.

2.2.3.1.2 Sequential Importance Re-sampling In regular SIS, sampled particles may have very small weights and thus contribute very little to the overall estimate. Re-sampling allows particles to be deleted in these cases or re-weighted in the event they are drawn from a meaningful part of the distribution. The key insight here is that propagating many particles which are unlikely to be near the most likely state estimate will significantly degrade the estimate over time and should thus be avoided. Technically, this problem can also be mitigated by using significantly more particles, but for obvious reasons this is undesirable.

2.2.3.1.3 Rao-Blackwellization The motivation for applying the Rao-Blackwell theorem [301, 44] to particle filters, often referred to colloquially as Rao-Blackwellization, is that by conditioning on the data associations and the sample deletion and re-sampling processes, the posterior distributions of the states of the particles can be approximated with Gaussian distributions. Thus, the particle states can be integrated out analytically and the particle filter only needs to be applied to the data associations and deletion and re-sampling processes. This significantly reduces the computational requirements and increases the efficiency of the particle filter.

2.2.3.2 Summary

Particle filters address the two most restrictive shortcomings of the Kalman filter. They allow for arbitrary process, measurement, and noise models, removing the requirement for linearity and Gaussian behavior, respectively. They also allow for recovery of the correct state estimate after divergence. This is due to the particle filter's ability to support multi-modal posterior distributions, which allow for the robot to maintain a useful belief about its state. This is a critical benefit in robotics since many sensor reading are ambiguous. However, particle filters are slow, due to sample complexity rising exponentially with state dimension. Although a relatively straightforward drawback, it is a major hurdle to adopting these techniques in practice as we can see from the subsequent sampling strategies proposed, all of which aim to increase sample efficiency.

2.2.4 Pose Graph Optimization

Beginning in 1997 with a seminal paper by Lu and Milios [206], roboticists studying SLAM were offered a radically different approach. While Kalman filters and particle filters directly estimate posterior distributions over state variables by evolving parametric or empirical a priori distributions, graph-SLAM approaches instead solve a dual representation of the problem which is represented as cost minimization [105, 358]. Pose-graph solvers first represent the state estimation problem as a dynamic Bayesian network (DBN) which grows over time as new time steps pass at which the robot receives date and would like to estimate state. This DBN is then converted into a factor graph, where variables representing robot poses at different times are connected to each other via factors if and only if there is a measurement, proprio- or exteroceptive, that relates the two poses. For example, for subsequent poses, inertial measurements taken in between the two time steps represent a factor. For poses separated by some time, re-observation of a visual feature could create a factor between the two observing poses.

Given process and measurement models, including uncertainty, we can write cost functions that represent the inverse probability of the robot measuring certain data and also have been in different relative locations. Taken collectively over all measurements minimizing this total cost by changing the estimate for the pose variables is equivalent to maximizing the likelihood of the trajectory.

Several key factors have since helped pose-graph SLAM become the state-of-the-art approach for most SLAM problems. First, consistently faster and more capable processors have made large optimizations problems more feasible for online solving. While there has been significant research into theoretical improvements to these optimizers as well, increased compute power, combined with better scaling than particle filters, has made pose-graph SLAM an option for all but the most compute-bound systems. Second, the advent of auto differentiation has drastically increased the performance of modern solvers, and has also allowed researchers to provide much more complex cost functions without needing to derive Jacobians by hand, or to rely on notoriously unstable numerical differentiation techniques.

Additionally, pose-graph solutions can more reliably deal with non-linear models and non-Gaussian noise than Kalman filters, while also solving the 'full' SLAM problem in which the entire history of poses, or trajectory, can be estimated, not just the current pose. While this is also true of particle filters, they are substantially less efficient in these cases, especially if one is also jointly estimating elements of the map independently from the trajectory.

More than simply practical improvements and theoretical advantages, pose-graph SLAM also offered an easy way to incorporate other types of inference without the burden of representing additional factors as components of a process or observation model or formulating a process for representing them in a Monte Carlo sampling procedure. In particular, it allowed SLAM researchers to experiment with modeling many phenomena that are more transient in nature and may have less bearing on immediate location, such as the approximate region in which couches and chairs are likely to be observed, or the times during which a hallway is likely to be crowded.

2.2.4.1 Cost Function Design

One of the main benefits of pose-graph SLAM is the ability of modelers to specify complex cost functions. Cost functions may express non-isotropic errors, or represent correlations between variables that might otherwise be difficult to express. One challenge in cost function design of course is modeling uncertainty. In simpler estimation frameworks, like the Kalman filter, one need only specify a co-variance matrix, or even just a list of variances. However, in more complicated cost functions, it can be challenging to accurately model the covariance and the rate at which the probability density function it is describing ought to change as it moves away from the mean.

2.2.4.2 Efficient Optimization

The main bottleneck of pose-graph systems is of course the non-linear least-squares optimization procedure. The basic solver configuration is typically to use stochastic gradient descent, and there are some established solvers, such as Ceres[3] and g2o[116]. However, because of its complexity, there have also been a number of efforts to either gain efficiency by exploiting additional structure in the problem [2, 84] or sparsity [22], or by incrementally solving the problem [158, 157, 202] or simply approximating the solution [53], potentially through variable marginalization [52].

2.2.4.3 Robust Optimization

Perhaps one of the most vexing problems in pose-graph optimization is the havoc that can be caused by false loop closures. That is, constraints added between distant poses along the trajectory that are not correct. Until recently, this was an unresolved issue. However over the last several years, a significant body of work on different techniques for 'robust' pose-graph optimization has evolved [185, 342, 281]. These techniques principally revolve around detecting and then rejecting loop closure constraints that seem erroneous.

2.2.4.4 Summary

Pose-graph SLAM offers a relatively efficient and highly expressive formulation for finding maximum likelihood trajectories. Although far more complicated to program than either particle filters or Kalman filters, and occasionally being a victim of robustness and stability issues, researchers have generally concluded that the ability to correct for past mistakes justifies the additional complexity. Thus, as of 2023, pose-graph SLAM systems are considered the best method in general for most SLAM applications.

2.3 Environment Representations (Mapping)

We often call representations or models of the environment “maps” as their original purpose was to delineate occupied and unoccupied regions of the environment to facilitate path planning and allow robots to navigate safely and efficiently. However, as the goals and capabilities of robotic systems have become more complex, and planners of all types have come to rely on richer sets of information, the role of maps has significantly expanded. In modern systems, maps may support other forms of reasoning, such as computing where to look for certain people or objects, predicting when a robot may need intervention or help, or providing a grounding of natural language phrases to physical locations. However, navigation, as one of the earliest and most fundamental goals of mobile robotics, is the only task supported nearly universally.

In the next subsections, we cover the benefits and drawbacks associated with using the most common types of maps. Volumetric, surface, and landmark models are types of metric maps, encoding purely spatial information. Topological maps are distinct in that they may not represent distances explicitly, but represent the structure of the environment and its connectivity in other ways. Both metric and topological maps may be extended to include semantic information and may also be combined in hybrid representations. Last, we include some discussion of sub-mapping, wherein the environment is modeled hierarchically.

2.3.1 Volumetric Models

Volumetric models, often called occupancy grids, were originally proposed in the mid-late 1980s in a sequence of papers by Alberto Elfes [98], occupancy grids represent all of the robot's operating environment as a lattice of cells of pre-determined size. For example, a grid mapping a single square meter of area using square cells of size 1 centimeter would result in a map that was 100 cells long and 100 cells wide, with a total of 10,000 cells. Each cell is labeled either empty, occupied, or unknown. Empty and occupied labels exist on a continuum, allowing a degree of confidence or belief to be assigned to each cell regarding its label, and allowing these beliefs to be updated over time as new evidence becomes available. There are several common strategies for updating these beliefs, depending on the type and quality of sensor data available and the dynamics of the environment. These include updating belief using Dempster-Shafer theory [382], representing dependencies between grid cells using Bayes nets[32], and filtering out observations from moving objects[238]. Unknown cells represent parts of the environment that have yet to be observed. Constructing occupancy grids requires data that measures depth or distance. This could come from a sensor that measures time of flight, like sonar or a laser, or one uses disparity calculations between two images, like structured light sensors or a stereo camera system.

Volumetric models offer several benefits, not least of which is their simplicity, intuitiveness, and ease of programming. They also synergize well with graph-based path planners, such as A*[125] and D*[338]. They also offer an ability to trade memory efficiency for fidelity via the specification of cell size. Given the widespread adoption of occupancy grids and their long-time status among the state-of-the-art techniques, it is no surprise that many other extensions to the basic occupancy grid have been explored, including clustering grid cells as higher level entities [262], using expectation maximization to solve for maps [357, 356], and online parameter tuning [228].

The 2D notion of occupancy grids has also been extended to 3D models many times [81, 135]. However, because they represent both empty and occupied space, they become exponentially less efficient as the dimensionality of the model increases. For example, representing a 10m$\times$10m room at centimeter-level fidelity requires, 1 million cells, whereas representing the same room in 3D up to a height of 3m at the same resolution requires 300 million cells. One method for dealing with this decrease in memory efficiency is to apply a form of lossless compression by storing occupancy grids within quad-trees (2D) [197] or oct-trees (3D) [151], such that large homogeneous sections of the environment can be compactly expressed as a single leaf node near the root rather than a large number of identical nodes at a lower level. Unfortunately, this method only works in relatively sparse environments where this compression can be maximally exploited. Occupancy grids can also be expensive to update as new data arrives, particularly if it provides information about a large area.

2.3.2 Surface Models

Surface models for mapping, also a product of the mid 1980s, were first proposed by Chatila and Laumond [57]. Unlike volumetric models, surface models do not explicitly represent empty space. This, combined with essentially no limit on their fidelity due to the absence of a prohibiting factor like grid cell size, make them a very desirable choice for mapping in theory. However, the choice to model surfaces directly creates several follow-up problems which the intervening decades have shown are arguably more challenging than finding engineering workarounds to unfavorable memory/fidelity tradeoffs.

Like occupancy grids, surface models rely on depth data. However, instead of representing depth data as being generated from independent grid cells, which may take arbitrary geometric forms, they represent data as being generated from a reconstructed or parameterized surface. Examples include line segments [413], planes [263], and B-splines [278]. This has several benefits. First, it is vastly more efficient with respect to memory use, particularly in 3D, often reducing use by several orders of magnitude. Second, parameterized models can sometimes offer better natural defense against some types noise since outliers that do not fit the model closely enough will be rejected. Moreover, their (often) larger size size makes them more reliable in terms of re-observation and correspondence calculation. Last, it is sometimes possible to use these models as tools for extrapolation about yet unseen parts of the environment, since their parametric forms are much easier to reason about than the structure of many independent grid cells.

However, surface models do have several drawbacks. Although they often reject outliers, any outliers that are not rejected can have a large impact on the parameters of an individual feature since parametric models are often fit using least squares, which is very susceptible to noise. Computationally, it is also marginally more expensive to run several instances of model-fitting every frame than to simply update an occupancy grid, and these methods are not amenable to parallelization. Furthermore, developers need to decide before deployment which types of parametric models to use. This can have drastic impacts on the performance during deployment depending on how well-suited the operating environment is to description by the chosen parametric model. Last, surface models often require additional definitions, such as rules for extending, merging, or removing, different sections of the map represented by different models, or how to represent the uncertainty associated with a model's location. Many of these shortcomings are addressable in practice, but they nonetheless inspire substantial ongoing research.

2.3.3 Landmarks and Keypoints

The concept of building maps from collections of landmarks or keypoints is likely the earliest metric strategy explored by roboticists, and grew naturally out of many related efforts in computer vision in the early 1980s. One major advantage of landmark maps is that they do not require (although they are compatible with) depth sensors, which historically are more expensive and work accurately in fewer domains. Furthermore, in deployment contexts where there is an opportunity to engineer the environment, landmark systems are often the best choice to maximally exploit this. For example, robots designed to carry pods around the floors of large Amazon warehouses use a version of landmark-based localization, where unique, easily identifiable tags are placed on the floor, allowing a simple downward-facing camera system to observe them reliably and thus localize reliably. This method is suitable because there is ample opportunity to engineering the environment, and the additional cost of the tags and complexity of the camera system is very small compared to building an equally robust general purpose localization system.

However, in the open world, unstructured or unpredictable environments present a significant challenge to landmark maps. Primarily, this stems from the difficulty of defining how a landmark should appear. Generally, we know the properties we would like them to have: a) easily detectable, b) easily disambiguated, c) numerous, d) stationary, and e) retaining properties a) - d) under all possible sensing conditions the robot may encounter. Defining these properties mathematically has turned out to be one of the most challenging problems in robotic perception and is still an open question in the general sense. Despite this, there has been considerable success in a related problem known as keypoint detection. Mathematically, the role of keypoints within maps and localization problems is very similar. The most common distinction, which we will adopt here, is that keypoints occur on the 2D image plane, while landmarks are full 3D points, potentially with an orientation component as well. Some researchers also maintain a difference in scale, stability, and information as well. For example, a piece of paper (a plane) with a geometric pattern printed on it (such as an AR Tag) contains enough information to completely determine the pose of a camera observing the tag, while also potentially being composed of many keypoints, which do not individually contain the same degree of information. Moreover, as a virtue of its relative size and salience, the estimated pose of this landmark is much more reliable than the estimated position of a single keypoint.

There has been a large volume of research on keypoint detection and description methods for both camera and depth data. These include hand-crafted local feature descriptors, such as SIFT [205], SURF [27], BRIEF [51], BRISK [188], ORB [311], FREAK [5], and

AKAZE [6], as well as methods for computing local image statistics, such as histograms of oriented gradients (HOG) [76], discrete wavelet transforms, including Haar wavelets, discrete Fourier transforms, and Zernike moments [168], among others, are gradually being replaced by local image descriptors learned using convolutional neural networks (CNNs) or generative adversarial networks (GANs) [326]. These methods have been shown to generate numerous reliable keypoints in well-lit, cluttered environments, and there have been several SLAM systems that been developed on the backs of these descriptors. However, they still often produce keypoints that correspond to cannot be reliably re-observed, such as reflections in street puddles or moving objects like people and cars. While modern solvers are improving in their ability to deal with large numbers of bad keypoints, this is still far from the idea as it can still cause compute and stability problems. Recently, there has been some work on trying to reject keypoints if they are determined to come from an unreliable source [350, 187].

Landmark maps are the simplest of all environment representations. They simply maintain a list of landmarks — everything else is thrown out. This can be incredibly effective when a small number of high-quality landmarks are available, and in extreme cases, some research looks at learning which landmarks are useful, with the goal being to limit number of landmarks [339]. In some sense, models which store all raw depth data could also be considered a landmark map, where every depth reading is a landmark. However, for obvious memory reasons, this is not a good strategy for most applications. Moreover, landmark maps cannot be used for physical planning since they do not contain enough information related to the presence or absence of obstacles on order to guarantee safety.

2.3.4 Topological Maps

The application of topological maps to robot navigation and localization was pioneered by Ben Kuipers in 1978 [178]. Unlike the previous representations, they do not necessarily encode metric data, and instead represent the environment as a graph, where nodes represent ‘places’ (for example, rooms of a house), and edges represent the possibility of traversing from one place to another. These edges may be given values associated with a notion of distance or time between nodes. One popular example of topological maps are the navigation systems on phones or within cars. While they present an interface that also shows metric information, the underlying planning and navigation is done using a topological graph representing the connectivity and relative cost of moving between different points on the road network.

Topological graphs in their purest form are primarily used for route planning. However, many robotic systems combine aspects of topological maps with other representations in order to facilitate better planning across tasks beyond just navigation, as we will discuss later. Topological maps do have several advantages. First, they are very simple to program, maintain, use, and understand. Second, they offer a much easier and more flexible way to represent the cost of moving between locations. Third, they are many orders of magnitude more space efficient than all other types of maps, primarily because they encode much lower fidelity data. Last, there is a large body of work on ‘place recognition’, which is the task of recognizing when the robot is re-visiting a region [253]. This is a slightly easier and less informative problem than true loop closure, but for localization within topological maps, it is sufficient.

Obviously, topological maps, due to their poor fidelity, are generally not sufficient for supporting mobile robot operation independently. However, they do support several key functions elegantly, which makes them a mainstay component of most mobile robotic systems.

2.3.5 Semantic Maps

Semantic maps contain semantic information, which in robotics typically means additional labels or prior beliefs. They are somewhat of a misnomer in that there are no map representations that contain purely semantic descriptions of places, and their popularity as a research topic has correlated with the attention paid to the problem of robotic mapping in general. These labels and any associated concepts or statistics are always additionally associated with either a topological map, in which case the node labels would contain additional meaning, or a metric map, in which case at least parts of the map would be labeled with additional information [108, 387].

Semantic maps facilitate two important functions. The first is human interpretability in interaction. For example, some modern campus, office, or home robot systems augment their metric maps with semantic information regarding either the natural language description for a set of locations, such as "kitchen" or "break room," or information regarding privacy, such as "do not enter" labels, or operational complexity, such as labels identifying busy streets or crowded hallways. These labels not only add nuance to the robot's model that can be used for planning, but also increase the ability and proficiency with which humans can interact with the model, either to update it or to use it for their own independent purpose.

The second important function is connection to other models. Allowing semantic information to be encoded into the map opens up a much larger space of potential information beyond simply describing the physical space the robot operates within. Once information, of any kind, can be stored and accessed efficiently within a model, it has the potential to support planning, and therefore enables robots to reason about tasks that would otherwise be either far too expensive and complex, or simply not formally represented at all. Of course, having the ability to represent complex knowledge is different than having a compact, effective representation, and research on this front has been and will continue to be highly impactful and influential for current and future robotic systems.

One reason for this ongoing research is that semantic mapping is not simple. Most notably, labeling automatically can be difficult and hand labeling is extremely tedious. Moreover, the information that different researchers have argued falls under the heading of 'semantic' is very broad, meaning that the way in which these pieces of information can be used (e.g. which types of planners can take advantage of them and for which types of tasks) is not homogeneous. Therefore, these systems can require drastically more complex programs to integrate the new information into the robot's reasoning.

2.3.6 Hybrid Maps

Hybrid maps are catchall term for maps that combine multiple functionalities discussed above. One particularly compelling example are conceptual-spatial maps [409], which represent several concepts at different levels of abstraction. Other examples have extended these general ideas to create even more informative and performant systems [295]. Generally speaking, research on hybrid maps focuses on integration of knowledge, either with the robot or with human users, for example, via natural language, rather than developing more precise or efficient representations. Research on hybrid map systems is still much less mature than metric or topological mapping, and many of these systems represent the state-of-the-art for deployed or commercial systems which both perform autonomous navigation tasks as well as interact with humans.

2.3.7 Local Submaps and Multi-Robot Maps

For as long as roboticists have wanted to build models of operating environments, they have debated whether it is best to have a singular, global model, or a collection of partially overlapping local maps. Any of the above types of representation could theoretically be stored as either a single monolithic map or a set of local maps, but practically this distinction makes little sense for topological maps. Proponents of local maps, or localists, argue that maintaining local maps is preferable due to their lightweight memory and compute options when updating or caching, their better synergy with multi-robot systems for exploration, and protection against wide scale map degradation during optimization should a particular submap be corrupted [265, 359]. In other words, failures during map building may be confined to a local map rather than propagate to the whole map.

On the other hand, globalists argue that global representation facilitate easier and more efficient planning, and can sometimes allow encoding more information about relationships between different entities in the map. Moreover, global maps support better metrics for understanding and maintaining global consistency within the map. Last, globalists also argue that maintaining multiple maps actually creates an additional, difficult problem, which is to properly align multiple, partially overlapping maps in order to ensure accurate models of the environment at the boundaries. Overall, there is not a clear consensus yet on the best approach, and the answer will likely depend on the specific capabilities of the robot and the deployment conditions.

2.3.8 Conclusion

There are clearly many options for representing the environment. Often, their fitness for a given application is a complex function of a variety of factors including the nature and dynamics of the operating environment, the quality and quantity of data from different sensing modalities, the ease of programming and maintaining the software, the availability of memory, and the predictability and cost of failure. Unfortunately, there is no universally best option. In fact, many systems end up using more than one of these representations, often in an ad-hoc way, as it is determined during development that certain localization, planning, or safety needs cannot be simultaneously satisfied by a single representation. Thus, research on more comprehensive representations remains an important effort.

2.4 Anatomy of a Modern SLAM System

2.4.1 Overview

Generating maps using pose-graph optimization follows a general process. Data from sensors is combined with the existing map and an estimate of the robot's previous location to estimate the robot's new location. Given this new location, the sensor data is combined with the existing map to estimate the new map. When new data from a sensor is available, this process is repeated using the most current estimates of the map and robot location. Below, we detail these processes roughly in the order in which they occur in an actual implementation. Naturally, as the process progresses, higher-level or more abstract data representations are considered and manipulated and such sub-processes often occur at increasing time-scales.

2.4.2 Feature Extraction

2.4.2.1 Proprioception

Strictly speaking, there are no features extracted from proprioceptive data. However, since these sensors typically operate much faster ($\approx$10x) than the exteroceptive sensors, they produce a sequence of data (a time-series) over which some pre-processing must still be run prior to creating factors in the factor graph. Generally, there are two operations: smoothing or filtering, and numerical integration. The former is technically optional, but in systems where the sensor has a known bias or is very noisy, it is common to remove the bias and apply some flavor of moving average. Integration is performed to transform the time-series data into a single data item representing the cumulative motion measured by the proprioceptive sensors between the last exteroceptive reading and the current one. This occasionally also involves interpolating time-series points to exactly match the time bounds of the integration period to the timestamp of the exteroceptive data. The product of this process is an estimate of the robot's current pose using proprioception alone, similar to the process update step in a Kalman filter.

2.4.2.2 Exteroception

The specifics of feature extraction on exteroceptive data vary significantly, depending on the modality. Generally speaking, features encode local signals within an image or a point cloud that represent both the appearance of a particularly salient part of the signal as well as its location relative to the robot at the time it is sensed. Exteroceptive features are usually found in two steps. First, a feature *detector* is run. The feature detector applies, usually through some form of convolution, a detection pattern across all areas of the input. A common example for feature detectors are corner detectors[235, 21].

Once a set of points have been labeled as features, a different set of operations is applied that compute feature *descriptors* for all detected features. Feature descriptors are often represented as vectors or matrices, and represent the local appearance of the signal in a color image (ORB) [311] or a laser scan (FLIRT) [361], (FALKO) [159]. Recently, there has been significant work on using deep learning to improve the feature extraction process, mostly using color images. These efforts include learning latent representations of descriptors [410, 302].

2.4.3 Correspondence Calculation

Given a list of newly detected features, along with their descriptors and locations, these features can be compared against previously observed features in order to establish re-observation. We call this step correspondence calculation. The goal is to detect re-observations while avoiding false positive identifications, since their inclusion into the inference process can create large errors in location estimate. In the most basic setting, where the task is to compare, for example, two dense point clouds, then correspondences must be calculated between (almost) every point in the previous frame to every point in the current frame. This can be very expensive for raw data, and there are extensions based on spatial partitions, like the kd-tree [293, 416], that help minimize this cost. Generally, systems also attempt to find correspondences between the current data and data from multiple frames prior. This is known as 'loop closure'. There are several ongoing research efforts investigating methods to search efficiently for potential loop closures among past data instances in order to meet robots' real-time operating constraints [391, 204, 179, 254].

2.4.4 Pose Optimization

In this step, we use the correspondences established in the previous step to finish defining one of the localization problems. In a Kalman filter or particle filter, correspondences would checked against the map to determine the measured state (Kalman filter) or the probability of being in a given state (a particle in a particle filter). In pose-graph, the correspondences would create new cost functions that would be added to the optimization problem. Generally, this step involves adding any new observations (Kalman filter, particle filter) or cost functions (pose-graph), and then running inference (filters) or non-linear optimization (pose-graph) to get the maximum likelihood current pose or trajectory. Please see section 2.2 for a more comprehensive overview.

2.4.5 Map Curation

This step is typically the last step before processing the next sensor data (camera image, laser scan, etc.). During this step, the map is updated with the new information sensed during the current time step. Maps may be updated in three different ways: addition, revision, and deletion. Additions are most common when the robot begins its deployment, and usually consist of initializing new elements within the map's data structure. These may be raw data, such as images or laser scans, but are more often stored as the memoized outputs of feature detectors or descriptors, or a collection of parametric models that have been fit to the data. Map representations tend to grow without bound either with respect to the number of features observed or the volume of space explored. In both cases, as the robot stops exploring completely new regions of its operating environment, additions decrease and eventually stop and the memory footprint of the map stops growing.

Revisions occur during both the exploration phase, as well as during future deployments when the map already represents all of the navigable area. Generally, revisions and map updates are some of the most challenging processes to model correctly, since they can be heavily influenced by both dynamics in the environment as well as observation errors. The method of map update, more so than the underlying representation of the map itself, is what explains the majority of the diversity in mapping research. Deletions, unlike additions and revisions, a are typically not common. They usually represent a major event that drastically affects the robot's belief about the world, although they can also technically occur as the result of a revision, for example if two line segment features are determined to be two different observations of the same wall.

2.4.6 Learning and Other Offline Processes

Beyond the core processes that run in every SLAM system, there are a number of auxiliary processes that are not required, but nevertheless frequently helpful. These can include post-hoc corrections to maps and models, such as Human-in-the-Loop SLAM [255] or semantic labeling [88]. These tools generally help build higher quality models of the environment more efficiently.

There are tools roboticists can use to understand different types of failures and diagnose their provenance. For example, the Laser2Vec system [254] creates latent representations of raw data and stores it in a database where it can then be provided to answer a range of general similarity queries in order to aid data exploration. Moreover, some problems are easy to detect and diagnose but very difficult to solve. Because of this, many researchers have proposed more expansive visions for human-robot teams, where robots operate in a largely autonomous manner, but can engage human-based resources should they encounter such a problem. For example, during navigation [65].

Last, and perhaps most significantly, many of the processes described in this section contain sub-problems that may be improved by learning. Although the back-end algorithms are well-motivated and stand on theoretically solid ground, most of the front-end steps do not share this property. Therefore, it is plausible that learning different functions to perform front-end subroutines which have previously been largely hand-crafted may lead to better performance overall. Previously, we highlighted this phenomenon in the context of feature detection and description, but it is also true for tasks like correspondence calculation [325] or scene similarity [401]. Last, one additional auxiliary area in which learning has been particularly helpful has been failure prediction. This includes for SLAM-adjacent tasks such as [89, 101]. However, currently there is no work we are aware of that has learned to predict SLAM algorithm performance.

2.5 Decision Making and SLAM

So far, SLAM systems have been presented as passive systems that simply do their best to localize and build maps given data streams, with no internal agency to affect the contents of the data stream. This is the most modular view of these systems and the view which this thesis adopts, where SLAM systems operate in a highly integrated but causally detached manner from the rest of the robotics stack. In this section, we present several common problem formulations that break this assumption and introduce an element of decision making into so-called 'integrated' or 'active' SLAM systems. From a theoretical perspective, these formulations allow potentially greater performance while marginally increasing computational burden, but their main drawback is often a significant increase in software complexity and potentially a less robust system.

2.5.1 Exploration

Exploration of a new environment, whether explicit or implicit, human guided or fully autonomous, is a necessary problem for all mobile robots unless they are given a map predeployment. Here, we focus on explicit algorithms for exploration [287]. These algorithms take a partially complete map and the robot's current location estimate and produce a new waypoint or drive goal that is generally designed to further complete the map. The very earliest methods for exploration optimized for covering the unknown areas as quickly as possible, usually using measures of information gain in a greedy manner [395, 337]. However, later methods recognized the need to balance map coverage with both map accuracy and exploration time[172]. Therefore, more complicated 'localizability' metrics were introduced, based on the estimate of the lowest vehicle pose covariance attainable from a given location, and were used to balanced evaluation of alternative motion actions [216].

Other methods reject a more custom metric and instead estimate probability of a "good enough" map as the probability that the Kullback-Leibler divergence between the true posterior and our particle-based approximation is smaller than a given threshold [54]. Regardless, many of these greedy methods are based on the assumption that the environment is represented by a completely unexplored occupancy grid and that information gain is the driving factor. Moreover, although these methods want to maximize information gain, they can only do so over a set of sampled target poses.

Rather than a greedy approach, some proposals suggest optimizing the entire trajectory. Due to the accumulation of localization errors over time as the robot explores, it can often be advantageous to re-observe some features in the environment. These re-observations, often called 'loop closures' although they need not occur due to a loop in the trajectory, provide valuable data in the form of additional cost functions within the optimization problem. If these re-observations can be established with bounded error, they serve to bound to accumulation of localization drift that occurs between re-observations.

A natural problem then is how to optimally trade off between spending time re-observing known parts of the environment in order to reduce uncertainty versus spending time observing new parts of the map in order to complete exploration efficiently. Unfortunately, this problem is usually an instance of a partially observable Markov decision process (POMDP) [221]. One such work avoids solving for the optimal sequence of actions exactly, which typically involves enormous computational cost, by using a version of breadth first search (BFS) and some additional bookkeeping to generate high-quality approximations [329]. Other methods plan actions specifically for loop closure, often called 'active' loop closure [62, 58].

Last, there have several attempts to learn policies for active loop closure during exploration. These include both end-to-end exploration policies [172, 60], as well as more hierarchical and modular systems [56]. Unfortunately, these systems are yet to surpass their non-learning counterparts in any meaningful metric, and in most cases are significantly less performant.

2.5.2 Navigation

Even completed maps offer many opportunities for more tightly coupled planning and localization. Of course, path, route, or trajectory planning requires a map, but here we focus on situations where efficiency in navigating from one point to another is not the only planning objective. For example, path planning that takes localizability into account in belief space [128] via extensions to common path planning frameworks; here they use BRMs[292], an extension or PRMs[163], which require linear belief updates. They use UKF posteriors. Localizability has also been modeled using terrain complexity as a cost in A* path planning for an AUV [196], within the belief update in a particle filter [355], and using potential fields [327].

2.5.3 Disambiguation

Unfortunately, there are many scenarios when a robot's localization estimate has degraded enough such that it cannot disambiguate between two or more hypotheses supported by its current observations. In such a case we might say that the robot is 'lost' even if its belief about its location is not completely uniform. It is possible in these scenarios that the robot will encounter the right stimuli in the right order so that it may recover an accurate and precise belief about its location simply by moving about randomly. However, it is usually the case that planning for disambiguation can significantly speed up this process. This can be done actively, via belief space planning [313], or passively, such as through adjusting parameters of a particle filter online [227]. When done actively through planning or a pre-determined routine this is sometimes referred to as 'recovery' [405].

2.5.4 Coordination

Many environments and applications require multiple robots to explore and map in an economical or timely manner. All of the problems previously highlighted apply to these scenarios, but there are several new issues that arise specifically when dealing with multi-agent systems. Roughly speaking, these are issues of coordination involving questions about if and when to send data between robots, calculation of navigation goals in order to maintain a physical structure, such as line-of-sight connectivity, and how to best integrate data from different robots into a single map. Finally, there is a robust community of research on swarm robotics and decentralized systems applied to SLAM [19, 273, 164], which we omit from further discussion here as in many ways the research challenges fundamentally differ.

2.5.4.1 Communication

Many multi-agent SLAM systems require explicit communication between agents. However, constraints such as bandwidth or network connectivity can make these problems challenging and have promoted research to, for example, efficiently communicate information about loop closures in multi-robot settings [282, 377, 111].

2.5.4.2 Formation

One challenging component of many communication constraints is that they often arise from physical constraints, such as maintaining line of sight connections between agents. Thus, several efforts have been directed towards formation control [24, 305, 230], with the idea being that maintaining particular physical parameters while operating simplifies other higher-level decision-making.

2.5.4.3 Loop Closure Representation

There are also significant challenges in understanding how to adapt single-robot loop closure detection schemes to multi-robot settings [375, 322], and how to use these loop closures once they have been detected. Related to the problem of map merging faced by proponents of local maps, maintaining maps generated by multiple agents is not only a challenging perceptual problem, but also contains elements of decision making.

2.6 Decision Making Under Uncertainty

There are several paradigms for reasoning under uncertainty. One of the most effective and robust is that of the Markov decision process and its extension, the partially observable Markov decision process. Robots, like humans, operate in a partially observable world. Very few phenomena of interest to a mobile robot are fully observable, and it is hard to overstate the significance of this complication. Therefore, almost all robotic systems that engage in sequential decision making must either model partial observability explicitly, or inherit the risks that come with assuming fully observable state. Planning formalisms for modeling partial observability explicitly are luckily common, and MDPs, like many other planning models, have been extended to deal with this complication, although at a significant cost to computation. Here, we give a brief overview of their theory and solution techniques, particularly as they relate to their use on resource-bounded, embodied systems, such as robots.

2.6.1 Markov Decision Processes

A Markov decision process (MDP) is a model for reasoning in fully observable, stochastic environments [29], defined as a tuple $\langle S, A, T, R, d, \gamma \rangle$, where:

- S is a finite set of states;
- A is a finite set of actions;
- $T : S \times A \times S \to [0, 1]$ represents the probability of reaching a state $s' \in S$ after performing an action $a \in A$ in a state $s \in S$;

- $R : S \times A \times S \to \mathbb{R}$ represents the immediate reward of reaching a state $s' \in S$ after performing an action $a \in A$ in a state $s \in S$;
- $d : S \to [0, 1]$ represents the probability of starting in a state $s \in S$ and
- $\gamma \in [0, 1)$ is the discount factor.

A solution to an MDP is a policy $\pi : S \to A$ indicating that an action $\pi(s) \in A$ should be performed in a state $s \in S$. A policy π induces a value function $V^{\pi} : S \to \mathbb{R}$ representing the expected discounted cumulative reward $V^{\pi}(s) \in \mathbb{R}$ for each state $s \in S$. An optimal policy π^* maximizes the expected discounted cumulative reward for every state $s \in S$ by satisfying the Bellman optimality equation

$$V^*(s) = \max_{a \in A} \sum_{s' \in S} T(s, a, s')[R(s, a, s') + \gamma V^*(s')]. \tag{2.14}$$

The primary objective when defining an MDP is to capture all of the information and relations between variables essential for decision-making, and not more. As MDPs built by hand can quickly become cumbersome to maintain, inefficient to solve, and even incorrect or misleading in terms of their transition or reward functions. MDPs are best used when there are many factors and uncertainty that affect a range of decisions, but the decision outcomes are easily assessed by one or two measures. For example, describing a warehouse with uncertain supplies and demands, but where revenue or profit is easily measured is a potential application.

To solve MDPs exactly, one can use either linear programming [218] or take advantage of the recurrence relation and apply it iteratively in what is know as value iteration, as originally suggested by Bellman [29]. Both approaches end up being forms of dynamic programming. MDPs unfortunately do not scale very well as their models get larger, particularly the size of the state space. Thus, considerable attention has been paid to their approximate solution.

Techniques for doing so generally adopt one of three approaches. First, there are *approximate solvers* that use dynamic programming methods based on value or policy iteration [34, 291] and linear programming [118, 280, 290, 217]. Second, some methods compute *partial policies* on a subset of the ground states and re-plan if the agent encounters a state for which the partial policy is undefined [333, 286]. FF-Replan [402], a remarkably simple yet effective algorithm for planning in MDPs, works by determinizing an MDP (i.e. removing some or all of the stochasticity from the model), constructing a plan in the determinized model, and re-planning if the agent reaches an unexpected state. Third, optimal policies are computed on *abstractions* of the original problem, where there is a surjective mapping from the original ground states to the abstract states [191]. Our approach combines insights from both partial policies and abstractions but does not preclude the use of approximate solvers. Using abstractions to reduce the size of a problem is a natural and popular approach to solving large MDPs. The quality of these policies depends heavily on the abstraction scheme, and many abstraction methods have been proposed. Some strict definitions include bisimulation [112], statistical bisimulation [102], and bounded MDPs [80]. Abstractions based on homomorphisms [303, 43] and generic change of basis have also been proposed [404].

2.6.1.1 Common Variants

MDPs are a state transition representation, and many types of specialized MDP solvers have been proposed to support problem solving in many different, but common, contexts. Here, we briefly cover a small number of the most prominent, before reviewing more in-depth the subclass of variants that is perhaps most useful in robotics. Constrained MDPs [12] solve for a value-maximizing policy subject to some other constraints, such as constraints on occupancy measures or on state visitation probability. Multi-objective MDPs [388] are similar, but instead of optimizing a single value function, they optimize several, each representing a different objective. Decentralized MDPs [33] represent multi-agent problems in which agents are operating in a stochastic environment and seek to maximize reward, but do not collaborate in order to plan their policies and thus do so in a decentralized fashion. Reward-uncertain MDPs [304] represent uncertainty over the reward function, usually parameterized by a probability distribution over possible reward functions. Partially abstract MDPs [261] are a lossy compression of MDPs used for approximately solving for policies online. They are designed to increase efficiency while minimally affecting policy quality. Last, mixed-observable MDPs [271] can model scenarios where some state factors are fully observable and others are not. They represent a sort of half-way point between MDPs and partially observable MDPs (POMDPs).

Since their introduction, MDPs and their variants have enjoyed widespread application to many problems in AI, including logistics [324], autonomous driving [389], and adversarial modeling [374, 252]. Recently, constrained MDPs have been proposed to address more complex and holistic problems involving potential negative side effects of agent behavior [346, 347, 348, 251, 260], and with this, interest in the explainability of such systems has also grown [257, 214].

2.6.2 Partially Observable Markov Decision Processes

A *partially observable Markov decision process* (POMDP) is a formal decision-making model for reasoning in partially observable, stochastic environments [154]. A POMDP is described as a tuple $\langle S, A, T, R, \Omega, O, b_0, \gamma \rangle$, where S, A, T, R, and γ are as in an MDP and

- Ω is the set of observations of the agent;
- $O : S \times A \times \Omega \to [0, 1]$ is the observation function that maps each state $s \in S$ and action $a \in A$ to the probability of emitting observation $\omega \in \Omega$ and
- b_0 is an initial belief state.

In a POMDP, the agent does not necessarily know the true state of the world at any given time. Instead, the agent makes noisy observations that reflect its state and action. To represent its uncertainty, the agent maintains a belief state $b \in B$, a probability distribution over all states, where B is the space of all belief states. Initially, the agent begins with an initial belief state $b_0 \in B$. After performing an action $a \in A$ and making an observation $\omega \in \Omega$, the agent updates its current belief state $b \in B$ to a new belief state $b' \in B$ using the belief state update equation

$$b'(s'|b, a, \omega) = \alpha O(a, s', \omega) \sum_{s \in S} T(s, a, s') b(s), \tag{2.15}$$

where α is the normalization constant $\alpha = Pr(\omega|b, s)^{-1}$. Exact POMDP solutions are EXP-hard [201] and in many cases undecidable [211]. Thus, approximate solutions are the norm. Both point-based value-iteration (PBVI) [285] and solutions the represent the POMDP policy as a finite state controller (FSC) [289], are popular approximate methods.

A policy π of a POMDP or MDP, for that matter, is often represented as a *finite-state controller* (FSC) of a fixed size. Formally, an FSC is a tuple $\pi = \langle Q, \lambda, \eta \rangle$. Q is a set of nodes representing a region of the belief space B. $\lambda : Q \rightarrow A$ is an action function that maps a node $q \in Q$ to an action $a \in A$. $\eta : Q \times \Omega \rightarrow Q$ is a transition function that maps a node $q \in Q$ and an observation $\omega \in \Omega$ to a successor node $q' \in Q$. At each time step, the agent begins in a node $q \in Q$ associated with its current belief state $b \in B$, performs an action $a \in A$ following the action function λ, and ends in a successor node $q' \in Q$ following the transition function η. An FSC π induces a value function $V^\pi : Q \times S \rightarrow \mathbb{R}$ that represents the expected cumulative reward of a node $q \in Q$ and a state $s \in S$, and an optimal FSC π^* maximizes this value function. Note that solution methods that use FSCs of a fixed size may not be optimal as they restrict the space of policies [46].

2.7 Machine Learning for SLAM

Machine learning, and in particular deep learning, has become popular in SLAM systems in order to detect, describe, or reject features from exteroceptive sensors or compute similarity between features as a component of SLAM front ends. While there are some lines of work that use neural networks to learn mappings directly from raw sensor data to pose, we will focus on efforts confined to the front end as the end-to-end systems seem generally not promising and are perhaps somewhat misguided as a whole, at least for most robotics applications where exploration and generalization are required. Naturally, machine learning techniques are most applicable for scenarios where we need to measure similarity between two data items, but there is no a priori obvious definition for similarity. Physical objects have sizes, shapes, colors, and masses, among many other attributes, and there is no obvious mathematical way to combine measurements of these attributes into a singular notion of similarity. For example, it is not clear whether a blue ball should be considered more similar to a red ball or a blue cube. Moreover, although we have many heuristics for defining the appearance of an image, either as a whole or some subset of the image, these heuristics are not based on unassailable truths about the physics of our world. They are instead based mostly on intuition and on concepts that are easy to understand and program in order to process data. These issues are present in most low-level signal processing tasks for robots, including feature extraction, which includes detecting feature locations and constructing a representation in data of the feature's appearance, as well as calculating similarity between features in order to determine correspondence (data association) and deciding when to reject features because of the surrounding context in the input (for example using features from a cloud is not likely a robust strategy). In the following subsections we will provide a brief overview of different applications of machine learning within SLAM front ends, though almost all of the applications have original implementations that rely much more heavily on hand-crafted signal processing solutions.

2.7.1 Feature Detection and Description

Feature detection is the process of identifying salient points in the input space, for example (x, y) pixel coordinates in an image or the corresponding re-projected 3D coordinates. A salient point is a point that is likely to be both re-observable (i.e. it is not due to a transient object like a cloud, puddle, or pedestrian) as well as distinct in that it is not easily mistaken for another feature. The first feature detectors were designed to detect edges, corners, or blobs in images using hand-crafted image processing techniques, frequently involving convolution with specially designed kernels (see the review paper by Li et al. for more details [195].) Some of these systems performed both feature detection and description simultaneously.

Recently, there has been a large body of work on using machine learning to replace hand-crafted methods for feature detection. Beginning in 2004, researchers began using different custom features along with statistical machine learning tools like logistic regression [220, 306, 215], spectral clustering [17], random forests and other boosting techniques using decision trees [199, 92, 93], and support vector machines [392] to identify locations of edges or other salient points in images. These methods could adapt to new data sets and addressed several longstanding issues in feature detection such as dealing more robustly with textured surfaces. However, they still relied on hand-crafted or intuitive pre-processing of the image. Very recently,

Feature description is the process of computing a data representation, usually a vector or a matrix though occasionally a collection of heterogeneous data items, of a section of the signal, usually centered on a point identified by a feature detector. Feature descriptors are designed to capture difference across a range of signal attributes (for example size, shape, color, etc.) such that descriptors for features observed at different times during the deployment can be compared to identify when a feature is being observed for the first time or re-observed. The first widely adopted descriptors were based on histograms of gradients (HoGs) taken at various orientations around the feature location, followed closely by so-called binary descriptors that constructed feature vectors using sets of carefully chosen pairwise pixel comparisons. See Nashed [253] and Hartmann et al. [126] for more comprehensive coverage.

With the advent of deep convolutional neural networks (CNNs), researchers could offload both the feature description *and* the image pre-processing (what used to be corner detection, for example) to the neural network. This led to the feature detection and description processes being combined in many CNN-based systems. Aiding in this shift was the increasing availability of labeled data that could be used for training, either from RGB-D sensors or from other inference problems like structure from motion [96]. The performance advantages of deep convolutional features for SLAM have been investigated in several proposed systems, including LIFT-SLAM [48], which uses LIFT features [399]; DF-SLAM [161], which uses TFeat features [26] that are trained via a triplet network; DXSLAM [189] which uses HF-Net features [317] consisting of a hierarchical set of features including both local and global descriptors; and SuperPointVO [123], which uses SuperPoint features [87]. Other approaches have opted to train their own feature descriptors specifically for SLAM, also using CNNs [117]. Beyond raw image description, there have also been attempts to include semantic information and train descriptors of scenes given semantic labels of objects [393]. Most systems train both detectors and descriptors using the same network. There have also been proposals to maintain a distinction between the two processes by training independent neural networks for each task [272]. Applying deep convolutional networks to image description, both local and global, is still an active area of research and consensus has not been reached as to the optimal feature descriptors for use in SLAM nor whether those descriptors are also optimal for other related robotics tasks.

2.7.2 Feature Correspondence

Feature correspondence is calculated between feature descriptors to determine if the descriptors, which typically represent points in space relative to the robot, correspond to the same physical entity and thus can be informative as to the relative transformation of the sensor between the observations. This calculation may be done between descriptors that represent signals captured in consecutive frames or between descriptors separated by a large amount of time. In the latter case, identifying such correspondences is often called 'loop closure.' If the descriptors represent an area larger than a single image patch or section of a laser scan, this has also been referred to as 'place recognition', and CNNs have been shown to quite good at identifying whole-image similarity [344], both for other tasks as well as loop closure detection [143, 331, 345].

Originally, correspondence calculations were done using custom metrics defined over the given hand-crafted feature descriptor. However, as learning-based systems have become more popular for generating feature descriptions, this has changed. Because the outputs of learned feature descriptors are generally much less interpretable, researchers have moved away from defining custom similarity functions and are now more likely to use common metrics for high-dimensional vector spaces, like Euclidean distance and cosine distance. For example, some approaches specifically guide the training of their networks so that the output descriptors are well-behaved with respect to Euclidean distance[330, 360] or cosine similarity [247, 144].

Because neural networks have no a priori obligation to produce embedding spaces with straightforward geometric interpretations, many researchers have opted to train secondary networks to evaluate the similarity between neural feature descriptors [407, 124, 254]. Moreover, some systems skip explicit correspondence calculations entirely and attempt to solve the pose registration or homography estimation problem directly using a second network on the output of a previous network that produces feature descriptions [86]. While such approaches are not as brittle as many end-to-end neural SLAM systems, they do share many of the same drawbacks.

2.7.3 Feature Rejection

Feature rejection is a relatively new component in SLAM systems. Feature detectors are often designed to be conservative in that they do not identify regions of interest in a signal unless they are very salient. However, because these detectors often have a very localized view of the signal when deciding whether to tag a region for description, it is extremely difficult to remove or avoid false positives (parts of the signal erroneously identified as being salient) entirely.

Thus, new research on methods for rejecting potentially low-quality features has become more prominent. These methods either exclude certain regions of the signal from examination entirely, for example by using custom learned models [297] or based on semantic segmentation [160], or by rejecting features based on other qualities after detection, for example by using a random forest to select long-track features from images [318], or using optimization to minimize the number of features from a LiDAR scan while preserving the information matrix structure [152]. Overall, feature rejection methods are still relatively unexplored and likely have substantial untapped potential for many SLAM applications.

2.7.4 Common Neural Network Architectures

By far the most influential neural network architecture for SLAM has been the convolutional neural network (CNN) [183, 414, 378]. This is not surprising considering many of the most common sensor inputs robots receive are most naturally represented in 2D matrices, and many CNNs have been designed specifically to detect patterns in 2D data. Moreover, there are many CNN network architectures that exploit the multi-scale signals present in many images in a much more direct and efficient manner than typical feed-forward networks. See the following excellent survey for a more comprehensive view of literature on CNNs [198].

Beyond CNNs, there have been several other influential network architectures that align nicely with common problems in robotic perception. Perhaps most well known is the autoencoder [314]. Autoencoders are often used to learn representations of data in an unsupervised manner, with the primary benefit being that they are free to exploit many non-linear patterns in the data that traditional transformation or dimensionality reduction techniques, like principal component analysis, cannot. While autoencoders are often the tool of choice to learn representations of data, learning similarity functions *between* data is also important. In this area, Siamese networks [47] and triplet networks [138] have become essential tools. These networks again allow the training of non-linear similarity measures in a very low-supervision manner, and form the backbone of most non-linear, neural-based similarity measures for robotic perception.

Despite the large number of applications of machine learning, and deep learning in particular, to many processes in the SLAM front end, there have been relatively few applied to more meta-level problems and processes in SLAM, such as those discussed at length in Chapters 7-9. For these problems which often involve sequential reasoning it is likely that other architectures that handle time more natively, such as recurrent neural networks (RNNs) [99, 137, 67] or Transformers [373], will eventually play an important role. However, we do not cover their applications to such problems in this thesis.

CHAPTER 3
CURATING LONG-TERM VECTOR MAPS

One challenging aspect of long-term autonomy in a human environment is robustness to changes in the environment. Many approaches have been proposed to reason about a changing environment, including estimating the latest state of the environment [379, 176], estimating different environment configurations [38, 182], or modeling the dynamics of the environment [42, 316, 363, 18, 386]. However, in large environments, it may not be feasible to make the periodic observations required by these approaches. Therefore, an alternative approach is to model observations in human environments as arising from three distinct types of features [40]: Dynamic Features (DFs) or moving objects such as people or carts; Short-Term Features (STFs) or movable objects such as tables or chairs; and Long-Term Features (LTFs) which persist over long periods of time, such as office walls or columns. Episodic non-Markov Localization (EnML) [40] simultaneously reasons about global localization information from LTFs, and local relative information from STFs. A key requirement of EnML is an estimate of the Long-Term Vector Map: the features in the environment that persist over time, represented in line segment or vector form (Fig. 3.1).

This chapter introduces an algorithm to build and update Long-Term Vector Maps indefinitely, using observations from all deployments of all the robots in an environment. This algorithm filters out observations corresponding to DFs from a single deployment using a signed distance function (SDF) [73]. Merging the SDFs from multiple deployments then filters out the short-term features. Remaining observations correspond to LTFs, and are used to build a vector map via robust local linear regression. Uncertainty estimates of

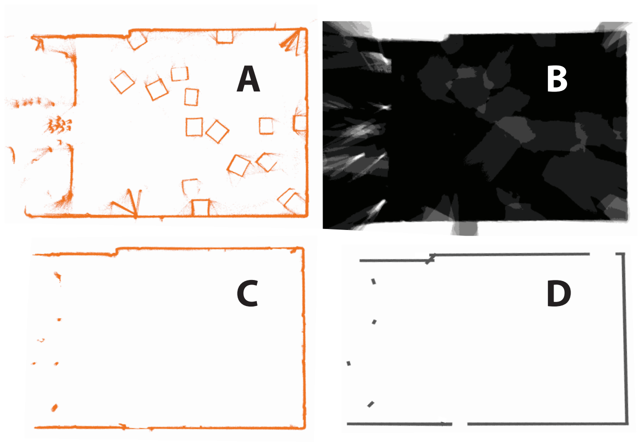


Figure 3.1: Observations at different stages of the LTVM pipeline. In alphabetical order: raw data from all deployments, weights computed by the SDF, filtered data, final LTVM.

the resultant Long-Term Vector Map are calculated by a novel Monte Carlo uncertainty estimator. The algorithm thus consists of the following steps:

1. *Filter:* Use the most recent observations to compute an SDF and discard points based on weights and values given by the SDF.

2. *Line Extraction:* Use greedy sequential local RANSAC [104] and non-linear least-squares fitting to extract line segments from the filtered observations.

3. *Estimate Feature Uncertainty:* Compute segment endpoint covariance estimates via Monte Carlo resampling of the observations.

4. *Map Update:* Add, merge, and delete lines using a de-coupled scatter matrix representation [39].

This method takes advantage of the robust filtering provided by the SDF while avoiding dependency on a discrete world representation and grid resolution by representing LTFs as line segments in $\mathbb{R}^2$. The benefits of this approach are illustrated in several experiments that find vector maps constructed via SDF filtering comparable or favorable to occupancy grid based approaches along several metrics.

The problem of long-term robotic mapping has been studied extensively, with most algorithms relying on one of two dominant representations: occupancy grids [97, 239] and geometric or polygonal maps [57, 413]. Recently, work towards metric map construction algorithms that are able to cope with dynamic and short-term features has accelerated. Most of these approaches fall into one of four categories: dynamics on occupancy grids, latest state estimation, ensemble state estimation, and observation filters.

One common approach models map dynamics on an occupancy grid using techniques such as learning non-stationary object models [42] or modeling grid cells as Markov chains [316, 363]. Alternatively, motivated by the widely varying timescales at which certain aspects of an environment may change, some approaches seek to leverage these differences by maintaining information relating to multiple timescales within one or more occupancy grids [18, 386]. Other approaches estimate the latest state of the world, including dynamic and short-term features. Dynamic pose-graph SLAM [379] can be used in low-dynamic environments, and spectral analysis techniques [176] attempt to predict future environment states on arbitrary timescales.

Instead of estimating solely the latest state, some approaches estimate environmental configurations based on an ensemble of recent states. Temporal methods such as recency weighted averaging [38] determine what past information is still relevant, and other techniques such as learning local grid map configurations [182] borrow more heavily from the dynamic occupancy grid approach. Another approach filters out all observations corresponding to non-LTFs, resulting in a "blueprint" map. Previous algorithms have had some success filtering dynamic objects, specifically people [121], but have struggled to differentiate between STFs and LTFs. Furthermore, all of the methods mentioned above rely on an occupancy grid map representation, whereas this method produces a polygonal, vector map that does not rely on discretization of the operating environment.

3.1 Long-Term Vector Mapping

Long-term vector mapping runs iteratively over multiple robot deployments, operating on the union of all registered laser observations from the given deployment, aligned to the same frame. We call these unions composite scans (see Fig. 3.2) and denote them $C = \cup_{i=1}^{N} S_i$, where S_i is a single laser scan. Composite scans are processed in batch after each deployment, and may be generated via Episodic non-Markov Localization [40] or a similar localization algorithm.

After each deployment, a short-term signed distance function (ST-SDF) given by a set of weights W' and values V' is computed over the area explored by the robot by considering the composite scan C. SDFs are typically used to implicitly represent surfaces, and are essentially a paring of two different weighted sums: one related to the consistency with which particular observations are made (weights) and the other related to the data contained in the observation (values). Splitting the aggregation of data into these two independent sums allows more robust filtering techniques as reasoning can be done conditioned on two (aggregate) measurements instead of one. The ST-SDF is then used to update the long-term SDF (LT-SDF), given by W and V, which aggregates information over all previous deployments. The updated LT-SDF is denoted W^* and V^*, and is used to determine a filtered composite scan $C' \subset C$, containing observations corresponding exclusivley to LTFs. We call this process SDF-filtering. Note that after only one deployment, it is not possible to distinguish STFs from LTFs. However, as the number of deployments increases, the number of observations corresponding to STFs in C' approaches zero.

The next step in our algorithm is feature (line) extraction. Line extraction does not rely on any persistent data, using only the filtered composite scan C' to extract a set of lines L'. Each $l'_i \in L'$ is defined by endpoints p_{i_1} and p_{i_2}, a scatter matrix S_i, a center of mass $p_{i_{cm}}$, and a mass M_i. Uncertainties in the endpoints of each line segment are computed by analyzing a distribution of possible endpoints generated via Monte Carlo resampling the initial set of observations and subsequently refitting the samples. For a given line l_i the

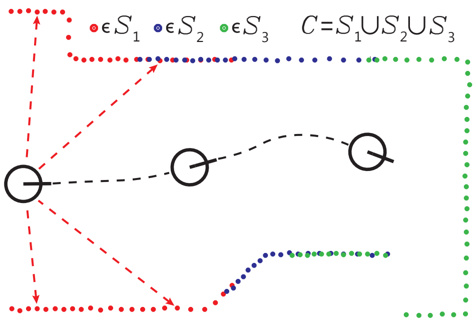


Figure 3.2: Composite scan. Each dot represents an observation $c \in C$. Composite scan C is constructed by aligning scans $S_1 \dots S_N$ to the same frame. Alignment could be done by Episodic non-Markov Localization [40] or a similar localization algorithm. We assume this step is already complete.

uncertainty estimation step takes as input the endpoints p_{i_1} and p_{i_2} and a set of inliers I_i, and produces covariance estimates Q_{i_1} and Q_{i_2} for these endpoints.

The long-term vector map is updated based on the newest set of extracted lines and the current SDF. Similar lines are merged into a single line, obsolete lines are deleted, and uncertainties are recomputed. Thus, this step takes the results from the most recent deployment, L', as well as the existing map given by L, W^*, and V^*, and outputs an updated map, L^*. Fig. 3.3 presents an overview of the algorithm as it operates on data from a single deployment.

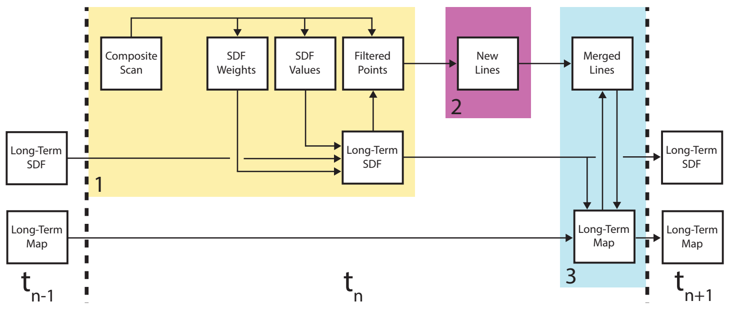


Figure 3.3: Flow of information during processing of a single deployment, deployment n. Boxes 1, 2, and 3 correspond to SDF filtering, line finding and uncertainty estimation, and map updating, respectively.

3.2 SDF-based Filtering

Let C be a composite scan, where $c \in C$ is a single observation providing depth ρ and angle α with respect to the laser's frame and the robot pose $p = (x, y, \theta)$ at which ρ and α were recorded. That is, $c = [\rho, \alpha, p]$. The filtering problem is to determine $C' \subset C$ such that all $c' \in C'$ originate from LTFs and all $c \in C \setminus C'$ originate from STFs and DFs.

Determining C' is a three-step process. First, we construct a ST-SDF over the area observed by the robot during the deployment corresponding to C. Next, we update the LT-SDF based on the ST-SDF. Finally, we use the updated LT-SDF to decide which observations correspond to LTFs.

3.2.1 SDF Construction

SDFs operate over discretized space, so we create a grid of resolution q containing all $c \in C$. Each pixel in the SDF maintains two measurements, a value d_0 and a weight w_0. For every observation $c \in C$, all pixels that lie along the given laser ray update their respective values according to $d_0 = \frac{w_0 d_0 + w d}{w_0 + w}$ and $w_0 = w_0 + w$, where d_0 and w_0 are the current distance and weight values, respectively, and d and w are the distance and weight values for the given reading c. d and w are given by

$$d(r) = \begin{cases} \delta & \text{if } r > \delta \\ r & \text{if } |r| \leq \delta \\ -\delta & \text{if } r < -\delta \end{cases}, \quad w(r) = \begin{cases} 1 & \text{if } |r| < \epsilon \\ e^{G} & \text{if } \epsilon \leq |r| \leq \delta \\ 0 & \text{if } |r| > \delta, \end{cases} \tag{3.1}$$

where $G = -\sigma(r - \epsilon)^2$ and r is the signed distance from the range reading to the pixel, with pixels beyond the range reading having $r < 0$ and those in front having $r > 0$. σ and ϵ are parameters that depend on the accuracy of the sensor. Pixels that are located along the ray but are more than δ beyond the detection point are not updated since we do not know whether or not they are occupied. Fig. 3.4 illustrates a single pixel update during SDF construction. Note that this process is parallelizable since weights and values for each pixel may be calculated independently. Thus, the SDF construction step, outlined in Algorithm 1, runs in time proportional to $|C|$.

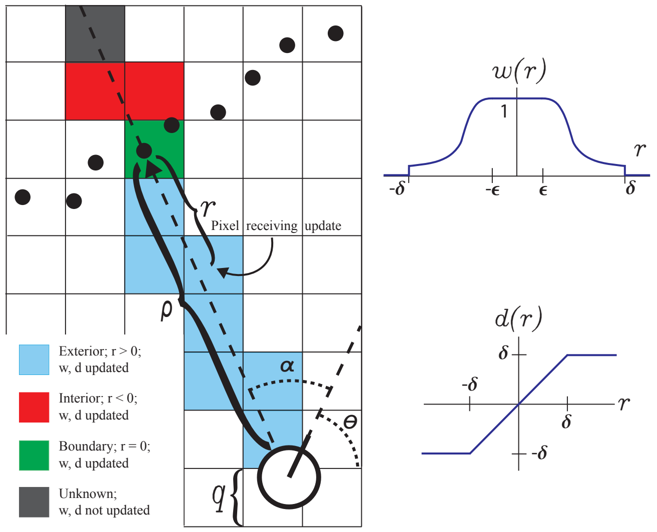


Figure 3.4: SDF construction from a single laser ray. Pixels along the laser ray are updated if they are free, or if they are just beyond the obstacle. Over many ray casts, pixels may be marked as belonging to more than one category (boundary, interior, exterior) due to sensor noise. The SDF's main advantage is that it ignores erroneous readings.

The intuition for the weight and value functions comes from two observations. First, capping $|d(r)|$ by δ helps keep our SDF robust against anomalous readings. Second, $w(r)$ follows common laser noise models. Other choices for $d(r)$ and $w(r)$ may yield similar results; however, these choices have already been successfully adopted elsewhere [49].

3.2.2 SDF Update

Once the ST-SDF is calculated we normalize the weights:

$$w_{norm} = \begin{cases} 0 & \text{if } \frac{w}{w_{max}} \leq T_1, \\ 1 & \text{otherwise.} \end{cases} \tag{3.2}$$

Here, w_{max} is the maximum weight over all pixels and T_1 is a dynamic feature threshold. The LT-SDF is then updated as the weighted average of the ST-SDF and the LT-SDF, i.e. $W^* = \textsc{WeightedAverage}(W, W')$. Pixel weights loosely map to our confidence about the associated value, and values are an estimate for how far a pixel is from the nearest surface.

3.2.3 SDF Filter

Given an up-to-date SDF, we determine C' using bicubic interpolation on the position of each observation c and the updated LT-SDF. The filtering criteria are $c' \in C'$ if $\textsc{BicubicInterpolation}(c, W^*) > T_2$ and $\textsc{BicubicInterpolation}(c, V^*) < T_d$, where T_2 is a STF threshold and T_d is a threshold that filters observations believed to be far from the surface of any object. Lines 11-15 in Algorithm 1 detail the filtering process. Thus, after running SDF $\textsc{Filtering}(C, W, V)$, we obtain a filtered composite scan C', used to find LTFs. Additionally, SDF $\textsc{Filtering}$ updates the LT-SDF needed for the map update step.

Intuition for the effect of thresholds T_1 and T_2 is helpful in understanding SDF filtering. Both thresholds act on SDF weights, but T_1 is applied *before* weight normalization, whereas T_2 is applied *post* normalization. T_1 can be thought of as a DF filter, rejecting pixels which contained observations infrequently relative to the highest weight pixel, implying a transient state of occupation. A too-high value of T_1 filters some of the static features due to the non-uniform nature of the laser scan; e.g. some objects (pixels) are observed more frequently than others. A too-low value of T_1 causes objects which are dynamic to be misclassified as static, and thus promotes erroneous feature creation. An extreme case of $T_1 = 0$ marks all pixels as containing static objects and thus prevents the map update step from deleting features.

T_2 can be thought of as the STF filter. After each deployment, the map update step calculates an average of all previous deployments, and decides to accept or reject current observations and features based on T_2. A too-high value of T_2 results in fracture of true LTFs due to noise in observations, since there is a small chance the laser fails to register enough observations of a part of an LTF, causing a failure to pass the T_1 threshold on

Algorithm 1 SDF FILTERING

```
1: **Input:** Raw composite scan $C$, long-term SDF weights $W$ and values $V$
2: **Output:** Filtered composite scan $C'$, updated SDF weights $W^*$ and values $V^*$
3: $V' \leftarrow$ empty image
4: $W' \leftarrow$ empty image
5: for all range readings $c \in C$ do
6:     $V' \leftarrow$ VALUEUPDATE$(W', c)$
7:     $W' \leftarrow$ WEIGHTUPDATE$(W', c)$
8: $W' \leftarrow$ NORMALIZE$(W')$
9: $W^*, V^* \leftarrow$ UPDATESDF$(V', W')$
10: $C' \leftarrow \emptyset$
11: for all range readings $c \in C$ do
12:     $b_w \leftarrow$ BICUBICINTERPOLATION$(W^*, c)$
13:     $b_v \leftarrow$ BICUBICINTERPOLATION$(V^*, c)$
14:     if $b_w > T_w$ and $b_v < T_v$ then
15:         $C' \leftarrow C' \cup c$
```

some deployments. A too-low value of T_2 causes STFs which are only rarely absent to be classified as LTFs, such as doors which are most often closed.

3.3 Line Extraction

Given C', extracting lines $l_1 \ldots l_n$ requires solving two problems. First, for each l_i, a set of observations $C_i \subset C'$ must belong to line l_i. Second, endpoints p_{i_1} and p_{i_2} defining line l_i must be found. We take a greedy approach, utilizing sequential local RANSAC to provide plausible initial estimates for line endpoints p_1 and p_2. Points whose distance to the line segment $\overline{p_1 p_2}$ is less than T_r, where T_r is proportional to the noise of the laser, are considered members of the inlier set I. Once a hypothetical model and set of inliers with center of mass p_{cm} have been suggested by RANSAC (lines 5-7 in Algorithm 2), we perform a non-linear least-squares optimization using the cost function

$$R = \frac{||p_{cm} - p_1|| + ||p_{cm} - p_2||}{|I|} + \begin{cases} ||p - p_2|| & \text{if } \; t < 0 \\ ||p - p_1|| & \text{if } \; t > 1 \\ ||p'_1 + t(p'_2 - p'_1) - p|| & \text{otherwise} \end{cases} \cdot \tag{3.3}$$

The new endpoints p'_1 and p'_2 are then used to find a new set of inliers I'. $t = \frac{(p-p'_1)\cdot(p'_2-p'_1)}{||p'_2-p'_1||^2}$ is the projection of a point $p \in I$ onto the infinite line containing p'_1 and p'_2. Iteration terminates when $||p_1 - p'_1|| + ||p_2 - p'_2|| < T_c$, where T_c is a convergence threshold.

This cost function has several desireable properties. First, when all points lie between the two endpoints, the orientation of the line will be identical to the least-squares solution. Second, when many points lie beyond the ends of the line segment, the endpoints are pulled outward, allowing the line to grow and the region in which we accept inliers to expand. Last, the $\frac{||p_{cm}-p_1||+||p_{cm}-p_2||}{|I|}$ term allows the line to shrink in the event that the non-linear least-squares solver overshoots the appropriate endpoint. Once a set of lines L' has been determined by running LINE EXTRACTION(C') we complete our analysis of a single deployment by estimating our uncertainty in feature locations.

Algorithm 2 LINE EXTRACTION

```
1:  Data: Filtered composite scan $C'$
2:  Result: Set of new lines $L'$
3:  $L' \leftarrow \emptyset$
4:  while $C'$ not empty do
5:      Propose $p_1, p_2$ via RANSAC
6:      $I \leftarrow$ findInliers($p_1$, $p_2$)
7:      $p'_1, p'_2 \leftarrow$ fitSegment($I$)
8:      while $||p'_1 - p_1|| + ||p'_2 - p_2|| > T_C$ do
9:          $I \leftarrow$ findInliers($p'_1$, $p'_2$)
10:         $p_1, p_2 \leftarrow p'_1, p'_2$
11:         $p'_1, p'_2 \leftarrow$ fitSegment($I$)
12:     $L' \leftarrow L' \cup \overline{p'_1 p'_2}$
13:     $C' \leftarrow C' \setminus I$
```

3.4 Uncertainty Estimation

Given a line l_i, with endpoints p_{i_1} and p_{i_2} and a set of inliers I_i, uncertainty estimation produces covariance estimates Q_{i_1} and Q_{i_2} for p_{i_1} and p_{i_2}, respectively. To estimate Q_{i_1} and Q_{i_2} we resample $c_i \sim I_i$ using the covariance Q_i^c of each range reading. Q_i^c is derived based on the sensor noise model in [283]. In world coordinates Q_i^c is given by

$$\begin{aligned} Q_i^c =& \frac{\rho^2\sigma_\alpha^2}{2}\begin{bmatrix} 2\sin^2(\alpha+\theta) & -\sin(2(\alpha+\theta)) \\ -\sin(2(\alpha+\theta)) & 2\cos^2(\alpha+\theta) \end{bmatrix} \\ &+\frac{\sigma_\rho^2}{2}\begin{bmatrix} 2\cos^2(\alpha+\theta) & \sin(2(\alpha+\theta)) \\ \sin(2(\alpha+\theta)) & 2\sin^2(\alpha+\theta) \end{bmatrix}, \end{aligned} \tag{3.4}$$

where σ_ρ and σ_α are standard deviations for range and angle measurements for the sensor, respectively. Resampling k times, as shown in Fig. 3.5, produces a distribution of p_1 and p_2. We then construct a scatter matrix S from the set of hypothetical endpoints, and compute covariances Q_1 and Q_2 by using the SVD of S.

The Monte Carlo approach, detailed in Algorithm 3, is motivated by the following factors: 1) There is no closed-form solution to covariance for endpoints of a line segment. 2) A piece-wise cost function makes it difficult to calculate the Jacobian reliably. 3) Resampling is easy since we already have I_i and can calculate Q_i^c.

3.5 Map update

Given the current map L, the LT-SDF, and a set of new lines, L', where every $l_i \in L'$ is specified by a set of endpoints p_{i_1} and p_{i_2}, a set of covariance matrices Q_{i_1} and Q_{i_2}, a center of mass $p_{i_{cm}}$, a mass M_i, and a partial scatter matrix S_i, the update step produces an updated map L^*.

The map updates are divided into two steps outlined in Algorithm 4. First, we check if all current lines in the map are contained within high-weight regions. That is, we check that the weight w_{xy} of every pixel a given line passes through satisfies $w_{xy} \geq T_2$. If a

Algorithm 3 FEATURE UNCERTAINTY ESTIMATION

```
1:  Input: Set of new lines $L'$, number of samples $k$
2:  Output: Set of endpoint covariance estiamates $Q'_1$, $Q'_2$
3:  $Q'_1 \leftarrow \emptyset$, $Q'_2 \leftarrow \emptyset$
4:  for all $l'_i \in L'$ do
5:      $P'_1 \leftarrow \emptyset$, $P'_2 \leftarrow \emptyset$
6:      $I_i \leftarrow$ inliers associated with $l'_i$
7:      for $k$ iterations do
8:          $I'_i \leftarrow \emptyset$
9:          for all $c \in I_i$ do
10:             $c' \leftarrow$ SAMPLE$(I_i, c, Q^c_i)$
11:             $I'_i \leftarrow I'_i \cup c'$
12:         $p'_1, p'_2 \leftarrow$ FITSEGMENT$(I'_i)$
13:         $P'_1 \leftarrow P'_1 \cup p'_1$, $P'_2 \leftarrow P'_2 \cup p'_2$
14:     $Q'_1 \leftarrow Q'_1 \cup$ ESTIMATECOVARIANCE$(P'_1)$
15:     $Q'_2 \leftarrow Q'_2 \cup$ ESTIMATECOVARIANCE$(P'_2)$
```

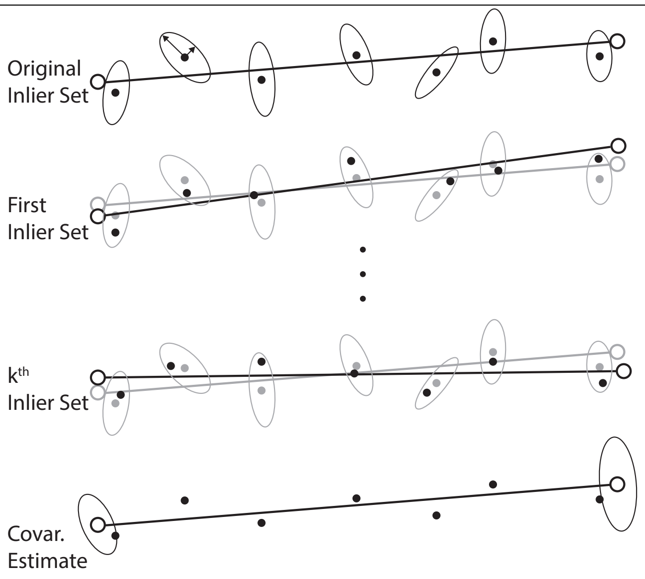


Figure 3.5: Monte Carlo uncertainty estimation of feature endpoints. Given an initial set of observations and their corresponding covariances represented by ellipses, we resample the observations and fit a line k times. The resulting distribution of endpoints is used to estimate endpoint covariance.

line lies entirely within a high-weight region, it remains unchanged. Similarly, if a line lies entirely outside all high-weight regions, it is removed. If only part of a line remains within a high-weight region, we can lower bound the mass of the remaining region by $M' = M\frac{||p'_1 - p'_2||}{||p_1 - p_2||}$, where p'_1 and p'_2 are the extreme points of the segment remaining in the high-weight region (line 11). We then resample M' points uniformly along the line between p'_1 and p'_2, adding Gaussian noise in the perpendicular direction with a standard deviation σ based on the sensor noise model. The points are then fit using the cost function given in (4). The sampling and fitting process is executed a fixed number of times (lines 12-16), and the distribution of fit results is then used to compute covariance estimates for the new line.

The second part of the update involves merging new lines, L', with lines from the current map, L. This process consists of first finding candidates for merging (lines 22-23) and then computing the updated parameters and covariances (lines 24-25). Note that the mapping from new lines to existing lines may be onto, but without loss of generality we consider merging lines pairwise. Because our parameterization uses endpoints, lines which ought to be merged may not have endpoints near one another. So, we project p'_{i_1} and p'_{i_2} from l'_i onto l_j, resulting in $p^{proj}_{j_1}$ and $p^{proj}_{j_2}$, respectively. l'_i and l_j are merged if they pass the chi-squared test:

$$\chi^2 = \Delta l_k^T (Q^{int}_{j_k} + Q'_{i_k}) \Delta l_k < T_\chi, \quad k = 1, 2 \tag{3.5}$$

where $\Delta l_k = p'_{i_k} - p^{proj}_{j_k}$, and $Q^{int}_{j_k}$ is given by a linear interpolation of the covariance estimates for p_{j_1} and p_{j_2} determined by where p'_{i_k} is projected along l_j.

We would like our merged LTFs and their uncertainty measures to remain faithful to the entire observation history. However, storing every observation is infeasible. Instead, our method implicitly stores observation histories using decoupled scatter matrices [39], reducing the process of merging lines with potentially millions of supporting observations to a single matrix addition.

The orientation of the new line is found via eigenvalue decomposition of the associated scatter matrix, and new endpoints are found by projecting the endpoints from the original lines onto the new line and taking the extrema. Thus, after executing MAP UPDATE(L', L, W^*, V^*), we have a set of vectors L^* in $\mathbb{R}^2$ corresponding to LTFs. Table 3.1 displays the major parameters and physical constants needed for long-term vector mapping.

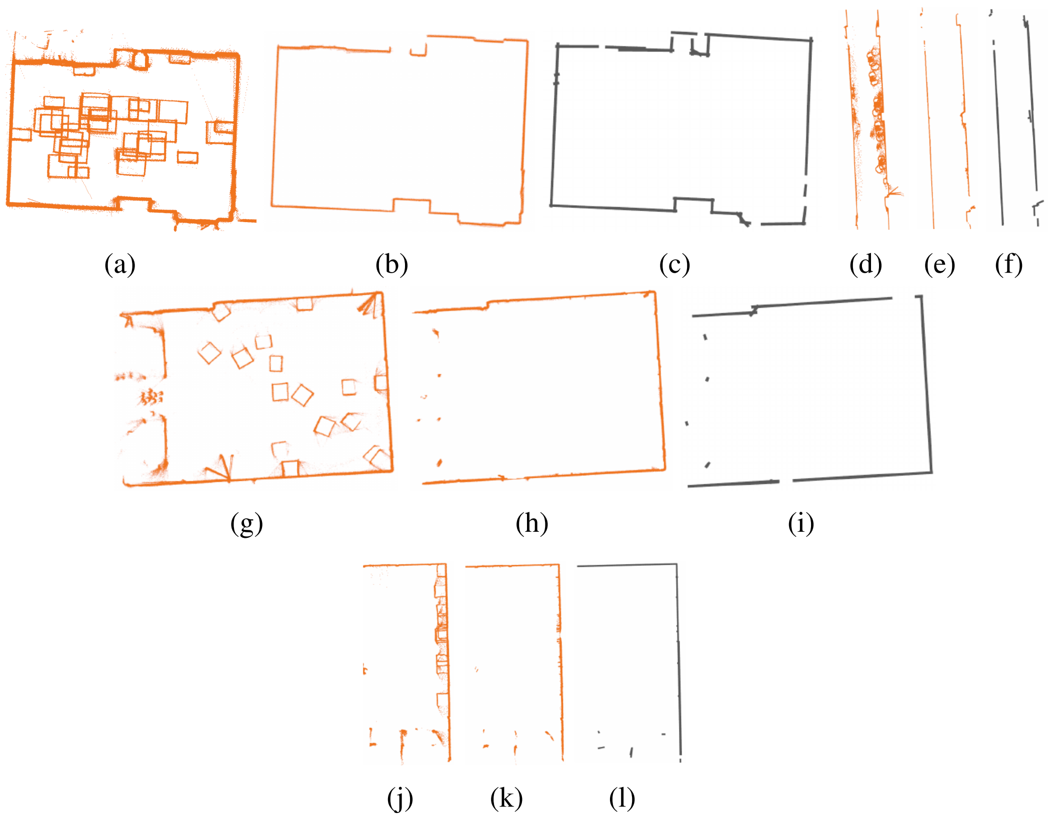


Figure 3.6: Raw data, filtered data, and resultant maps for MIT (a-c), AMRL (d-f), Wall-occlusion (g-i) and Hallway-occlusion (j-l) datasets. Data shown in the left and center columns are aggregates of all deployments and are not stored while the algorithm is operating. The last column is the resultant LTVM which is stored in full, requiring only a few KB. Note the absence of STFs from the final maps, as well as the presence of doorways. In the MIT dataset, some doors were open only once over all deployments. Hallway-occlusion demonstrates the algorithm's robustness to STFs, as it is able to distinguish the column in the hallway even as it is partially or completely occluded on every deployment.

3.6 Results

To demonstrate the effectiveness of our algorithm we mapped 4 different environments, including standard data. Data was also collected from three separate environments, in addition to the MIT Reading Room: AMRL, Wall-occlusion, and Hallway-occlusion, using a mobile robot platform and a Hokuyo UST-10LX laser range finder. The AMRL, Wall, and Hall datasets consist of 8, 5, and 5 deployments, respectively. MIT Reading Room contains 20 deployments. Each deployment contains hundreds of scans, corresponding to hundreds of thousands of observations. Datasets are intended to display a high amount of clutter and variability, typical of scenes mobile robots need to contend with.

Table 3.1: Thresholds and physical constants

Name	Symbol	Domain	Our Value
DF Threshold	T_1	$(0, 1)$	0.2
STF Threshold	T_2	$(0, 1)$	0.95
Line Merge Criteria	T_{χ^2}	> 0	30
Sensor Noise Threshold	T_d	> 0	0.05 meters
RANSAC Inlier Criteria	T_r	> 0	0.12 meters
Line Fit Convergence	T_c	> 0	0.05 meters
SDF Max Value	δ	> 0	0.2 meters

The MIT and AMRL data sets are intended to test the accuracy of our algorithm over larger numbers of deployments. Both rooms contain multiple doors, walls of varying lengths, and achieve many different configurations, testing our ability to accurately identify LTF positions. The Wall- and Hallway-occlusion datasets are intended to measure our algorithm's robustness in environments where LTFs are heavily occluded.

Quantitatively we are concerned with accuracy and robustness, defining accuracy as pairwise feature agreement, where inter-feature distances are preserved, and feature-environment correspondance, where vectors in the map correspond to LTFs in the environment. Vectors should also be absent from areas where there are no long-term features such as doorways. Metric ground truth is established by either measuring wall lengths or hand-fitting the parts of the data we know correspond to LTFs. Robustness refers to a map's ability to deal with large numbers of STFs and lack of degradation over time.

Over all datasets, our approach correctly identifies all 7 doorways (4 in MIT, 2 in AMRL, 1 in Hallway), and correctly ignores all 73 STFs. Using the AMRL and MIT datasets, we find the average difference between pair-wise feature separation in the generated map versus ground truth to be on the order of 2cm. Our line extraction method yields MSE values in the order of 0.0001 meters, while marching squares yields a MSE of roughly 0.0003, about three times as large. Additionally, marching squares yields over 3000 features while LTVM maintains on the order of 30 LTFs. Furthermore, the maps do not degrade over the timescales tested in this paper, with no noticeable difference in av-

Algorithm 4 MAP UPDATE

```
1:  Input: Set of new lines $L'$, set of long-term lines $L$, long-term SDF $W$, $V$
2:  Output: Updated long-term lines $L^*$
3:  for all $l_i \in L$ do
4:      trace along $l_i$
5:      if $l_i$ exists entirely outside high-weight regions then
6:          $L \leftarrow L \setminus l_i$
7:      if $l_i$ exists partially outside high-weight regions then
8:          $L \leftarrow L \setminus l_i$
9:          $P_1', P_2' \leftarrow \emptyset$
10:         for all remaining parts of $l_i$, $\partial l_i$ do
11:             $p_1, p_2 \leftarrow$ GETENPOINTS($\partial l_i$)
12:             for $k$ iterations do
13:                 $I \leftarrow$ REGENERATEINLIERS($p_1, p_2$)
14:                 $p_1', p_2' \leftarrow$ FITSEGMENT($I$)
15:                 $P_1' \leftarrow P_1' \cup p_1'$
16:                 $P_2' \leftarrow P_2' \cup p_2'$
17:             ESTIMATECOVARIANCE($P_1', P_2'$)
18:             $L \leftarrow L \cup \partial l_i$
19: $L^* \leftarrow \emptyset$
20: for all $l_i' \in L'$ do
21:     for all $l_j \in L$ do
22:         $\chi^2 \leftarrow \Delta l_k^T (Q_{j_k}^{int} + Q_{i_k}') \Delta l_k < T_\chi, \quad k = 1, 2$
23:         if $\chi^2 < T_{\chi^2}$ for $k = 1, 2$ then
24:             $l_j \leftarrow$ MERGE($l_j, l_i'$)
25:             $L^* \leftarrow L^* \cup l_j$
26:         else
27:             $L^* \leftarrow L^* \cup l_i'$
```

erage pair-wise feature disagreement between maps generated after one deployment and those considering all deployments. Fig. 3.6 displays qualitative results for all environments.

3.7 Conclusion

This chapter introduced an SDF-based approach to filter laser scans over multiple deployments in order to build and maintain a long-term vector map, as well as several novel sub-components for extracting individual line segments and estimating their uncertainty when viewed at multiple points in the trajectory. We experimentally showed the accuracy and robustness of the generated maps in a variety of different environments and further evaluated the effectiveness of long-term vector mapping compared to more standard, occupancy grid techniques. Moreover, we also showed this method to effectively detect and maintain representations of semantically important objects, like doorways, and geometrically important objects, like pillars and corners. In general, in the context of multi-SLAM systems, SLAM algorithms may be considered black boxes with no particular restriction on their internal form.

Two key principles motivate long-term vector mapping, and they will appear several times throughout the remainder of this thesis. The first is to make as few assumptions about the nature of deployments as possible. Given that the robot is operating in an environment where a laser or other depth sensor is reasonable choice, we try to place no further restrictions on the model, such as on the robot's dynamics, the homotopy of the trajectory, or geometric relations of different features. Second, it is important that this system can leverage computation that is likely to already exist in the stack, allowing this system to limit its marginal computational cost and its disruption of the existing architecture.

Fundamentally, this method is a form of outlier rejection, which is an incredibly common problem in robotic perception as well as many other fields of AI. The necessity of designing outlier rejection methods, rather than treating all data points equally, is a fundamental consequence of our inability to precisely model sensor noise across the entirety of potential sensing conditions. This creates two operating regimes: one in which we know the relative likelihood of different signals and can thus robustly estimate maximum likelihoods in the presence of signal errors, and one in which we cannot. Unfortunately, we do not yet have systematic methods for determining which regime we are operating in at a given time, and thus we approximate this knowledge with filters. At a meta level, later chapters will show how multi-SLAM systems, through a variety of separate techniques, can also be thought of as a type of filtering technique wherein the outputs of different SLAM front-ends are filtered to curate higher quality sets of features and more accurate localization.

CHAPTER 4

LOCALIZATION UNDER TOPOLOGICAL UNCERTAINTY FOR LANE IDENTIFICATION OF AUTONOMOUS VEHICLES

Most localization algorithms use metric maps as aids in the localization process, which represent features in continuous coordinates. Topological maps represent space as discrete components (vertices) and their logical-spatial relationship (edges) where vertices and edges are taken in the graph theoretic sense. The motivating example in this chapter is an autonomous vehicle (AV) which, in addition to requiring a metric location estimate, also requires a topological location estimate at the lane level. For example, the AV may need to know not just that it is on Pleasant Street, but whether or not it is in the left turn only lane. Moreover, events such as construction, traffic accidents, natural disasters, and native map errors may result in discrepancies between the topology suggested by the map and reality. The localization algorithm on the AV must be able to reason about this possibility.

Reasoning *globally* about all possible topologies is computationally intractable, since the number of unique topologies scales exponentially with the number of locations. Furthermore, there may be uncertainty in the number of locations. Moreover, global topological information is rarely present. Instead, we propose a method for reasoning about location and structure within the *local*, observable topology. Restricting the scope allows inference algorithms to reason about multiple topologies with varying numbers of nodes.

The discrete nature of topological location, combined with the requirement to reason about multiple possible realities simultaneously, motivates our approach. The first key idea, shown in 4.1, is to use Hidden Markov Models (HMMs) for localization by modeling expected observations and transitions at and between topological nodes. The second key idea

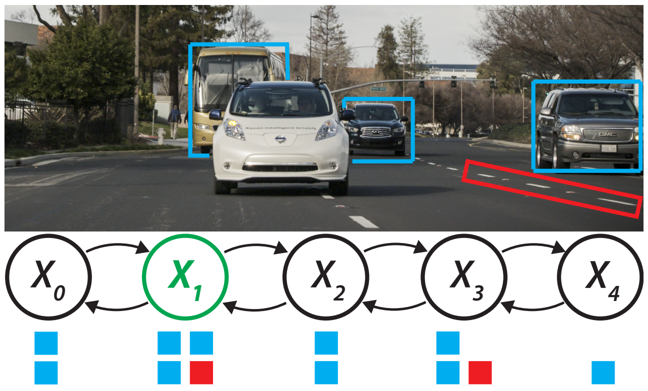


Figure 4.1: Agreement of observations to HMM lane-states. State x_1, representing being between the right and center lanes, is the only state for which all three vehicle detections (blue) and the lane line detection (red) are likely for the AV (green).

is to use a variable set of HMMs to model a variety of possible realities. Here, HMMs model the transition dynamics and observation models of a topological map, analogous to how Kalman filters model these aspects of a tracked object. Thus, we call this approach Variable Structure Multiple Hidden Markov Models (VSM-HMM); its function being similar to Variable Structure Multiple Models [194]. An important distinction is that VSMMs allow tracking of objects which follow a variety of process (or dynamical) models, whereas the VSM-HMM approach reasons about multiple world models.

This chapter covers three related contributions. First, we demonstrate a method for applying HMMs to lane-level localization on an AV (§4.1). Second, we describe our VSM-HMM approach to managing a dynamic set of HMMs, each of which estimates a location within a unique local topology, allowing us to reduce dependence on high-definition (HD) maps (§4.2). Third, we extend the Earth Mover's Distance (EMD) in order to handle distributions which have domains of different sizes during model belief initialization (§4.3).

Our approach is evaluated in simulation as well as on 6 real-world data sets gathered on public, multi-lane roads in Silicon Valley. Results presented in §4.4 show that the VSM-HMM model can provide accurate topological location estimates, as well as detect disagreements between the topology specified by the map and that supported by observation [249].

There are several different approaches to topological localization. The problem of global topological localization may be modeled as a partially observable Markov decision process (POMDP). However, approaches using POMDPs, either in conjunction with geometric landmarks [406] or fingerprints from visual input [352], do not scale well with the number of topological nodes.

To reduce complexity, some approaches eliminate most of the reasoning about uncertainty by directly matching the robot's current view against representative feature vectors from topological locations in the map. A variety of map representations, feature extraction methods, and distance measures have been examined, including Generalized Voronoi Graphs [69], SIFT and SURF features [14], and Jeffrey divergence [371]. These approaches work well for environments in which nodes can be visited often, and their representative feature vectors kept up-to-date, as in [79]. However, this is rarely possible for an AV.

Localization algorithms specifically for AVs have typically focused on metric location, relying on high-definition (HD) maps for reliable, global data association. There are many variants, including approaches which use vanilla particle filters [319], Rao-Blackwellized particle filters [186], and Kalman Filters [321, 351, 107]. However, these approaches require HD maps and compute both metric and topological location in a single pass.

In contrast, the proposed VSM-HMM model allows decoupling of metric and topological estimates, creating a hybrid metric-topological problem [299, 300], although this chapter focuses only on the topological component. Similar to FastSLAM [237], VSM-HMM maintains multi-modal belief over not only topological location, but also the local topological structure. Until now, particle filters were the only viable multi-modal topological localization framework. Moreover, particle filters resampling ignores local topological structure, whereas the VSM-HMM exploits this.

Dynamic Bayesian networks [244] are common tools for representing localization problems [156, 40], and HMMs specifically have been used for topological localization before. HMMs have also been used as an optional layer to process video feed in the case of a low-confidence nearest neighbor match [173], and trained to detect and classify specific railroad turnouts using sequences of signals from eddy current sensors [133]. Multiple HMMs have been suggested for other tasks where the number of classes is high, such as genome sequencing [115], and hierarchical HMMs [103] have been used for language, handwriting, and speech recognition.

4.1 Lane Identification using HMMs

Hidden Markov Models are promising candidates for topological localization for two primary reasons. First, HMMs are well understood theoretically and support many efficient modes of inference. In our application, we use the Forward algorithm because it affords low computational cost. Second, HMMs support learning and are easily designed for specific sensor features and or topological structure. Additionally, methods such as [332], can be used to refine observations prior to evaluation by the HMM.

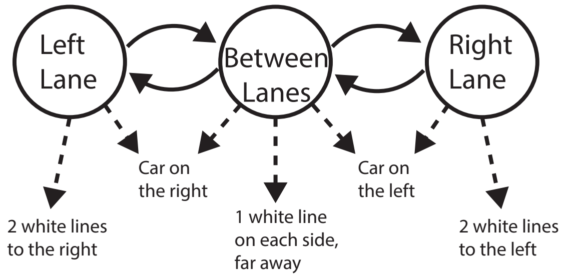

Left Lane
Between Lanes
Right Lane
Car on the right
Car on the left
2 white lines to the right
1 white line on each side, far away
2 white lines to the left

Figure 4.2: Simplified example lane-state HMM. Solid lines represent non-zero transition probabilities between states. Self-loops are not shown. Dashed lines represent observation or emission probabilities, some of which are exclusive for a single state. Note that the transition structure of the HMM models the topological structure of the environment. Some types of obervations have been omitted for simplicity.

Topological location is modeled as occupancy of a state $x_i \in X$ within an HMM. The application of this work is toward lane-level localization, and thus the states X in the HMM correspond to either being in the center of a lane or being between two lanes. For example, an HMM representing a two lane road, detailed in 4.2, has three states x_0, x_1, and x_2. x_0 and x_2 correspond to occupying the right lane and the left lane, respectively. x_1 represents having some portion of the car over the lane divider. We call states like x_1 *switching states*. In general, an HMM representing a road with L lanes will have $2L - 1$ states.

A particular strength of HMMs in topological localization is their ability to efficiently model real-world dynamics via their transition function. Since the AV can only move from one lane to an adjacent lane by moving through an immediately adjacent switching state, the transition matrix representing the transition function is sparse. We define the state transition matrix for an n-state HMM, $\tau^{\mathbf{n}}$, as

$$\tau^n_{ij} = \begin{cases} t_r \text{ if } i = j \\ t_s \text{ if } |i - j| = 1 \\ 0 \text{ otherwise,} \end{cases} \tag{4.1}$$

where t_r and t_s are parameters for the probability of remaining in the same state and switching to an adjacent state, respectively. This transition matrix reduces reports of physically impossible events, such as instantaneous changes across multiple lanes, which is a key advantage over other multimodal approaches such as particle filters.

In addition to GPS and inertial sensors, we use lane line detections and observed relative locations of other vehicles to inform our topological estimate, shown in 4.3. For lane lines we use a combination of learned parameters and information from a map to parametrize a Gaussian Mixture Model which describes the likelihood of observing a lane line at positions and orientations relative to the AV, given a particular lane-state.

Since observations of lane lines are unreliable due to occlusions, weather and lighting conditions, and absence of the markings themselves on many roads, we also use the relative positions of nearby tracked vehicles to support occupancy of certain lane-states. The key idea is that, although other vehicles move both globally and relatively with respect to the AV, they do so according to a specific local pattern defined by lane membership. For example, if the AV is traveling on a two-lane road and senses a vehicle immediately to its right, then there is a high probability that the AV is in the left lane, since the observed vehicle is

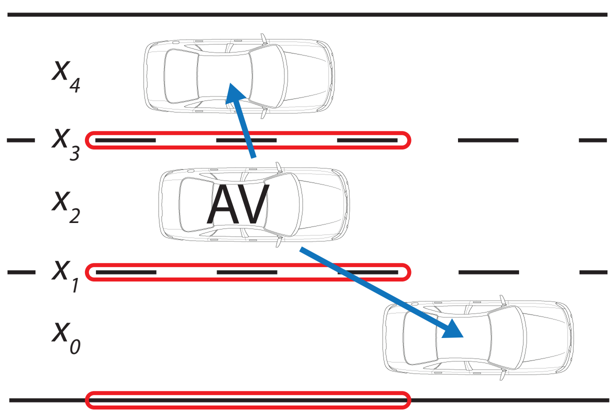


Figure 4.3: Diagram of lane-states and observable features. Lane-states $x_0, \ldots x_4$ correspond to distinct topological regions on the road for which there are expected observations. x_2 is the only state which both lane line measurements (red) and vehicle detections (blue) support. Lane line measurements alone would result in equal belief in states x_2 and x_4.

far more likely to be traveling in the right lane than to be traveling beyond the edge of the road. We denote the observation function for state x_i and sensor q as $\phi(x_i, q)$.

Although single HMMs are reliable for topological localization when the map is correct and the number of states in the HMM matches the number true lane-states, they can fail when this is not true. To deal with this, we introduce the Variable Structure Multiple Hidden Markov Model.

4.2 Variable Structure Multiple HMMs

The Variable Structure Multiple Hidden Markov Model (VSM-HMM) is similar to the variable-structure multiple-model set of Kalman Filters used in many tracking domains. However, whereas the multiple-model approaches in the tracking and control literature predict the current dynamical mode of the tracked object, the VSM-HMM approach estimates the current structure of the local topology. That is, the VSM-HMM hypothesizes about the outside world state rather than an internal process model. This is necessary when the local topological map has non-zero uncertainty.

We define a VSM-HMM as a set of all possible models U, an active model set U_A and an inactive model set U_N. Every $u \in U$ is an HMM with a unique transition matrix defined by the number of lanes. $U_A \cup U_N = U$, and $U_A \cap U_N = \emptyset$.

At every time step, U_A is determined using a variation of the Likely Model Set (LMS) algorithm [193], outlined in Algorithm 5. For every $u \in U$, we compute a model likelihood (line 6). Pr$(u|M)$ is the probability of model u given the map M. If there was no uncertainty in M, then Pr$(u|M)$ would be 1 for the model suggested by the map, u_M, and 0 otherwise. In our implementation, Pr$(u|M) = \alpha 2^{-|(|u|-|u_M|)/2|}$, where $|u_M|$ is the number of states in u_M, and α is a normalizing constant. Pr$(z|u)$ is the maximum probability of observing z given some state in u, $\textsc{Max}_{x_i}$Pr$(z|x_i \in X_u)$. In 4.3, Pr$(z|u)$ would be low for models with fewer than three lanes since z contains features which are unexpected in models with fewer than three lanes.

After model likelihood is computed, up to κ models are chosen so long as the ratio between their likelihood and the maximum likelihood of all models is above a threshold T_{active} (lines 8-14). κ is chosen based on computational constraints. Last, belief is copied from active models and initialized for inactive models (lines 15-19). Initializing belief is done using the Extended Earth Mover's Distance (§4.3).

Discrepancies between the map and reality are detected by calculating the entropy, H, of the posterior probability (belief) over the states in each model:

$$H(bel(X)) = -\frac{1}{\log(|X|)} \sum_{i=1}^{|X|} bel(x_i) \log(bel(x_i)).$$

If the model suggested by the map has a high entropy compared to another model, the map is likely incorrect since high entropy indicates no state in the suggested topology can explain the observations. Note that the normalized equation for entropy calculates equal values for all models when no information is present. Thus, having a lack of information altogether will not cause the algorithm to flag the map as having an error. Only observations which both contradict the map's topology *and* may be explained by a different topology

Algorithm 5 LIKELY MODEL SET

```
1: Input: All models $U$, active models $U_A$, observations $z$, max number of models $\kappa$,
   threshold $T_{active}$, map $M$
2: Output: Set of updated most likely models $U'_A$
3: $U'_A \leftarrow \emptyset$
4: $S \leftarrow [\,]$
5: for all $u \in U$ do
6:     $\mathcal{L} \leftarrow \Pr(u|M) \times \Pr(z|u)$
7:     $S \leftarrow S.\text{APPEND}(\mathcal{L}, u)$
8: $\mathcal{L}_{max} \leftarrow Max_{\mathcal{L}}(S)$
9: for all $\mathcal{L}, u \in S$ do
10:     if $U'_A = \emptyset$ then
11:         $U'_A \leftarrow U'_A \cup u$
12:     else
13:         if $|U'_A| < \kappa$ and $\frac{\mathcal{L}}{\mathcal{L}_{max}} > T_{active}$ then
14:             $U'_A \leftarrow U'_A \cup u$
15: for all $u' \in U'_A$ do
16:     if $u' \in U_A$ then
17:         $u' \leftarrow \text{COPYEXISTINGBELIEF}(u')$
18:     else
19:         $u' \leftarrow \text{INITNEWBELIEF}(u')$
   return $U'_A$
```

will result in a high entropy ratio. If $|U_A| > 1$, serving a localization requests amounts to picking the most likely state from the model with the lowest entropy.

In theory, one could devise a single HMM with a dense transition matrix τ^*, which models the same problem. We can define τ^* in terms of the sub-blocks representing each distinct topological hypothesis, τ^1, τ^2, ..., τ^n (models in the VSM-HMM), and the transitions between them. Let each sub-block τ^k lie on the diagonal of τ^*. Further, let the off-diagonal blocks hold the probabilities of switching between sub-blocks, t_m. Note that t_m is a notational placeholder for a range of values corresponding to the probabilities of transitioning between two specific sub-blocks (models).

$$\tau^*_{ij} = \begin{cases} \tau^k_{lm} \text{ if } i, j \text{ index } l, m \text{ in sub-block } k \\ t_m \text{ otherwise.} \end{cases} \tag{4.2}$$

In general, this results in a $p \times p$ transition matrix, where $p = \sum_{u \in U} |X_u|$. Similarly, X^* and ϕ^* are defined by the union of all state spaces and observation functions, respectively.

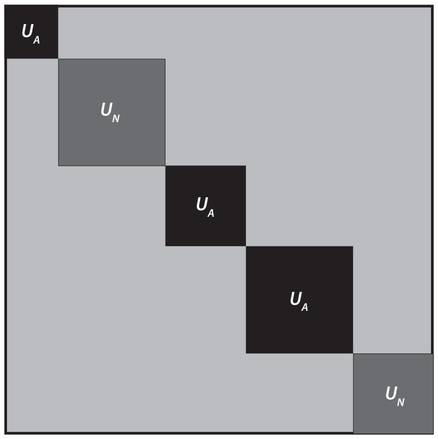


Figure 4.4: Transition matrix for single HMM analog to VSM-HMM. Light gray areas hold probabilities of switching models, t_m. Block diagonals represent all models in U. Note that the VSM-HMM reasons about only the active models, represented by the black sub-blocks.

However, even when $U_A = U$, the VSM-HMM is more computationally efficient than its single HMM analog since calculating $\Pr(x_t|x_{t-1})$ only considers the block diagonals instead of the entire dense matrix, and calculating when to switch models depends only on the block diagonal size. In practice, $U_A \subset U$ is much more common. In this case, the VSM-HMM approximates the equivalent HMM by reasoning over a subset of the most likely belief points, shown in 4.4.

4.3 Extended Earth Mover's Distance

Whenever a model is initialized, it needs a starting belief. Suppose an AV has a belief, β, about its current topological position in a local topology. β is discrete and lies on the n-simplex, Δ^n, where $n+1$ is the number of local topological states. Here, $n+1 = 2L-1$, where L is the number of lanes on the road the AV is traveling. Further, suppose the AV is nearing an intersection or merge in which the number of lanes on the road the AV will end up on is L'. Once on the new road the AV will initialize a new model and need a new belief, β', about its topological location. However, if $L' \neq L$, β and β' will be over different numbers of states. The question is, if $L' \neq L$ how do we initialize β', given β.

One option is to erase all previous belief and start over from uniform, $\beta' = \mathcal{U}(0, m)$. Another option is to heuristically initialize β', such as right- or left-alignment of lane-states. Both options are computationally efficient, but do not perform optimally in many cases. A third option is to initialize β' as the 'closest' distribution in Δ^m to β, where 'closeness' is defined by some statistical metric. This is preferable, but there are no metrics which satisfy the constraints of the problem since there is no isomorphism between Δ^n and Δ^m, and the mapping from some belief point in Δ^n to the corresponding point Δ^m is uncertain. One example of uncertainty is a two-lane road which becomes a three-lane road across an intersection. It is unclear whether the two lanes in the first road correspond to the two rightmost, two leftmost, or some other combination of lanes in the three-lane road.

Thus, we introduce a statistical metric based on the Earth Mover's Distance (EMD) [312], called the Extended Earth Mover's Distance (EEMD), which measures the *expected* distance between distributions on simplices of arbitrary relative size, given the probability of all mappings between simplices. We initialize β' such that EEMD(β, β') is minimized.

Before defining EEMD, we introduce some notation. Let $\mathbb{P}^n$ and $\mathbb{P}^m$ be normalized distributions on Δ^n and Δ^m, respectively, and define $N = n+1$, and $M = m+1$. Without loss of generality, suppose $n > m$. Let the function $\mathbf{f}^{m,n} : \Delta^m \to \Delta^n$ be defined as

$$f_j^{m,n}(\mathbb{P}^m) = \begin{cases} \mathbb{P}_j^m \text{ if } j \leq M \\ 0 \text{ if } j > M. \end{cases} \tag{4.3}$$

Thus, $\mathbf{f}^{m,n}$ pads $\mathbb{P}^m$ with dimensions with zero belief, making it the same size as $\mathbb{P}^n$. We denote this new distribution, now on Δ^n, $\mathbb{P}^{m'}$. We can now use the original formulation of EMD to compute distance between $\mathbb{P}^{m'}$ and $\mathbb{P}^n$. However, in general, there may be uncertainty in the mapping between the two distributions. It may be that $\mathbb{P}_j^{m'}$ and $\mathbb{P}_j^n$ do not correspond to the same real world state. There are N^N possible mappings from $\mathbb{P}^{m'}$ to $\mathbb{P}^n$. We calculate the expected distance by summing over all possibilities for $\mathbb{P}^{m'}$, and compute the EMD weighted by the probability of each mapping, Pr($\mathbb{P}^{m'_i}$). Thus, we define the Extended Earth Mover's Distance between $\mathbb{P}^n$ and $\mathbb{P}^m$ as

$$\text{EEMD}(\mathbb{P}^n, \mathbb{P}^m) = \sum_{i=1}^{N^N} \text{Pr}(\mathbb{P}^{m'_i})\text{EMD}(\mathbb{P}^n, \mathbb{P}^{m'_i}). \tag{4.4}$$

It is assumed that Pr($\mathbb{P}^{m'_i}$) is known and normalized. In practice we calculate it using information from the map, and our problem has structure allowing us to ignore most of the summands since the corresponding Pr($\mathbb{P}^{m'_i}$) term is 0. We use the EEMD as a principled guide to constructing distributions for model initialization. β' is calculated such that EEMD(β, β') is minimized.

4.3.1 Proof of EEMD Metric Properties

Theorem 1. *EEMD is a metric, having the properties of non-negativity, identity, symmetry, and the triangle inequality.*

Non-negativity: Since EMD is a metric, it is always positive. Pr($\mathbb{P}^{m'_i}$) is always non-negative. Thus, their product is always non-negative, and the sum of non-negative elements is also non-negative.

Identity:

(EEMD$(\mathbb{P}^n, \mathbb{P}^m) = 0 \implies \mathbb{P}^n = \mathbb{P}^m$).

If EEMD$(\mathbb{P}^n, \mathbb{P}^m) = 0$, then for all summands, either Pr$(\mathbb{P}^{m'_i}) = 0$ and or EMD$(\mathbb{P}^n, \mathbb{P}^{m'_i}) = 0$. Since Pr($\mathbb{P}^{m'_i}$) is a distribution, there must be at least one i such that Pr$(\mathbb{P}^{m'_i}) > 0$. If there are more than one such i, then EEMD$(\mathbb{P}^n, \mathbb{P}^m)$ cannot be zero, since each $\mathbb{P}^{m'_i}$ is unique and EMD is a metric, violating the assumption EEMD$(\mathbb{P}^n, \mathbb{P}^m) = 0$. If there is a single i such that Pr$(\mathbb{P}^{m'_i}) > 0$, then since EMD is a metric and must be 0, EEMD$(\mathbb{P}^n, \mathbb{P}^m) = 0 \implies$ EMD$(\mathbb{P}^n, \mathbb{P}^{m'_i}) = 0 \implies \mathbb{P}^n = \mathbb{P}^m$.

$(\mathbb{P}^n = \mathbb{P}^m \implies \text{EEMD}(\mathbb{P}^n, \mathbb{P}^m) = 0)$.

This is clear from the definition of EEMD and Pr$(\mathbb{P}^{m'_i})$. All Pr$(\mathbb{P}^{m'_i})$ will be 0 except when $\mathbb{P}^{m'_i} = \mathbb{P}^m$. For this term, the corresponding EMD will be 0 since EMD is a metric.

Symmetry: The smaller dimension is always augmented, regardless of order. Thus, since Pr$(\mathbb{P}^{m'_i})$ is constant, the exact same calculation is performed for both EEMD$(\mathbb{P}^n, \mathbb{P}^m)$ and EEMD$(\mathbb{P}^m, \mathbb{P}^n)$. So, EEMD$(\mathbb{P}^n, \mathbb{P}^m)$ = EEMD$(\mathbb{P}^m, \mathbb{P}^n)$.

Triangle Inequality: Let m, n, and k be non-negative integers, and consider the three simplices, Δ^m, Δ^n, and Δ^k. Without loss of generality, let $m < n < k$ and define $N = n+1$, $M = m+1$, and $K = k+1$.

Lemma 1.

EEMD$(\mathbb{P}^n, f^{m,n}(\mathbb{P}^m)) =$ EEMD$(f^{n,k}(\mathbb{P}^n), f^{m,k}(\mathbb{P}^m))$.

From the definition of EEMD,

$$\text{EEMD}(\mathbb{P}^n, f^{m,n}(\mathbb{P}^m)) = \sum_{i=1}^{N^N} \text{Pr}(\mathbb{P}^{m'_i})\text{EMD}(\mathbb{P}^n, \mathbb{P}^{m'_i}) \tag{4.5}$$

and

$$\text{EEMD}(f^{n,k}(\mathbb{P}^n), f^{m,k}(\mathbb{P}^m)) = \sum_{i=1}^{K^K} \text{Pr}(\mathbb{P}^{m''_i})\text{EMD}(\mathbb{P}^{n'}, \mathbb{P}^{m''_i}). \tag{4.6}$$

We call dimensions $1{:}N$ *essential* dimensions, and dimensions $(N+1){:}K$ *extra* dimensions. The sum on the RHS of equation (6) can be decomposed into two parts: one sum, with N^N terms, which corresponds to all mappings in which the extra $K - N$ dimensions map only amongst themselves, and another sum, with $K^K - N^N$ terms, which corresponds to all mappings where at least one of the extra dimensions maps to one of the essential dimensions. Thus, we can rewrite the RHS of equation (6) as

$$\sum_{i=1}^{N^N} \mathrm{Pr}(\mathbb{P}^{m_i''})\mathrm{EMD}(\mathbb{P}^{n'}, \mathbb{P}^{m_i''}) + \sum_{i=N^N+1}^{K^K} \mathrm{Pr}(\mathbb{P}^{m_i''})\mathrm{EMD}(\mathbb{P}^{n'}, \mathbb{P}^{m_i''}). \tag{4.7}$$

Since none of the extra dimensions has any meaning in the original problem and therefore cannot possibly map to any essential dimension, $\mathrm{Pr}(\mathbb{P}^{m_i'}) = 0$ for all $i > N^N$. Thus, the second sum is 0 and equation (6) becomes

$$\sum_{i=1}^{K^K} \mathrm{Pr}(\mathbb{P}^{m_i''})\mathrm{EMD}(\mathbb{P}^{n'}, \mathbb{P}^{m_i''}) = \sum_{i=1}^{N^N} \mathrm{Pr}(\mathbb{P}^{m_i''})\mathrm{EMD}(\mathbb{P}^{n'}, \mathbb{P}^{m_i''}). \tag{4.8}$$

Furthermore, since all extra dimensions have weight 0, their contribution to all summands in equation (8) is 0. So, equation (8) becomes

$$\sum_{i=1}^{K^K} \mathrm{Pr}(\mathbb{P}^{m_i''})\mathrm{EMD}(\mathbb{P}^{n'}, \mathbb{P}^{m_i''}) = \sum_{i=1}^{N^N} \mathrm{Pr}(\mathbb{P}^{m_i'})\mathrm{EMD}(\mathbb{P}^{n}, \mathbb{P}^{m_i'}), \tag{4.9}$$

establishing the lemma. Now, let $\mathbb{P}^m$, $\mathbb{P}^n$, and $\mathbb{P}^k$ be distributions on Δ^m, Δ^n, and Δ^k, respectively. Consider

$$\mathrm{EEMD}(\mathbb{P}^m, \mathbb{P}^k), \text{ and}$$
$$\mathrm{EEMD}(\mathbb{P}^m, \mathbb{P}^n) + \mathrm{EEMD}(\mathbb{P}^n, \mathbb{P}^k).$$

By Lemma 1, we can rewrite these as

$$\mathrm{EEMD}(f^{m,k}(\mathbb{P}^m), \mathbb{P}^k), \text{ and}$$
$$\mathrm{EEMD}(f^{m,k}(\mathbb{P}^m), f^{n,k}(\mathbb{P}^n)) + \mathrm{EEMD}(f^{n,k}(\mathbb{P}^n), \mathbb{P}^k).$$

All distributions now lie on Δ^k. Since all nodes in the simplex are unit distance from all other nodes, then the distance calculated by EMD for weight moving from one node to any other node will be the magnitude of the weight. Thus, if $\mathbb{P}^n$ maintains belief on any nodes with different magnitude than $\mathbb{P}^m$ or $\mathbb{P}^k$, then

$$\text{EEMD}(f^{m,k}(\mathbb{P}^m), \mathbb{P}^k) <$$
$$\text{EEMD}(f^{m,k}(\mathbb{P}^m), f^{n,k}(\mathbb{P}^n)) + \text{EEMD}(f^{n,k}(\mathbb{P}^n), \mathbb{P}^k).$$

Otherwise,

$$\text{EEMD}(f^{m,k}(\mathbb{P}^m), \mathbb{P}^k) =$$
$$\text{EEMD}(f^{m,k}(\mathbb{P}^m), f^{n,k}(\mathbb{P}^n)) + \text{EEMD}(f^{n,k}(\mathbb{P}^n), \mathbb{P}^k).$$

4.4 Results

To test the VSM-HMM framework, we perform two experiments. The first measures localization accuracy, and the second tests the framework's ability to reason about local topological structure and detect discrepancies between the map and reality.

Localization accuracy was tested on 6 hand-annotated datasets gathered by an AV on public, multi-lane roads near Nissan Research Center in Silicon Valley. Each dataset was recorded over about a mile of stop-and-go traffic and ranged in time from 2 to 6 minutes. All road segments had between 3 and 6 lanes, corresponding to between 5 and 11 states. In these experiments no metric location information, such as GPS, was used, and topological ground truth was provided.

Because of the intermittent nature of the lane line and vehicle detections, not all timesteps possess enough observations to disambiguate lane-states. Thus, in the results presented in 4.1 we do not consider instances in which either no observations were recorded or the observations voted for at least half of all states, such as seeing only a single lane line immediately to the left of the vehicle. These instances are labeled "Missing Observations." Given sequences of timesteps with little or no observations, it is possible to have multiple states tie for the same belief. Localization is considered correct if the true state is among those with maximum belief, and incorrect otherwise. Length and observation quality are shown so as to give an idea about the difficulty of the dataset. Predictions are made at 100Hz.

Dataset	Length (mins)	Missing Obs.	Accuracy
1	2.0	11%	83%
2	3.7	18%	74%
3	4.8	6%	83%
4	4.4	9%	95%
5	3.5	30%	81%
6	5.7	20%	73%

Table 4.1: Location estimation results

Testing topological structure estimation was done using simulated data, since there were too few cases in the real world data to draw concrete conclusions. To simulate false positive data, lane line and vehicle detections are generated according to the topology given by the map with probability P_M, and according to some other, randomly selected topology with probability $(1 - P_M)$. Further, to simulate the intermittent nature of real-world data, with probability $(1 - P_E)$ no observations are emitted. To see how our approach handles increased sensor noise, we tested different levels of variance. Given a variance σ based on real-world data, simulated observations are generated with variance $K_\sigma \sigma$, where K_σ is an experimental parameter.

Locations of lane lines and vehicle detections are sampled from multivariate normal distributions with means as a function of the given topology, and variances $K_\sigma \sigma$. Observation generation runs independently for each lane line and vehicle detection. Table 4.2 displays the results of local topological structure estimation. Combined, these results demonstrate VSM-HMM as an effective framework for dealing with topological uncertainty.

4.5 Conclusion

This chapter presented a framework, Variable Structure Multiple Hidden Markov Models (VSM-HMM), for topological localization in the presence of topological uncertainty, and established empirical results from both simulated and real-world data on an autonomous vehicle which support VSM-HMM's effectiveness. Not only does this method provide accurate estimates of lane membership, and topological location more broadly, but

P_E		0.9			0.7			0.5		
K_σ		1	2	3	1	2	3	1	2	3
P_M	0.9	96	95	92	94	88	83	83	78	72
	0.8	87	85	80	84	79	72	63	60	54
	0.7	75	74	72	66	65	61	55	50	50
	0.6	66	63	63	62	63	59	51	48	49

Table 4.2: Local topological structure estimation accuracy. Results are reported as the percent of timesteps during which the correct topological structure was estimated with highest probability (lowest entropy). P_M is probability of sampling from the correct topology. P_E is the probability of emitting observations. K_σ is the amount by which the variance is scaled. Each entry in the table was computed from performance over 1000 timesteps.

it also provides significant robustness when the map and world do not match, and does so efficiently via use of an active model set.

As in the previous chapter, one underlying design principle is to make use of existing data and reliable deployment conditions, here in the form of lane line and vehicle detections, which are already necessary for many other processes on the AV. Additionally, we again aim to make as few assumptions about the world as possible, in this case allowing many possible road topologies and lane configurations rather just the one suggested by the map. Graceful handling of such ‘corner cases’ is especially important in safety critical applications as these scenarios are often relatively more dangerous than nominal conditions.

The general strategy of using multiple models has many analogs in machine learning and other areas of AI, but generally has been adopted less in robotics. Perhaps due to the inherent complexity of implementing such systems, or roboticists’ natural inclination towards simpler, specialized, and more predictable systems, this absence highlights a well-known tension between performance via specialization and generalization, but moves the contested space from the relatively familiar space of model design to the even more complex and abstract considerations of system architecture design. This chapter demonstrates that not only is it possible to increase key performance metrics by leveraging model portfolios, but also shows that such a strategy may in fact be necessary in order to reach certain levels of generalization.

CHAPTER 5

HUMAN-IN-THE-LOOP SLAM

While some robotics tasks may be effectively completed in a small area, many, including several outlined in the first chapter, are necessarily performed over large areas. In such tasks it is not uncommon for the robot to need to formulate a plan for traversing between two points separated by considerable distance. Without a globally consistent map, it may be impossible to correctly formulate such a plan. Often, the performance of SLAM systems may be adequate for smaller environments but as the overall size of the map increases it becomes challenging to maintain global consistency.

Building large-scale globally consistent metric maps requires accurate relative location information between poses with large spatial separation. However, due to sensor noise and range limitations, such correlations across distant poses are difficult to extract from real robot sensors. Even when such observations are made, extracting such correlations autonomously is a computationally intensive problem. Furthermore, the order of exploration, and the speed of the robot during the exploration affect the numerical stability, and consequently the global consistency of large-scale maps. Due to these factors, even state-of-the-art mapping algorithms often yield inaccurate or inconsistent large-scale maps, especially when processing data collected by novice users in challenging environments.

To address these challenges and limitations of large-scale mapping, this chapter presents Human-in-the-Loop SLAM (HitL-SLAM), a principled approach to incorporate approximate human corrections in the process of solving for metric maps. Figure 5.1 presents an example of HitL-SLAM in practice. HitL-SLAM operates on a pose graph estimate of a map along with the corresponding observations from each pose, either from

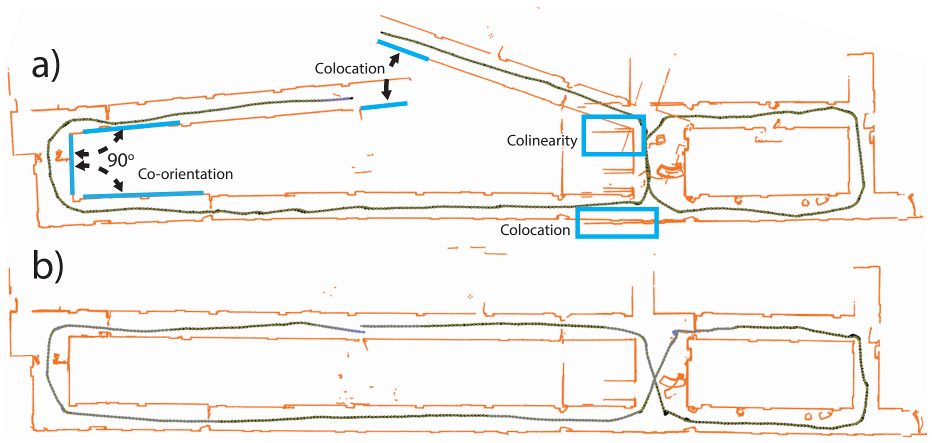


Figure 5.1: HitL SLAM example, showing a) the input initial map with global consistency errors, and b) the resulting final map produced by HitL-SLAM by incorporating human corrections (blue lines) along with the input.

an existing state-of-the-art SLAM solver, or aligned purely by odometry. In an interactive, iterative process, HitL-SLAM accepts human corrections, re-solves the pose graph problem, and presents the updated map estimate. This iterative procedure is repeated until the user is satisfied by the mapping result and provides no further corrections.

This chapter highlights three primary contributions: 1) an EM-based algorithm [85] to interpret several types of approximate human correction, 2) a human factor formulation to incorporate a variety of types of human corrections in a factor graph for SLAM, and 3) a two-stage solver to solve the resulting factor graph including human factors and pose graph factors, with minimal distortion and accounting for rank deficiency in the human corrections. We show how HitL-SLAM introduces numerical stability in the mapping problem by introducing off-diagonal blocks in the information matrix of the joint human factor graph for SLAM. Finally, we present several examples of HitL-SLAM operating on maps that intrinsically included erroneous observations and poor initial map estimates, and producing accurate, globally consistent maps.

Solutions to robotic mapping and SLAM have improved dramatically in recent years, but state-of-the-art algorithms still fall short at being able to repeatably and robustly produce globally consistent maps, particularly when deployed over large areas and by nonexpert users. This is in part due to the difficulty of the data association problem [90, 23, 20]. The idea of humans and robots collaborating in the process of map building to overcome such limitations is not new, and is known as Human-Augmented Mapping (HAM).

Work within HAM can be categorized depending on whether the human and robot collaborate in-person during data collection (C-HAM), or whether the human provides input remotely or after the data collection (R-HAM). Many C-HAM techniques exist to address semantic [266, 366] and topological [365] mapping. A number of approaches have also been proposed for integrating semantic and topological information, along with human trackers [231], interaction models [367], and appearance information [295], into *conceptual spatial maps* [408], which are organized in a hierarchical manner.

There are two limitations in these C-HAM approaches. First, a human must be present with the robot during data collection. This places physical constraints on the type of environments which can be mapped, as they must be accessible and traversable by a human. Second, these methods are inefficient with respect to the human's attention, since most of the time the human's presence is not critical to the robot's function, for instance during navigation between waypoints. These approaches, which focus mostly on semantic and topological mapping, also typically assume that the robot is able to construct a nearly perfect metric map entirely autonomously. While this is reasonable for small environments, globally consistent metric mapping of large, dynamic spaces is still a hard problem.

In contrast, most of the effort in R-HAM has been concentrated on either incorporating human input remotely via tele-operation such as in the Urban Search and Rescue (USAR) problem [245, 268], or in high level decision making such as goal assignment or coordination of multiple agents [269, 274, 94]. Some R-HAM techniques for metric mapping and pose estimation have also been explored, but these involve either having the robot retrace its steps to fill in parts missed by the human [169] or by having additional agents and sensors in the environment [171]. A number of other approaches have dealt with interpreting graphical or textual human input within the contexts of localization [28, 45] and semantic mapping [130]. While these approaches solve similar signal interpretation problems, this paper specifically focuses on metric mapping.

Ideally, a robot could explore an area only once with no need for human guidance or input during deployment, and later with minimal effort, a human could make any corrections necessary to achieve a near-perfect metric map. This is precisely what HitL-SLAM does, and additionally HitL-SLAM does not require in-person interactions between the human and robot during the data collection.

5.1 Human-in-the-Loop SLAM

HitL-SLAM operates on a factor graph $G = \{X, F\}$, where X is the set of estimated poses along the robot's trajectory, and $F = \{R, H\}$ is the set of factors which encode information about both relative pose constraints arising from odometry and observations, R, and constraints supplied by the human, H. The initial factor graph G_0 may be provided by any pose graph SLAM algorithm, and HitL-SLAM is capable of handling constraints in G_0 with or without loop closure. In our experiments, we used Episodic non-Markov Localization (EnML) [41] without any explicit loop closures beyond the length of each episode.

HitL-SLAM runs iteratively, with the human specifying constraints on observations in the map, and the robot then enforcing those constraints along with all previous constraints to produce a revised estimate of the map. To account for inaccuracies in human-provided corrections, interpretation of the such input is necessary before human correction factors can be computed and added to the factor graph. Each iteration, the robot first proposes an initial graph $G_i = \{X_i, F_i\}$, then the human supplies a set of correction factors H_i, and finally the robot re-optimizes the poses in the factor graph, producing $G'_i = \{X'_i, F_i \cup H_i\}$.

Definition 1. A **human correction factor**, is a tuple $h = \langle P_a, P_b, S_a, S_b, X_a, X_b, m\rangle$, where:

- $P_a, P_b \subset \mathbb{R}^2$: Sets of end-points of the two line segments a, b drawn by the human,

• $S_a, S_b \subset S$: Sets of observations selected by the two line segments a, b respectively,

• $X_a, X_b \subset X_{1:t}$: Sets of poses from which the observations S_a, S_b were made,

• $m \in M$: The mode of correction.

S_a, S_b are subsets of all observations S, and poses x_i are added to the sets X_a, X_b if there are observations in S_a, S_b that arising from pose x_i. M is an enumeration of the modes of human correction, a subset of which are shown in Figure 5.2. The modes M of correction are defined as follows:

1. *Colocation:* A full rank constraint specifying that two sets of observations are at the same location, and with the same orientation.
2. *Collinearity:* A rank deficient constraint specifying that two sets of observations are on the same line, with an unspecified translation along the line.
3. *Perpendicularity:* A rank deficient constraint specifying that the two sets of observations are perpendicular, with an unspecified translation along either of their lines.
4. *Parallelism:* A rank deficient constraint specifying that the two sets of observations are parallel, with an unspecified translation along the parallel lines.

Each iteration of HitL-SLAM proceeds in two steps, shown in Figure 5.3. First, the human input is gathered, interpreted, and a set of human correction factors are instantiated (Block 1). Second, a combination of analytical and numerical techniques is used to jointly optimize the factor graph using both the human correction factors and the relative pose factors (Block 2). The resulting final map may be further revised and compressed by algorithms such as Long-Term Vector Mapping [248].

We model the problem of interpreting human input as finding the observation subsets S_a, S_b and human input sets P_a, P_b which maximize the joint correction input likelihood, $p(S_a, S_b, P_a, P_b | P_a^0, P_b^0, m)$, which is the likelihood of selecting observation sets S_a, S_b and point sets P_a, P_b, given initial human input P_a^0, P_b^0 and correction mode m. To find S_a, S_b

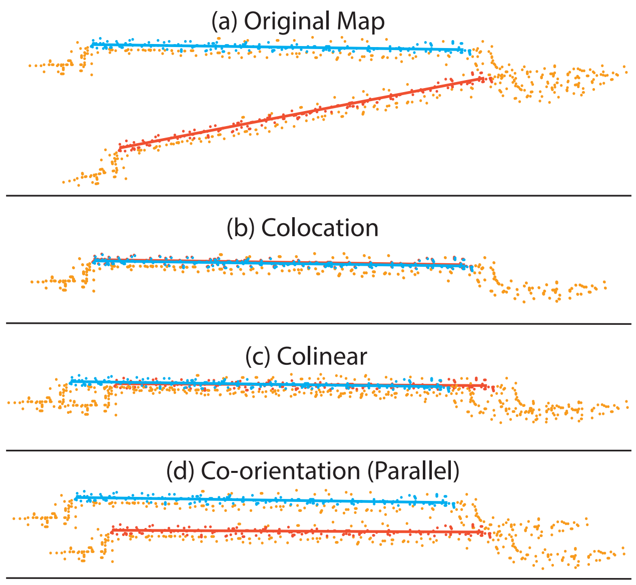


Figure 5.2: Result of transforming observation point clouds based on different human constraints, showing (a) Original map, (b) Colocation constraint, (c) Collinear constraint, (d) Co-orientation constraint. In all sub-figures the red and blue lines denote P_a and P_b, respectively, and red and blue points denote S_a and S_b. $S \setminus (S_a \cup S_b)$ appear in orange.

and P_a, P_b we use the sets P_a^0, P_b^0 and observations in a neighborhood around P_a^0, P_b^0 as initial estimates in an Expectation Maximization approach. As the pose parameters are adjusted during optimization in later iterations of HitL-SLAM, the locations of points in P_a, P_b may change, but once an observation is established as a member of S_a or S_b its status is not changed.

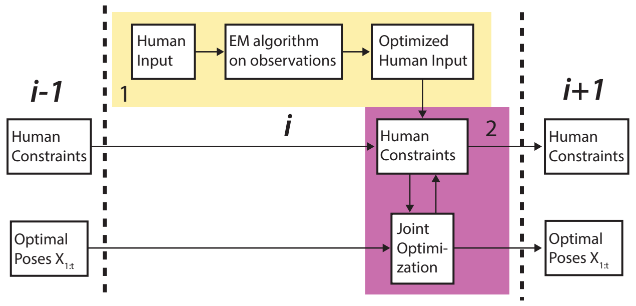


Figure 5.3: Flow of information during processing of the i^{th} human input. Block 1 (yellow) outlines the evaluation of human input, and block 2 (purple) outlines the factor graph construction and optimization processes. Note that the joint optimization process optimizes *both* pose parameters *and* human constraint parameters.

Once P_a, P_b and S_a, S_b are determined for a new constraint, then given m we can find the set of poses $X^*_{1:t}$ which best satisfy all given constraints. We first compute an initial estimate $X^0_{1:t}$ by analytic back-propagation of the most recent human correction factor, considering sequential constraints in the pose-graph. Next, we construct and solve a joint optimization problem over the relative pose factors r and the human correction factors h. This amounts to finding the set of poses $X^*_{1:t}$ which minimize the sum of the cost of all factors,

$$X^*_{1:t} = \underset{X_{1:t}}{\operatorname{argmin}} \left[\sum_{i=1}^{|R|} c_r(r_i) + \sum_{j=1}^{|H|} c_m(h_j) \right],$$

where $c_r : R \to \mathbb{R}$ computes the cost from relative pose-graph factor r_i, and $c_m : H \to \mathbb{R}$ computes the cost from human correction factor h_j with correction mode m. Later sections cover the construction of the human correction factors and the formulation of the optimization problem.

5.2 Interpreting Human Input

5.2.1 Human Input Interpretation

Due to a number of factors including imprecise input devices, screen resolution, and human error, what the human actually enters and what they intend to enter may differ slightly. Given the raw human input line segment end-points P^0_a, P^0_b and the mode of correction m, we frame the interpretation of human input as the problem of identifying the observation sets S_a, S_b and the effective line segment end-points P_a, P_b most likely to be captured by the potentially noisy points P^0_a, P^0_b. To do this we use the EM algorithm, which maximizes the log-likelihood ℓ,

$$\ell(\theta) = \sum_i \sum_{z_i} p(z_i|s_i, \theta^{\text{old}}) \log(p(z_i, s_i|\theta)),$$

where the parameters $\theta = \{P_a, P_b\}$ are the interpreted human input (initially assumed to be P_a^0, P_b^0), the $s_i \in S$ are the observations, and the latent variables z_i are indicator variables denoting the inclusion or exclusion of s_i from S_a or S_b. The expressions for $p(z_i|s_i, \theta^{\text{old}})$ and $p(z_i, s_i|\theta)$ come from a generative model of human error based on the normal distribution, $\mathcal{N}(\mu(\theta), \sigma^2)$. Here, σ is the standard deviation of the human's accuracy when manually specifying points, and is determined empirically; $\mu(\theta)$ is the center or surface of the feature.

Let $\delta(s_i, \theta)$ be the squared Euclidean distance between a given observation s_i and the feature (in this case a line segment) parameterized by θ. Note that $p(z_i|s_i, \theta)$ is convex due to our Gaussian model of human error. Thus, the EM formulation reduces to iterative least-squares over changing subsets of S within the neighborhoods of P_a, P_b. The raw human inputs P_a^0, P_b^0 are taken as the initial guess to the solution θ, and are successively refined of iterations of the EM algorithm to compute the final interpreted human input P_a, P_b.

Once P_a, P_b have been determined, along with observations S_a, S_b, we can find the poses responsible for those observations X_a, X_b, thus fully defining the human correction factor h. To make this process more robust to human error when providing corrections, a given pose is only allowed in X_a or X_b if there exist a minimum of T_p elements in S_a or S_b corresponding to that pose. The threshold T_p is used for outlier rejection of provided human corrections. It is empirically determined by evaluating a human's ability to accurately select points corresponding to map features, and is the minimum number of points a feature must have for it to be capable of being repeatedly and accurately selected by a human.

5.3 Solving HitL-SLAM

After interpreting human input, new pose estimates are computed in three steps. First, all explicit corrections indicated by the human are made by applying the appropriate transformation to X_b and subsequent poses. Next, any resultant discontinuities are addressed using Closed-Form Online Pose-Chain SLAM (COP-SLAM) [95]. And last, final pose parameters are calculated via non-linear least-squares optimization of a factor graph. The three-step approach is necessary in order to avoid local minima.

5.3.1 Applying Explicit Human Corrections

Although the user may select sets of observations in any order, we define all poses $x_i \in X_a$ to occur before all poses $x_j \in X_b$. That is, P_a is the input which selects observations S_a arising from poses X_a such that $\forall x_i \in X_a$ and $x_j \in X_b$, $i < j$, where X_b is defined analogously by observations S_b specified by input P_b.

Given P_a and P_b, we find the affine correction transformation A which transforms the set of points defined by P_b to the correct location relative to the set of points defined by P_a, as specified by mode m. If the correction mode is rank deficient, we force the motion of the observations as a whole to be zero along the null space dimensions. For co-orientation, this means that the translation correction components of A are zero, and for collinearity the translation along the axis of collinearity is zero. Figure 5.2 shows the effect of applying different types of constraints to a set of point clouds.

After finding A we then consider the poses in X_b to constitute points on a rigid body, and transform that body by A. The poses x_k such that $\forall x_j \in X_b$, $k > j$, are treated similarly, such that the relative transformations between all poses occurring during or after X_b remain unchanged.

5.3.2 Error Backpropagation

If $X_a \cup X_b$ does not form a contiguous sequence of poses, then this explicit change creates at least one discontinuity between the earliest pose in X_b, x_0^b and its predecessor, x_c. We define affine transformation C such that $x_0^b = A_{cb} C x_c$, where A_{cb} was the original relative transformation between x_c and x_0^b. Given C, and the pose and covariance estimates for poses between X_a and X_b, we use COP-SLAM over these intermediate poses to transform x_c without inducing further discontinuities.

The idea behind COP-SLAM is a covariance-aware distribution of translation and rotation across many poses, such that the final pose in the pose-chain ends up at the correct location and orientation. The goal is to find a set of updates U to the relative transformations between poses in the pose-chain such that $C = \prod_{i=1}^{n} U_i$.

COP-SLAM has two primary weaknesses as a solution to applying human corrections in HitL-SLAM. First, it requires translation uncertainty estimates to be isotropic, which is not true in general. Second, COP-SLAM deals poorly with nested loops, where it initially produces good pose estimates but during later adjustments may produce inconsistencies between observations. This is because COP-SLAM is not able to simultaneously satisfy both current and previous constraints. Due to these issues, we use COP-SLAM as an initial estimate to a non-linear least-squares optimization problem, which produces a more robust, globally consistent map.

5.3.3 HitL-SLAM Optimization

Without loop closure, a pose-chain of N poses has $\mathcal{O}(N)$ factors. With most loop closure schemes, each loop can be closed by adding one additional factor per loop. In HitL-SLAM, the data provided by the human is richer than most front-end systems, and reflecting this in the factor graph could potentially lead to a prohibitively large number of factors. If $|X_a| = n$ and $|X_b| = m$, then a naïve algorithm that adds a factor between all pairs (x_i^a, x_j^a), (x_i^a, x_j^b), and (x_i^b, x_j^b), where $x^a \in X_a$ and $x^b \in X_b$, would add $(m+n)^2$ factors for *every loop*. This is a poor approach for two reasons. One, the large number of factors can slow down the optimizer and potentially prevent it from reaching the global optimum. And two, this formulation implies that every factor is independent of every other factor, which is incorrect.

Thus, we propose a method for reasoning about human correction factors jointly, in a manner that creates a constant number of factors per loop while also preserving the structure and information of the input. Given a human correction factor $h =$

$\langle P_a, P_b, S_a, S_b, X_a, X_b, m\rangle$, we define c_m as the sum of three residuals, R_a, R_b, and R_p. The definitions of R_a and R_b are the same regardless of the correction mode m:

$$R_a = \left(\frac{\sum_{i=1}^{|S_a|} \delta(s_i^a, P_a)}{|S_a|}\right)^{\frac{1}{2}}, \; R_b = \left(\frac{\sum_{i=1}^{|S_b|} \delta(s_i^b, P_b)}{|S_b|}\right)^{\frac{1}{2}}.$$

As before, $\delta(s, P)$ denotes the squared Euclidean distance from observation s to the closest point on the feature defined by the set of points P. All features used in this study are line segments, but depending on m, more complicated features with different definitions for $\delta(s, P)$ may be used. R_a implicitly enforces the interdependence of different $x_a \in X_a$, since moving a pose away from its desired relative location to other poses in X_a will incur cost due to misaligned observations. The effect on X_b by R_b is analogous.

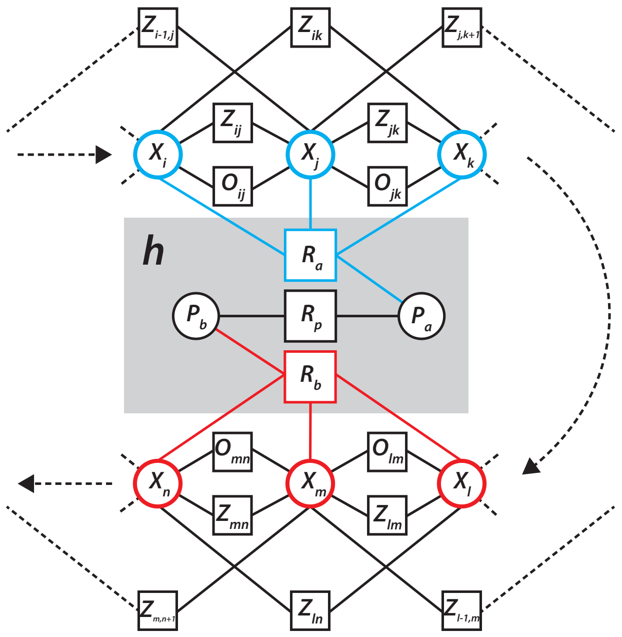


Figure 5.4: Subset of a factor graph containing a human factor h. Factors R_a and R_b drive observations in S_a and S_b toward features P_a and P_b, respectively. Factor R_p enforces the geometric relationship between P_a and P_b. Note that parameters in X_a (blue poses) and X_b (red poses) as well as P_a and P_b are jointly optimized.

The relative constraints between poses in X_a and poses in X_b are enforced indirectly by the third residual, R_p. Depending on the mode, colocation ($+$), collinearity ($-$), co-orientation parallel ($\|$), co-orientation perpendicular ($\perp$), the definition changes:

$$
\begin{aligned}
R_p^{+} &= K_1||cm_b - cm_a|| + K_2(1 - (\hat{n}_a \cdot \hat{n}_b)),\\
R_p^{-} &= K_1||(cm_b - cm_a) \cdot \hat{n}_a|| + K_2(1 - (\hat{n}_a \cdot \hat{n}_b)),\\
R_p^{||} &= K_2(1 - (\hat{n}_a \cdot \hat{n}_b)),\\
R_p^{\perp} &= K_2(\hat{n}_a \cdot \hat{n}_b).
\end{aligned}
$$

Here, cm_a and cm_b are the centers of mass of P_a and P_b, respectively, and $\hat{n}_a$ and $\hat{n}_b$ are the unit normal vectors for the feature (line) defined by P_a and P_b, respectively. K_1 and K_2 are constants that determine the relative costs of translational error (K_1) and rotational error (K_2). The various forms of R_p all drive the points in P_b to the correct location and orientation relative to P_a. During optimization the solver is allowed to vary pose locations and orientations, and by doing so the associated observation locations, as well as points in P_a and P_b. Figure 5.4 illustrates the topology of the human correction factors in our factor graph.

Note that HitL-SLAM allows human correction factors to be added to the factor graph in a larger set of situations compared to autonomous loop closure. HitL-SLAM introduces 'information' loop closure by adding correlations between distant poses without the poses being at the same location as in conventional loop closure. The off-diagonal elements in the information matrix thus introduced by HitL-SLAM assist in enforcing global consistency just as the off-diagonal elements introduced by loop closure. Figure 5.5 further illustrates this point – note that the information matrix is still symmetric and sparse, but with the addition of off-diagonal elements from the human corrections.

5.4 Results

Evaluation of HitL-SLAM is carried out through two sets of experiments. The first set is designed to test the accuracy of HitL-SLAM, and the second set is designed to test the scalability of HitL-SLAM to large environments.

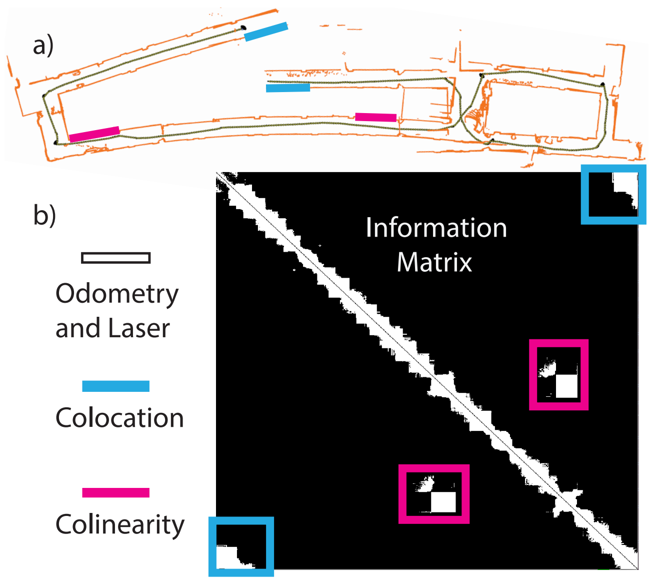


Figure 5.5: Example map (a) with corrections and resulting information matrix (b). The white band diagonal represents the correlations from the initial factor graph G_0. The colored lines on the map show the human correction input: colocation (blue) and collinear (magenta).. The constraints correspond to the blue and magenta off-diagonal entries in the information matrix.

To test the accuracy of HitL-SLAM, we construct a data set in a large room during which no two parallel walls are simultaneously visible to the robot. We do this by limiting the range of our robot's laser to 1.5m so that it sees a wall only when very close. We then drive it around the room for which we have ground truth dimensions. This creates sequences of "lost" poses throughout the pose-chain which rely purely on odometry to localize, thus accruing error over time. We then impose human constraints on the resultant map and compare to ground truth, shown in Figure 5.8. Note that the human corrections do not directly enforce any of the measured dimensions. The initial map shows a room width of 5.97m, and an angle between opposite walls of $4.1°$. HitL-SLAM finds a room width of 6.31m, while the ground truth width is 6.33m, and produces opposite walls which are within $1°$ of parallel. Note also that due to the limited sensor range, the global correctness must come from proper application of human constraints to the factor graph including the "lost" poses between wall observations.

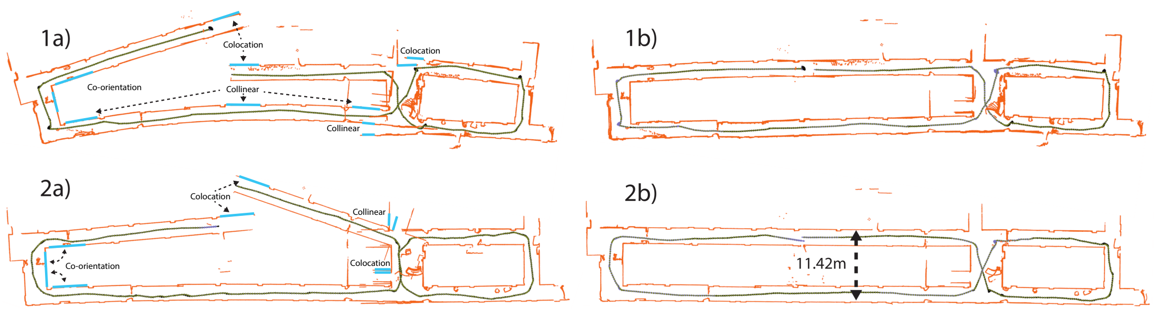


Figure 5.6: Initial and final maps from HitL-SLAM. Each map is of the same floor, and consists of between 600 and 700 poses. Maps in the left column (a) are initial maps, and maps in the right column (b) are final maps. Observations are shown in orange and poses are shown as arrows. Poses which are part of a human constraint are blue, while those which are not are in black.

To quantitatively evaluate accuracy on larger maps, where exact ground truth is sparse, we measured an inter-corridor spacing (Figure 5.6 2b), which is constant along the length of the building. We also measured angles between walls we know to be parallel or perpendicular. The results for ground truth comparisons before and after HitL-SLAM, displayed in Table 5.1, show that HitL-SLAM is able to drastically reduce map errors even when given poor quality initial maps.

We introduce an additional metric for quantitative map evaluation. We define the pair-wise inconsistency $I_{i,j}$ between poses x_i and x_j to be the area which observations from pose x_i show as free space and observations from pose x_j show as occupied space. We define the total inconsistency $\mathbf{I}$ over the map as the pair-wise sum of inconsistencies between all pairs of poses, $\mathbf{I} = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} I_{i,j}$. The inconsistency metric thus serves as a quantitative metric of global registration error between observations in the map, and allows us to track the effectiveness of global registration using HitL-SLAM over multiple iterations. The initial inconsistency values for maps LGRC 3A and 3B were $297.5m^2$ and $184.3m^2$, respectively. The final inconsistency values were $47.6m^2$ and $3.7m^2$, respectively, for an average inconsistency reduction of 91% relative to the initial map, thus demonstrating the improved global consistency of the map generated using HitL-SLAM. Figure 5.6 offers some qualitative examples of HitL-SLAM's performance.

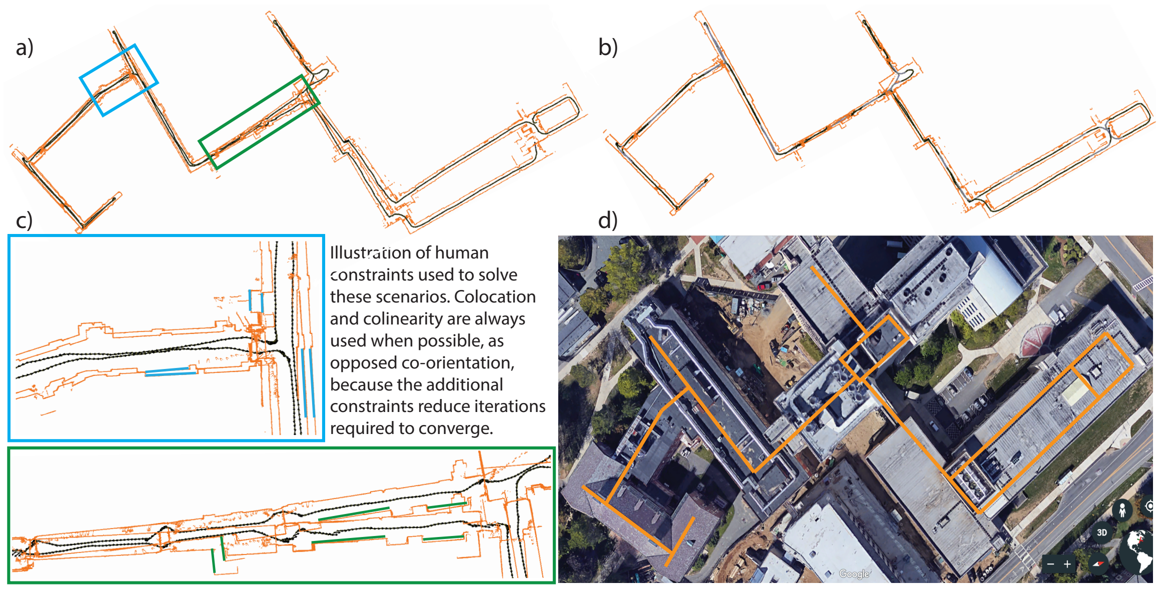

a)
b)
c)
d)
Illustration of human constraints used to solve these scenarios. Colocation and colinearity are always used when possible, as opposed co-orientation, because the additional constraints reduce iterations required to converge.

Figure 5.7: A large map a) corrected by HitL-SLAM b) using human correction, some of which are highlighted c). d) shows an approximate overlay of the map onto an aerial image of the complex from google earth. The map contains over 3000 poses.

To test the scalability of HitL-SLAM, we gathered several datasets with between 600 and 700 poses, and one with over 3000 poses and nearly 1km of indoor travel between three large buildings. Figure 5.6 shows some of the moderately sized maps, and Figure 5.7 details the largest map. 16 constraints were required to fix the largest map, and computation time never exceeded the time required to re-display the map, or for the human to move to a new map location.

Map	Samples		Input Err.		HitL-SLAM Err.	
	A	T	A(°)	T(m)	A(°)	T(m)
Lost Poses	10	4	3.1	0.07	1.0	0.02
LGRC 3A	14	10	9.8	3.3	1.5	0.06
LGRC 3B	14	10	7.6	3.1	1.1	0.02
BIG MAP	22	10	5.9	2.8	1.6	0.03
Mean	60	34	6.74	2.71	1.4	0.04

Table 5.1: Quantitative mapping errors using HitL-SLAM compared to ground truth, in the input maps, and after HitL-SLAM. The 'Samples' column denotes how many pairwise feature comparisons were made on the map and then compared to hand-measured ground truth. Angular (A) errors are in degrees, translation (T) errors in meters.

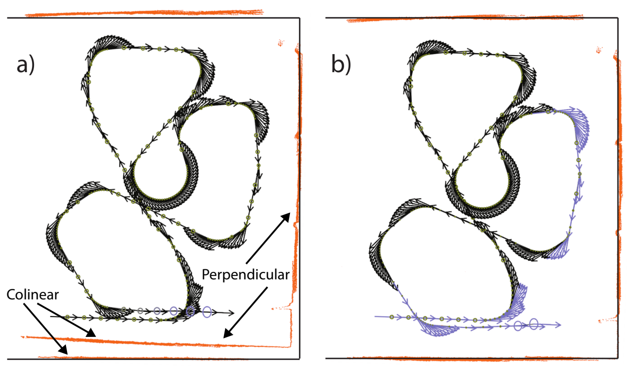


Figure 5.8: Initial a) and final b) maps for the 'lost poses' experiment. Observations are shown in orange, poses are black arrows, and ground truth (walls) is represented by the black lines. Poses involved in human constraints are colored blue.

All maps shown in 5.6 were corrected interactively by the human using HitL-SLAM in under 15 minutes. Furthermore, HitL-SLAM solves two common problems that are difficult or impossible to solve via re-deployment: 1) a severely bent hallway, in Figure 5.6 1a), and 2) a sensor failure, in Figure 5.6 2a) which caused the robot to incorrectly estimate its heading by roughly 30 degrees at one point. Combined, these results show that incorporating human input into metric mapping can be done in a principled, computationally tractable manner, which allows us to fix metric mapping consistency errors in less time and with higher accuracy than previously possible, given a small amount of human input.

5.5 Conclusion

This chapter introduced Human-in-the-Loop SLAM (HitL-SLAM), an algorithm designed to leverage human ability and meta-knowledge as they relate to the data association problem for robotic mapping. HitL-SLAM contributes a generalized framework for interpreting human input using the EM algorithm, as well as a factor graph based algorithm for incorporating human input into pose-graph SLAM. Future work in this area could proceed towards further reducing the human requirements, and extending this method for higher dimensional SLAM and for different sensing modalities.

As before, this method makes use of the fact that even if imperfect, all mobile robot platforms will run some version of SLAM. Thus, this method offers a solution to a common problem using commonly available data (SLAM output) and resources (humans), that is altogether significantly cheaper and more accurate than previously available options. This is largely made possible through the inherent generality of the factor-graph formulation of the SLAM problem. The source-agnostic nature of the factor-graph representation allows straightforward mixing of human factors alongside laser and inertial factors, which in other formulations may be quite convoluted. Not only does this property inform and allow the design of Multi-SLAM systems as shown in later chapters, but it also offers a rare counterexample to the typical tradeoff between generality over multiple types of task and performance on a single task.

One aspect of this work not highlighted in other chapters is the information interface between human and machine. Determining the form of information most readily provided by machines and interpreted by humans, and vice-versa is in general a very challenging problem. It often occupies an awkward no-man's-land between core autonomy research which focuses on fully autonomous capabilities and research on human-robot interaction, which often focuses on user experiences or perceptions. Similar to chapter 4, HitL-SLAM highlights another natural tension in robotics, this time between autonomy and performance. This is of course not always the case, especially in many complex control scenarios, but in planning cases that rely on strong priors or broad sets of knowledge and in almost all perception tasks, humans are vastly superior to fully automated systems. Generally speaking mixing human and machine capabilities leads to the best overall performance, but may not be desirable due to lack of scalability.

CHAPTER 6
ROBUST RANK DEFICIENT SLAM

Although simultaneous localization and mapping (SLAM) is a prerequisite for deploying autonomous mobile robots, the performance of SLAM systems can vary substantially depending on the environment. SLAM systems that use depth sensors are the preeminent choice for indoor scenarios due to the accuracy of modern sensors and the desire of many practitioners to build dense geometric models of the environment. In many cases, indoor environments present linear features such as line segments (2D) or planar facets (3D), which can be detected robustly [294]. Such features have many benefits, including ease of detection and quality of outlier rejection. However, they also present challenges, including correspondence calculation, optimization robustness, and long-term map quality. This chapter addresses these challenges individually and presents a state-of-the-art SLAM system for rank deficient constraints, which we call (RD-SLAM).

RD-SLAM is designed for environments that contain linear features and addresses two weaknesses inherent in dense iterative closest point (ICP) [35, 63] and correlative scan matching (CSM) [270]. First, ICP-based methods are not robust to outliers, while CSM cannot compute exact maximum likelihood transformations due to discretization. Second, neither algorithm is memory efficient online, typically requiring storage of raw point clouds or an occupancy grid. As robotic applications grow in scale and operate on ever lighter hardware, these representations become intractable. To address these problems, RD-SLAM extracts line segments or planar facets and computes relative transformations by calculating correspondences on these larger features. This is an especially clear example of specialization wherein some additional assumptions are made about the nature of the

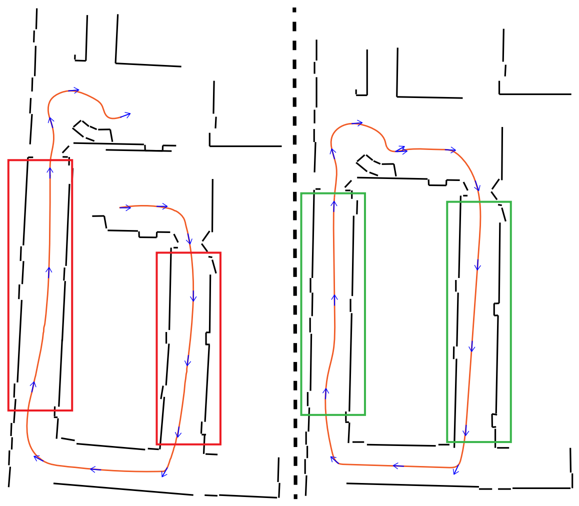


Figure 6.1: RD-SLAM on 2D laser data with (right) and without (left) prevention of optimization along degenerate axes, which are unconstrained directions detected as the null space of the set of visual constraints. Long straight hallways produce degenerate axes for some poses. Robot trajectory is in orange, and features in black.

available data (that it contains structures easily and reliably identified as planes or lines) and through these assumptions we can make improvements over very general algorithms like scan matching and ICP.

Given the lack of a distance metric between line segments or planar facets, we offer a set of algorithms and similarity functions for robust correspondence calculation. While linear features are easier to detect and track, using them in non-linear least-squares optimization problems can cause instability since correspondences may not fully constrain the robot's motion. We propose an algorithm for adding regularization terms to the optimization problem based on the approximate null space of sets of rank deficient constraints, and its effect is shown in Fig 6.1. These terms also reduce the sensitivity of the optimizer to the uncertainty models of different sensors, and in contrast to hard constraints may allow the optimizer to escape local minima. An existing method is also extended to construct and maintain highly compressed, maximum likelihood geometric maps to allow these maps to be updated online as the robot's trajectory estimate changes.

RD-SLAM is evaluated on real and simulated data experiments are presented examining the system's robustness to sensor noise, memory efficiency, compute load, and accuracy. Several ablation tests are also performed, including other popular methods for each contribution. Moreover, we show that the combined effect of improvements to correspondence calculations and co-linear factor design can lead to reductions in compute and memory load as well as decrease the frequency of large localization errors.

Metric SLAM is a well-studied topic of research and, broadly speaking, metric SLAM algorithms build either geometric reconstructions or maps of keypoints and landmarks. This chapter considers reconstructions of regular or man-made environments. Several existing SLAM algorithms have been designed for such environments. Many make strong assumptions, such as planar features appearing uniform [397, 72], completely rectilinear environments [68], or access to custom feature detectors [148]. Here, we make only the assumption that the environment contains line segments or planar facets that may be extracted from depth sensor data. In most deployment contexts this assumption is easily met.

Virtually all metric SLAM algorithms compute the affine transformation of the robot between successive frames, and many algorithms for computing these transformations are derivatives of the iterative closest point (ICP) method in that they compute correspondences and then minimize the distances between correspondences through optimization. ICP variants designed for specific environments or to address certain shortcomings of vanilla ICP include different methods of computing correspondences [64] or changing the minimization routine, such as by adding noise [279]. A survey of ICP algorithms is presented in [288]. Generalizations have also been established [192, 323], giving rise to algorithms using different physical primitives.

Both 2D and 3D RD-SLAM belong to a family of ICP-based methods which compute correspondences between geometric primitives such as line segments [10, 13], polylines [129], or planes [142]. One weakness of such approaches is their reliance on extraction of geometric objects and robust definitions of similarity or distance between objects in order to compute accurate correspondences. Fortunately, line segment extraction in 2D [264], and plane extraction in 3D [294], are mature areas of research. Various definitions for distance between line segments or between planar facets, which we extend here, have been explored in the context of line segment matching and are summarized nicely in [385].

Although many approaches use potentially rank deficient features, only a small number have investigated using co-linear constraints. Some approaches incorporate them into already constrained factor graphs [255], while most detect degeneracy online [134, 411, 66, 384, 155]. RD-SLAM takes the latter approach, but differs in that it does not explicitly project inertial measurements along degenerate dimensions or do any hard switching between sensing modalities. Instead, we detect degeneracies and add regularization terms to the existing optimization problem. An additional challenge when constructing long term maps is how to combine primitives that describe the same physical object. RD-SLAM follows the approach of Long Term Vector Mapping [248], which uses shape covariance matrix decomposition. A similar trick was presented in [335] under the term recursive least-squares. This paper extends these methods to work online in the event that primitives may need to be transformed when the underlying pose estimates change during optimization.

6.1 Rank-deficient SLAM

RD-SLAM does not describe a single trick or insight. Rather, this chapter describes a set of challenges and the corresponding types of approaches which result in functioning systems [233]. These challenges include choosing the right level of abstraction for feature detection and calculating feature correspondences §6.1.1, using rank deficient constraints robustly §6.1.2, and maintaining a consistent, high-accuracy, low-memory map in the context of online trajectory optimization §6.1.3.

6.1.1 Feature Correspondences

Both dense reconstructions from point clouds and keypoint-based systems typically compute correspondences. These computations are costly, can require non-trivial data structures, and often lack robustness. False correspondences are common and in the absence of advanced non-linear optimization techniques may cause catastrophic failure. Geometric primitives such as line segments and planar facets inherently mitigate some of these challenges in several ways.

First, we have fast, accurate, robust algorithms for line segment and planar facet detection [264, 294]. Second, because of the relatively low number of features detected per frame due to their inherent size, even naive correspondence calculations can be done quickly. Third, large feature size creates natural robustness to false correspondences, since the relative motion of the robot between frames is typically small compared to the distance between distinct features. However, such features present a unique challenge in defining similarity functions or distance measures. Since an established distance metric over $SE(2)$ or $SE(3)$ does not exist, similarity functions are typically constructed heuristically, often by summing or otherwise combining proper distance metrics defined over subsets of $SE(2)$ or $SE(3)$. Below, we present pseudometrics for robustly computing correspondences between line segments and planar facets.

6.1.1.1 Line Segments in 2D

We derive a measure for line-segment similarity (LSS) under the following assumptions. First, corresponding line segments extracted from successive scans should have similar location and orientation. And second, corresponding line segments do not need to be co-located; they may be only co-linear. Not only does co-linearity provide sufficient information as long as there are at least 2 non-parallel segments, but it is also more robust than co-location in some scenarios where this condition is not met, such as when travelling down a straight corridor, or when only part of the feature can be detected by the robot due to occlusion or range and field of view limitations.

Many formulae for LSS have been proposed [385], but none meet all of the criteria which follow from the assumptions above. Thus, we present a definition of LSS which is sensitive to the relative positions of segments anisotropically. Given line segments a and b, we define LSS(a, b) as

$$\text{LSS}(a,b) = \left((d_\theta/\tau_\theta)^2 + (d_\perp/\tau_\perp)^2 + (d_\parallel/\tau_\parallel)^2\right)^{\frac{1}{2}} \tag{6.1}$$

where d_* are different metrics over subspaces of $SE(2)$, and τ_* are scale factors based on sensor characteristics.

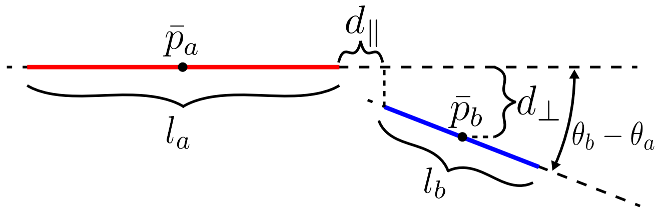


Figure 6.2: Variables for computing LSS between line segments a (red) and b (blue). This figure represents features existing in the x-y plane, essentially a top-down view of the robot and its environment.

Let line segments a and b have endpoints p_1^i, p_2^i, lengths l_i, centers of mass $\bar{p}_i$, and orientations θ_i for $i = a, b$, as in Figure 6.2. We define the d_* terms of LSS as follows.

$$d_\theta = |\sin(\theta_b - \theta_a)| \quad \text{and} \quad d_\perp = |(\bar{p}_b - \bar{p}_a) \cdot \hat{n}_a|, \tag{6.2}$$

where $\hat{n}_a$ is the unit vector normal to line segment a. Defining segment a such that $l_a \geq l_b$ allows $d_\perp$ to be symmetric. Distance in the parallel direction is non-zero if the projection of both p_1^b and p_2^b onto the line defined by segment a fall outside the boundaries of segment a. That is,

$$d_\parallel = \min(d_\parallel^1, d_\parallel^2), \tag{6.3}$$

where

$$d_\parallel^j = \begin{cases} t_j - l_a & t_j > l_a \\ 0 & 0 \leq t_j \leq l_a \\ |t_j| & t_j < 0. \end{cases} \tag{6.4}$$

Here,

$$t_j = (p_j^b - p_1^a) \cdot \frac{(p_2^a - p_1^a)}{||(p_2^a - p_1^a)||} \quad \text{where} \quad j = 1, 2. \tag{6.5}$$

6.1.1.2 Planar Facets in 3D

Planar facet similarity (PFS) can be computed robustly in a similar manner. Given two planar facets a and b, PFS is composed of similar terms.

$$\text{PFS}(a, b) = \left((d_\theta/\tau_\theta)^2 + (d_\perp/\tau_\perp)^2 + (d_\parallel/\tau_\parallel)^2\right)^{\frac{1}{2}}, \tag{6.6}$$

where d_θ and $d_\perp$ become the angle between normal vectors and the point to plane distance, respectively. One key difference is the definition of $d_\parallel$. Determining if a pair of planar facets overlap is expensive when considering the hull of each facet explicitly. Therefore, we represent each facet by an ellipse which we derive from the eigenvectors and eigenvalues of a shape covariance matrix constructed from the subset of the pointcloud corresponding to the planar facet, shown in Figure 6.3. Given ellipses a and b where $\delta_\theta(a, b) < \tau_\theta$, defined by centers $\bar{p}_a$, $\bar{p}_b$, eigenvectors e_1^a, e_2^a and e_1^b, e_2^b, and eigenvalues λ_1^a, λ_2^a and λ_1^b, λ_2^b, we project b onto a, producing sets of eigenvectors which are co-planar. Let these new eigenvectors and eigenvalues be $e_1^{b'}$, $e_2^{b'}$ and $\lambda_1^{b'}$, $\lambda_2^{b'}$, respectively. We define $d_\parallel$ as

$$d_\parallel = \max(0, (||\bar{p}_a - \bar{p}_b|| - ||\Gamma_a|| - ||\Gamma_b||)). \tag{6.7}$$

The equations presented below for Γ_* assume ellipse $*$ has been transformed so $\bar{p}_*$ is at the origin with $e_1^* \cdot \hat{y} = 0$.

$$\Gamma_* = \langle t\lambda_1^*\lambda_2^* \cos(\theta), t\lambda_1^*\lambda_2^* \sin(\theta)\rangle, \tag{6.8}$$

where

$$t = \left(\sqrt{(\lambda_1^*)^2 \cos^2(\theta) + (\lambda_2^*)^2 \sin^2(\theta)}\right)^{-1}, \tag{6.9}$$

and

$$\theta = \arctan(\bar{p}_y^*, \bar{p}_x^*). \tag{6.10}$$

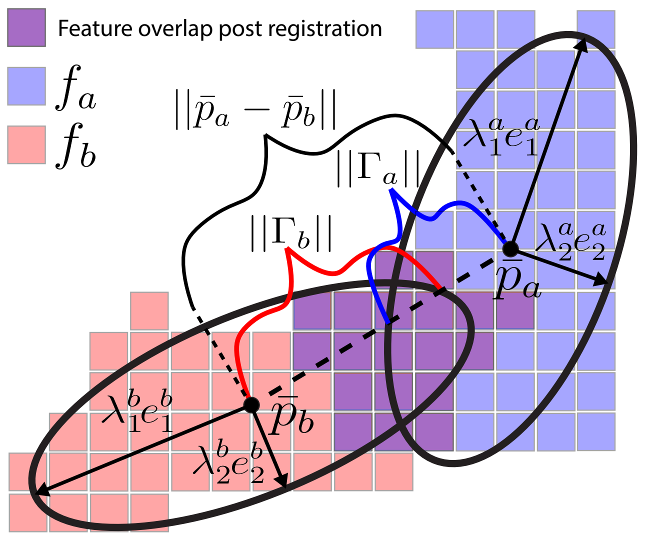


Figure 6.3: Ellipses for features a (blue) and b (red) detect overlap using the sum of projections (Γ_*) and the distance between feature centers. Purple patches show actual feature overlap.

In addition to the LSS and PFS pseudometrics, we also find that discarding features based roughly on size adds additional robustness. We discard line segments shorter than 20cm and discard planar facets with area less than 0.16m^2.

Because changes in viewing perspective cause the number of supporting observations to not always monotonically increase with the perceived size of the feature, and the fact that raw observations to geometric features is a many to one mapping, we find that length and area both robust measures of uncertainty as well as far cheaper to compute compared to models derived from models for individual depth observations. Calculating length is trivial in 2D, but for planar facets some methods, such as CAPE [294], find small planar sections of point clouds and then merge them to create the final facets. This has the advantage of extracting planar objects with arbitrary topologies, but can make fast area calculation via algorithms based on n-gon hulls challenging. In order to compute area quickly, we use the Varignon theorem to find the area of each planar section given its four corners, and then sum all planar sections that compose the entire facet, resulting in a lower bound on the area due to unaccounted for space between planar patches caused by the discrete nature of depth sensors. In practice, we find that taking the l_2-norm in feature descriptor space results in more robust correspondence matching and ultimately higher accuracy.

6.1.2 Dealing with Rank Deficient Constraints

Given a set of correspondences C, where $c_{a,b} \in C$ relates feature a to feature b visible at times t_i and t_j, respectively, we can generally write the optimization problem for visual features using co-linear or co-planar constraints as

$$X^* = \underset{X}{\operatorname{argmin}} \sum_{c_{a,b} \in C} d_\theta(a, A_{ij}b) + d_\perp(a, A_{ij}b). \tag{6.11}$$

where X^* is the MLE trajectory and A_{ij} transforms features observed at pose x_j to the frame of pose x_i. The obvious limitation of such a constraint is the potential lack of information along one or more axes. Such situations are common when sensor sampling density, field of view, or range decrease, limiting the number and informatic diversity of observed features. Computational constraints that force smaller optimization windows have a similar effect, since they lower the probability of making informative data associations.

Consider pose x_t and the set of correspondences C_t, where $\forall c_{a,b} \in C_t$, either a or b was observed at time t. Let F be the set of all features f such that $c_{f,*} \in C_t$. A scaled version of the second moment matrix is then

$$M = \sum_{f \in F} \hat{n}_f \hat{n}_f^T, \tag{6.12}$$

where $\hat{n}_f$ is the unit normal of feature f. Analytically, degeneracy can be detected by performing Gaussian elimination on M. However, noise in depth data results in feature normal estimates that almost always produce matrices that are technically full rank. This can cause the optimizer to find global minima with respect to $d_\perp$ that are not accurate, due to low signal to noise ratios in the null directions.

To solve this, we approximate the null space of M and apply constraints along all null directions. In contrast to other approaches, which use ‘hard’ constraints to prevent the optimizer from moving along null directions at all, we enforce these constraints as regularization terms, or ‘soft’ constraints, essentially constraining the optimizer to find a solution by moving only along well-conditioned directions to within some tolerance.

The method for approximating null(M) is shown in Algorithm 6. We analyze M’s condition numbers, κ, which, because M is normal, are computed from its eigenvalues. Large condition numbers indicate degenerate dimensions, and in our experiments we used $\tau_\kappa = 10.0$. In practice, τ_κ is easy to tune as most degenerate axes have condition numbers orders of magnitude higher than well-conditioned axes. For each pose within the optimization window, Algorithm 6 produces a set $\mathcal{N}_t$ that represents a basis of the null space of constraints on pose x_t. Regularization terms of the form

$$J(X) = \sum_{t=1}^{t_f} \sum_{\hat{\eta} \in \mathcal{N}_t} ((x_t - x_{t-1}) - (u_t - x_{t-1})) \cdot \hat{\eta} \tag{6.13}$$

are then added to the cost functions for each pose. Here, $\hat{\eta}$ are basis vectors describing the null space of visual constraints, u are the inertial measurements, and x are the pose variables. Combining equations 11 and 13 together, along with a cost function for the inertial measurements, we get an overall objective similar to

$$\begin{aligned} X^* = \operatorname*{argmin}_X \sum_{t=1}^{t_f} &||(x_t - x_{t-1}) - (u_t - x_{t-1})|| \\ + \sum_{c_{a,b} \in C} &d_\theta(a, A_{ij}b) + d_\perp(a, A_{ij}b) \\ + \sum_{t=1}^{t_f} \sum_{\hat{\eta} \in \mathcal{N}_t} &((x_t - x_{t-1}) - (u_t - x_{t-1})) \cdot \hat{\eta}. \end{aligned} \tag{6.14}$$

6.1.3 Map Updates

To store long-term representations of the environment we adapt Long-term Vector Mapping (LTVM) [248] to work online. Online LTVM checks newly detected features registered in global frame against an existing map of features, which is empty when the robot is first deployed. Each frame, statistical tests are performed which estimate the likelihood

Algorithm 6 NULL SPACE APPROXIMATION

```
1: Input: Set of features $F$, threshold $\tau_\kappa$
2: Output: Basis of null($M_t$), $\mathcal{N}_t$
3: $\mathcal{N} \leftarrow \emptyset$, $M \leftarrow [0]$
4: for $f \in F$ do
5:     $M \leftarrow M + \hat{n}_f \hat{n}_f^T$
6: $V \Lambda V^{-1} \leftarrow$ EIGENDECOMPOSITION($M$)
7: for $\lambda_i \in$ diag($\Lambda$) where $\lambda_i \neq \lambda_{\text{max}}$ do
8:     $\kappa \leftarrow \lambda_{\text{max}} / \lambda_i$
9:     if $\kappa > \tau_\kappa$ then
10:        $\mathcal{N}_t \leftarrow \mathcal{N}_t \cup V_{*,i}$
11: return $\mathcal{N}_t$
```

that a given feature corresponds to a physical entity already represented in the map. If the new feature represents an unobserved object it is added to the map. If it represents an observation of an already mapped object, the map feature is updated as a weighted sum of the new feature and the map feature, where the weight is the number of raw observations supporting each feature. We find that different statistical tests work well for different features. For line segments we use chi-squared tests as in [248], and for planar facets we use conservative thresholds of projections based on the elliptical representation.

Merging can also be performed between two features already in the map if their boundaries grow together. This process is expensive in the sense that it scales as $O(n^2)$, where n is the number of features in the map, since every mapped feature must be checked to see if it can merge with any other feature. However, in practice n is small since features are merged incrementally. Even for building-scale maps, n is typically in the hundreds or thousands. Moreover, space partitioning data structures such as kd-trees can eliminate most comparisons, providing further speedup.

One of the major strengths of LTVM is the maintenance of maximum likelihood feature location estimates along with a high compression ratio. This is possible via storing the shape covariance matrix representation of the supporting observations of each line segment or planar facet in a decoupled manner. However, features are extracted in robot frame, but the map updates are done in global frame. Thus, the decoupled representation must be rotated to global frame. We represent features using decoupled shape covariance matrices constructed from N depth observations p,

$$S = \sum_{i=1}^{N} p_i p_i^T - N\bar{p}\bar{p}^T = S_o - N\bar{S}, \tag{6.15}$$

where S_o represents the orientation of the feature, and $\bar{S}$ is the outer product of the feature's centroid, $\bar{p}$. Given an affine transformation defined by rotation R and translation T, we can compute the transformed matrices $\bar{S}'$ and S_o' as

$$\bar{S}' = (R\bar{p} + T)(R\bar{p} + T)^T \tag{6.16}$$

and

$$S_o' = N(R\frac{1}{N}S_o R^T - R\bar{S}R^T + \bar{S}'). \tag{6.17}$$

6.2 Results

We are mostly concerned with compute and memory efficiency and with accuracy and robustness given low quality sensing containing significant noise and visual artifacts. We hypothesize that under these constraints, algorithms from the RD-SLAM family, and specifically the improvements presented in this paper, offer advantageous tradeoffs compared to dense ICP methods as well as methods that deal with rank-deficiency by enforcing hard constraints on factors.

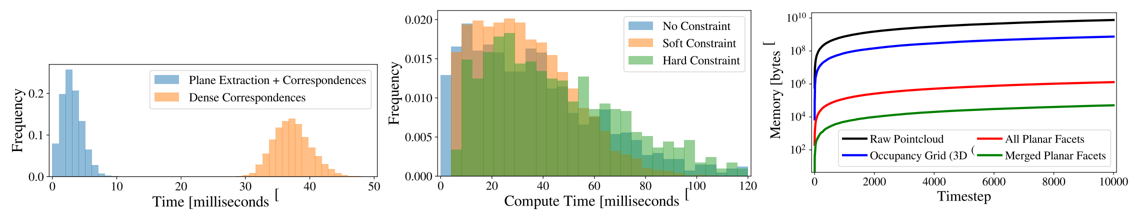
To test this hypothesis, we conduct several experiments using both simulated 2D lidar and 3D point cloud data and 2D and 3D data collected at the University of Massachusetts Amherst. We used a Hokuyo UST-10LX and an Asus Xtion PRO for lidar and point cloud data, respectively. Robots outfitted with these sensors were tele-operated around buildings at the university which include a number of straight hallways with limited features as well as some open areas with widths that exceed the sensor range in some places. Importantly, these data sets represent canonical human environments with large, easily observable features that contain only partial information. Moreover, many points in the trajectory cannot be fully constrained based on visual features. The robot often receives information that constrains only one positional axis, and this condition can persist for periods longer than the sliding window used for optimization which is typically 1 to 2 seconds. For each sensor, data was collected from 5 deployments, each taking a different path through the same environment. All timing experiments were done on a 3.70GHz quad-core processor.

6.2.1 Compute and Memory Efficiency

To test compute efficiency, we compare the time required for both correspondence calculations and pose optimization for RD-SLAM against dense ICP. We include the time required to extract features into the timing results for RD-SLAM, and we use a dense ICP implementation with a kd-tree for efficiently pruning non-correspondences. We find that RD-SLAM takes on average 7ms to produce 3D correspondences, while the ICP implementation requires 22ms on average. Time saved during optimization is also substantial. With an optimization window of 20 poses, RD-SLAM uses an average of 61ms to converge while point-to-point correspondences take on average 177ms.

To test memory efficiency, we compare the space required to store several different possible map representations using a simulated environment. Storing raw point clouds in 3D requires about 10MB per second. This grows unbounded over the deployment and is clearly infeasible. Creating a 3D occupancy grid with a resolution of 2cm requires nearly 100MB to explore our roughly 10m by 10m environment. Storing the map in an oct-tree reduces memory, but is still more expensive than planar representations. Storing all planar facets extracted during the deployment also grows without bound, but in our simulation it requires only roughly 1MB for every 10,000 poses. Lastly, merging planar facets incrementally allows the robot to store the entire map in about 10KB. Compute and memory savings are more pronounced for 3D data, but are still significant for 2D data.

Timing experiments, shown in Figures 6.4a and 6.4b, compare compute time for correspondence calculations and optimization with the proposed method against popular alternatives using the 3D point cloud data. Figure 6.4c shows memory use in 3D RD-SLAM



(a) Compute correspondences (b) Pose optimization (c) Memory consumption

Figure 6.4: Resource use for various RRD-SLAM sub-processes. (a) Time in milliseconds to compute correspondences. The vertical axis has been normalized to produce a probability density function. (b) Time in milliseconds to perform pose optimization. The counts have been normalized to produce a probability density function. (c) Memory required for mapping under various representations. Note the log scale on the vertical axis.

compared to other methods for storing maps. The compression effect of LTVM is less pronounced for 2D data, but still results in multiple orders of magnitude savings.

6.2.2 Accuracy

Figures 6.6c, and 6.6f show the effects of different optimization routines on accuracy using simulated 2D data. We use 2D simulations instead of 3D simulations because it is easier to construct more complex and realistic noise models. The main hypothesis is that soft constraints allow the solver more flexibility, which is beneficial in some scenarios, and the histograms illustrate how soft and hard constraints perform when optimizing co-linear constraints. The key takeaway is that although hard constraints are slightly more likely to produce very low error, they are also more likely to produce higher error, and in this respect soft constraints seem to increase the probability that a given location estimate will have error less than some ϵ, for sufficiently large ϵ.

Figure 6.5 shows a qualitative comparison between no constraints, hard constraints, and soft constraints on a real laser data set. We believe the increase in accuracy compared to the naive method (no constraints) is due primarily to the elimination of catastrophic localization failures, where the optimizer finds minima far from the ground truth. This behavior may be a natural consequence of optimization along directions with no real information or where the signal to noise ratio is very small, corresponding to the rank-deficient axes. We believe the decreased upper bound on error relative to optimizers using hard constraints is due to an entirely different phenomenon. In this case, soft constraints may open paths to minima with lower absolute values that are not accessible when using hard constraints, in some sense increasing the basin of convergence for some minima. This may be most beneficial when dealing with exceptionally noisy data that does not contain a strong signal.

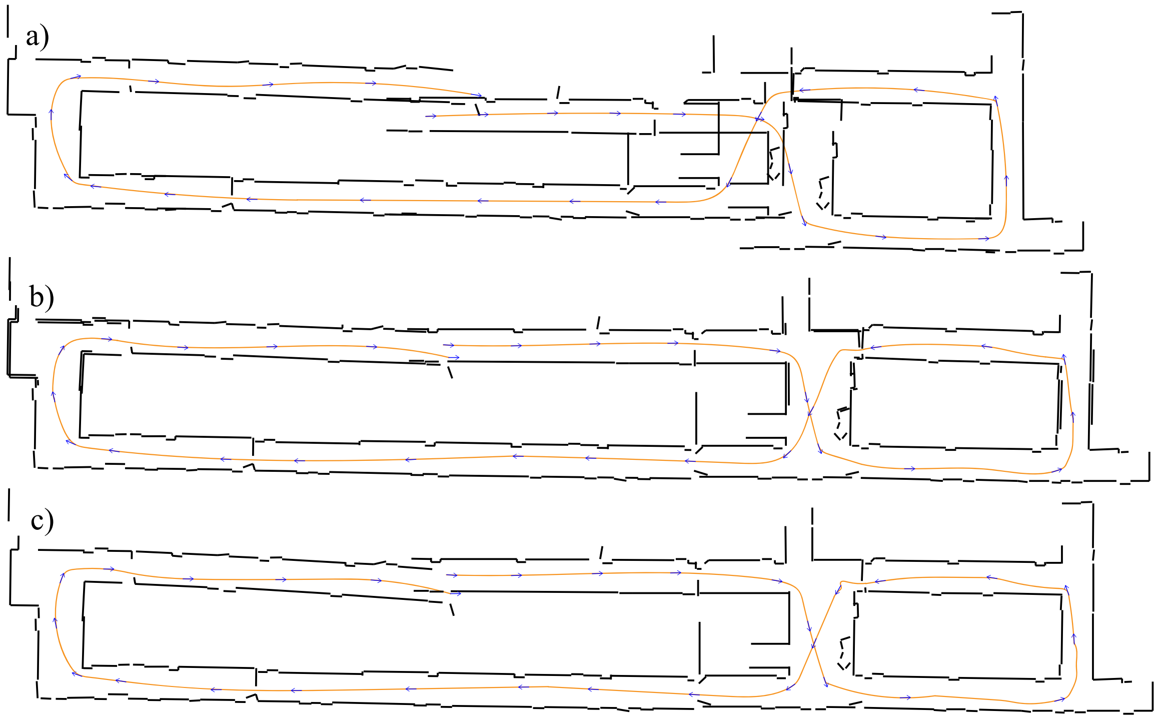
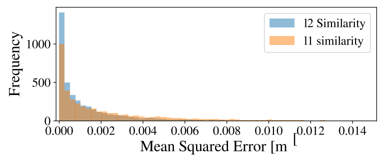


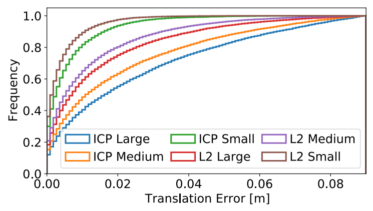
Figure 6.5: Maps produced using optimization over co-linear visual constraints. In a), no additional terms are added. In b), optimization along degenerate axes is prohibited. In c), regularization terms are added to discourage, but not prevent changes along degenerate axes.

6.2.3 Robustness

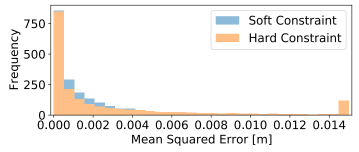
Figures 6.6b and 6.6e compare the accuracy of dense ICP and the proposed methods for computing correspondences between line segments under different levels of simulated noise. As expected, dense methods lack robustness to outliers in point-to-point correspondence calculations, and we see l_2 filtering increases robustness substantially. This is due to a small number of features during each deployment that erroneously pass each filter individually, but are not in reality reliable features. Decreasing the accepted range for



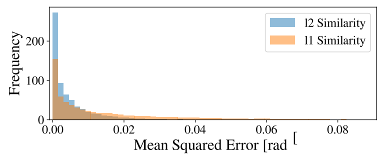
(a) Histogram of translation MSE w.r.t. pseudometric.



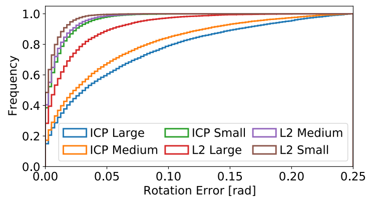
(b) Cumulative density function of MSE for translation.



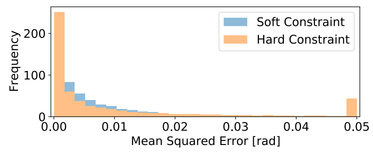
(c) Histogram of translation MSE w.r.t. constraint type.



(d) Histogram of rotation MSE w.r.t. pseudometric.



(e) Cumulative density function of MSE for rotation.



(f) Histogram of rotation MSE w.r.t. constraint type.

Figure 6.6: Performance characteristics for RRD-SLAM in simulated environments with several different noise levels and solving strategies. In sub-figures (b) and (e), labels Small, Medium, and Large denote noise levels, and labels ICP and L2 denote method.

correspondences also reduces this phenomenon, but at the cost of excluding many true correspondences.

6.3 Conclusion

This chapter presented several extensions to SLAM sub-systems which use constraints between large geometric features, resulting in a highly performant SLAM system for regular, man-made environments. We also demonstrated via ablation tests on simulated and real-world data that our extensions increase localization accuracy and reduce computation, memory use, and susceptibility to outliers. Moreover, this method offers a viable approach for getting the most out of low-quality or partial information by enabling the use of rank-deficient features.

This chapter again highlights the benefits of robust outlier rejection as well as the utility of generalizable constraints in pose-graph SLAM. It also provides the most prominent example in this thesis of the benefits of specialization. Because the target application is a robotic platform operating in the home, where there are many easily detectable planar surfaces, we may narrow the set of expected operating conditions and thus use sub-systems that more reliable and more accurately detect and reason about the types of stimuli that are most likely to be present.

The intended application and the motivation behind the development of this method for low-cost, mobile home robots also highlight some of the challenges faced by unimodal sensing suites. For example, other approaches to the problem of robust, all-conditions, low-cost, unstructured SLAM in a home environment use stereo cameras. While such systems work for some scenarios, many other common cases such as mirrors and other reflections, moving images on screens, or very low ambient light cause catastrophic failure, and these failure modes are often disjoint from the failure modes of RRD-SLAM. Additionally, constraints such as very cheap or limited sensing and highly constrained computation or memory can take a problem normally considered ‘solved’ into a new regime in which the standard ‘solutions’ are no longer possible. Together, these difficulties, which are largely results of the curse of ubiquity outlined in the chapter 2, further challenge the standard assumption of a ‘one-size-fits-all’ solution to SLAM and motivate the multi-SLAM approach outlined in subsequent chapters.

CHAPTER 7
LEARNING PERFORMANCE MODELS OF SLAM ALGORITHMS

One of the biggest challenges in deploying modern SLAM systems is that both the magnitude and frequency of localization failures are very hard to predict. Moreover, these failures can be difficult to recover from. If the maximum likelihood estimate for the current position $\mathbf{x}_t$ is several meters from the true location this can cause the robot to execute motor commands that may not be safe, or result in time spent on generating plans that are not feasible. It may also make subsequent estimates $\mathbf{x}_{t+1}$ less accurate since both filtering and smoothing techniques are initialized using the previous pose estimate, and thus poor initial estimates can lead to worse solutions in the future. While this problem is especially bad in Kalman filters, particle filters and non-linear optimization formulations are also affected. In some cases, such failures are worse than simply letting the robot operate using dead-reckoning until visual stimuli appear which afford more reliable localization.

The problem of predicting failures is made significantly more challenging in modern SLAM-solvers because these algorithms have large input spaces, large hyperparameter spaces, and a large set of possible internal states to which the algorithm may be initialized. These issues make SLAM and other robotic perception systems poor candidates for pre-deployment system verification [298]. Instead, run-time monitoring of perception subsystems has been proposed as a promising, tractable alternative. Generally, these prediction problems share some similarities, including their reliance on multiple streams of data, both raw signal data and states of computation internal to SLAM-solvers. However, signs of impending success or failure may differ substantially depending on the sensing modality.

The key idea advanced in this chapter is to exploit this similarity between many related SLAM performance prediction tasks, where the front-end feature extraction methods differ with respect to the data they operate on, but not the basic information they produce and interface with. Specifically, we highlight two contributions. First, we show that exploiting the independence of front-end and back-end pose-graph SLAM modules in order to run performance predictions that isolate front-end outputs allows us to train more generalizable models that are robust to different back-end optimization schemes since these prediction systems are largely tasked with evaluating the quality of the overall set of constraints produced by the front-end, a task that likely transfers across many modalities. Second, we show empirically that certain neural network architectures seem to be well-suited for performance prediction of streaming perception problems, regardless of sensor modality, potentially simplifying the implementation of such systems on other platforms in the future.

We demonstrate the potential of this approach on the KITTI data set [109], where we find that similar neural architectures, in particular convolutional structures that operate on several images at once, are generally well-suited for this task. We also present results on how different data availability and loss functions affect model performance. Notably, this is a regime in which common training practices such as data augmentation may be unreliable, since the underlying process being modeled (SLAM algorithm performance) does not necessarily change smoothly as a function of the raw input.

Developing models of performance for different algorithms has received considerable attention in various sub-fields of computer science for some time, and is often referred to as meta-learning [376, 372]. However, in robotic perception, and in SLAM in particular, the existing literature is relatively limited. Generally, approaches to this problem vary along four primary axes. First, whether predictions are made offline, when the operating environment is known but prior to deployment [284, 209], or online as the robot is operating, typically on a frame-by-frame basis [55, 297, 9, 8]. Second, whether performance is modeled as a binary 'failure' and 'nominal' classification problem [55] or whether it is modeled as a regression problem, either continuous or discrete [297, 284, 209, 9, 8]. Third, whether the predictions are made regarding the entire algorithm [55, 284, 209, 8] or with respect to specific subsets of the input [297]. And fourth, whether these performance models use hand-engineered features [55, 284, 209, 8] or process raw data, usually with neural networks [9].

Unlike many other computational systems, SLAM systems have several properties that make theoretical bounds on performance, including those for cooperative systems [240], difficult to rely on in practice since these bounds are often only possible under some exceptionally strong assumptions, such as completely known sensor noise distributions. This is primarily due to three reasons. First, the potential input space is extremely large, being the cross product of multiple high-dimensional sensor signals, a large set of possible hyperparameter settings for each algorithm, and a large set of possible initialization configurations. Second, small changes in any one of these factors may induce large changes in the quality of the resultant output. And third, sensor noise is very difficult to model precisely. Thus, learning-based solutions for modeling the performance of SLAM systems using neural networks, originally introduced as 'introspective' systems [75], were proposed to overcome the shortcomings of theoretical analysis of such complex systems.

There are also approaches more aligned with software engineering philosophy and based loosely on the concept of coverage testing, which ultimately identify a similar set of complicating factors, including environmental characteristics, sensor characteristics, and robot motion characteristics. One such approach attempts to partition the SLAM system input space into equivalence classes in order to efficiently generate test cases for these classes and achieve high coverage [353]. However, the premise of such approaches is often that SLAM system failures are the result of deficient code. Here, we argue this is not the primary source of errors in state-of-the-art systems; rather, it is the result of applying fundamentally limited algorithms to extremely complex data.

For this reason, most approaches focus on prediction. Offline prediction about the efficacy of different sensor payloads or perception algorithms as typically been researchers' and practitioners' purview, but recently there have also been proposals to do this automatically using regression [284, 209]. However, these approaches often require knowing a significant amount about the deployment environment, or event the planned route, in order to operate, which severely limit their utility in practice. One strength of such approaches though is their inclusion of features from both environment and robot.

Even when confined to online prediction, the notion of 'performance' for SLAM systems has taken several forms. Some approaches, such as IV-SLAM [297], predict the quality of features derived from certain subsets of the raw data to guide SLAM feature extraction to select more features from parts of sensor data that are likely to be higher quality; or Muca-SLAM [162] which predicts which camera from a set of multiple cameras will contain the best features. Other approaches treat performance as a binary variable, predicting either failure or nominal behavior in a binary classification task, for example using support vector machines [55]. These approaches may work well for specific applications or SLAM systems, but their general formulation does not necessarily cover many of the instances where performance prediction may be useful.

Perhaps the most general, SLAM-algorithm-agnostic, notion of performance is the magnitude of localization error for a given pose. Some have proposed estimating this error using learning and then directly correcting pose estimates by the predicted error [9]. Theoretically, if such a system were perfect, it may be the easiest way of incorporating performance prediction into existing robotics architectures since it does not require any additional processing to be useful and can be applied internally within a given SLAM system, without the requirement to interface with any other component of the stack. However, such a such a system must predict not only the magnitude of errors, but also the correct direction, transforming an already challenging one-dimension regression problem in the positive real numbers into, at the very least, a two-dimensional problem over all reals. Such an approach would become even more difficult to implement as the space of state estimates increases, for example to include vertical displacement or roll and pitch in addition to yaw.

A slightly more relaxed version of the problem, and one which is closest to our approach in this chapter, is to simply predict the magnitude of the error, without attempting to predict the exact error itself. Previously, this approach has been attempted for *entire trajectories* using random forest regression trained on a set of global features generated from raw (camera) sensor data passed through 1-D pooling [8]. The procedure we outline in this chapter has several benefits compared to this approach, including the ability to generalize to other types of sensor data without the need for custom features, and the ability to predict error frame-by-frame. Integration of similar predictive models into existing pose-graph optimization systems may be possible by predicting counterfactual outcomes where subsets of available features are omitted or down-weighted during optimization, though we leave confirmation of such hypotheses for future work.

7.1 Learning Performance Models of SLAM Algorithms

This section provides an overview of different design and engineering considerations regarding the general practice of applying convolutional neural networks to learning performance models of SLAM systems. Although there is no singular algorithm that is guaranteed to outperform all others, there are nonetheless several characteristics of the SLAM performance prediction problem that make certain learning setups more likely to be successful, which we focus on below.

At the lowest level, this is a regression task, where we wish to learn a function $f(\mathbf{z}, \mathbf{w})$ that predicts, for a given observation $\mathbf{z}$ and internal state of computation $\mathbf{w}$, the expected magnitude of the translation error $\varepsilon_{trans} = |\hat{\mathbf{x}}_t - \mathbf{x}_t|$ and rotation error $\varepsilon_{rot} = |\hat{\theta}_t - \theta_t|$ of a SLAM algorithm at each time step t. Regression is a notoriously difficult task for neural networks, so we instead model this task as a classification task, where the different classes represent different ranges of possible error.

Because the distribution of errors from SLAM systems is not uniform (often modeled as a half-normal distribution or a mixed distribution with a half-normal component), using classes that represent equally sized subsets of the positive real space of errors will create an imbalanced data set. Thus, we create a more or less balanced set using the quantile function of the half-normal distribution

$$Q(\varepsilon) = \sigma\sqrt{2}\,\text{erf}^{-1}(\varepsilon) \tag{7.1}$$

where $\text{erf}^{-1}(\varepsilon)$ is the inverse error function such that $\text{erf}(\text{erf}^{-1}(\varepsilon)) = \varepsilon$, and

$$\text{erf}\,(\varepsilon) = \frac{2}{\sqrt{\pi}}\int_0^{\varepsilon} e^{-t^2} dt. \tag{7.2}$$

We plot $Q(\varepsilon)$ using a numerical approximation of the inverse error function and a rough estimate of the standard deviation σ of the SLAM error to determine the regions of error space that each class represents. For a total of 5 classes, we define the following class boundaries: $0 < \varepsilon \leq 0.2\sigma < \varepsilon \leq 0.5\sigma < \varepsilon \leq 0.9\sigma < \varepsilon \leq 1.4\sigma < \varepsilon$. The value of σ will of course depend on the SLAM algorithm used for training. In Figure 7.1 we show this breakdown on a simulated data set for $\sigma = 1$.

7.1.1 The KITTI Data Set

The KITTI data set [109] is a large data set used for many SLAM algorithms tested in service of autonomous driving. It contains 11 total trajectories that include ground truth in both urban and rural settings. Each trajectory contains several minutes of driving and thus hundreds or thousands of individual pose estimates. For camera data we simply use the original raw image, and for LiDAR data we project the returns down to the xy-plane and use color to represent the height of the return. If there are multiple returns in the same pixel, we take the average height.

In addition to the raw sensor data we use two other sources of information, also represented as images. The first is a snapshot of the optimization problem, called the information

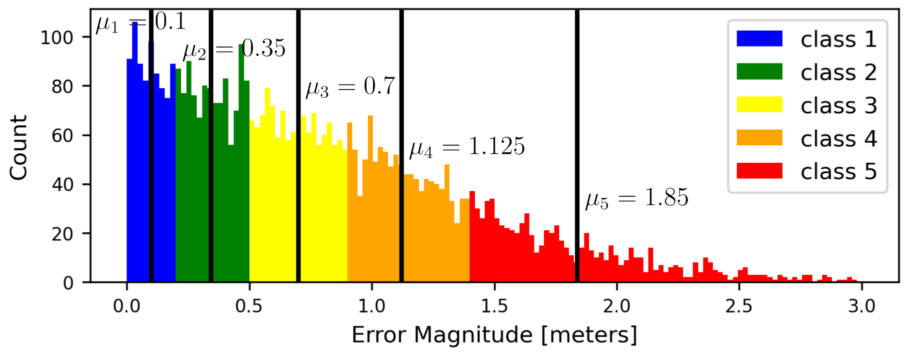


Figure 7.1: Example labeling scheme for error magnitude classes in order to promote more balanced classes. Here, errors falling into different colored bins are assigned different class labels y, and the boundaries of the class bins are determined by analyzing the quantile function of the half-normal distribution, which roughly models SLAM error probability density when deployed in environments close to those intended by developers. For the purposes of computing loss we use the mean of each bin, here denoted by black vertical bars. The distribution shown has a standard deviation of $\sigma = 1.0$, and during training the values μ_i are scaled up or down depending on σ in the real data.

matrix. The information matrix represents which poses have direct constraints with respect to which other poses. For example, if pixel i, j is highlighted, it means that there exists at least one factor linking pose x_i with pose x_j in the factor graph. Since the solvers we are predicting over are solving fixed-size problems, where variables are added or deleted from the problem as the robot experiences more data, it is possible to form a fixed-size input from such a setup. Here, we up sample to match the resolution of the other data.

The second is memoized data from the SLAM front-end. This takes the form of feature locations and feature descriptors, in the image plane (for cameras) or projected into the xy-plane and represented as an image (for LiDAR). Here, the exact format of the data depends on the information contained in the features extracted from the raw signal. Since the modalities we test on produce features in all three dimensions, we again use color to represent height. When there is additional information, such as scale or orientation from ORB features [311] we encode it using the RGB channels of the image. For example, orientation can be encoded as a unit vector within the red and green channels, while scale can be encoded in the blue channel.

7.1.2 Network Architecture

Because our input is in image format, both raw sensor data as well as memoized data such as feature discriptors, we use convolutional architectures for all modalities. Several have already been proposed for regression tasks in the literature, although with very different applications [320, 213]. Although the KITTI data set is large by robotics standards, it is not large enough to comfortably train a convolutional neural network from scratch. Moreover, there is no obvious way to perform automatic data augmentation on the KITTI data set. Thus, we experiment with two pre-trained models, AlexNet [177] and Inception V2/V3 [349], and fine tune these models on the performance data from running SLAM algorithms on trajectories from the KITTI data set. These architectures were originally trained on the ImageNet [315] data set, which has a large number of classes.

Normally, we could simply replace the last linear layer with our own linear layer of appropriate size, for example five nodes for five classes, as is common in other domains [1]. For example, given a penultimate layer of 2048 nodes, we would have a final fully connected layer represented by a 5×2048 weight matrix. However, because we are using three separate images (raw data, feature locations, information matrix), we instead need our fine-tuning layers to use all three 2048-dimensional latent representations. Thus, we add two fully connected layers for fine tuning, one from 6144 to 2048, where the input is formed by concatenating the latent representations from each individual image, and then final one from 2048 dimensions to 5, representing the error classes. Finally, we need 10 output dimensions with two groups of 5 normalized independently in order to classify both rotation and translation errors. Note it is also possible in theory to extract intermediate feature representations from the neural networks and then train an SVM or other classifier on the intermediate representations, though we do not empirically test such a setup.

7.1.3 Loss Function Design

In designing the loss function there are three main considerations. First, we need to preserve the underlying regressive nature of the task. To do this, we penalize incorrect classification proportional to the squared error in error prediction. For example, given predicted labels ℓ_{trans} and ℓ_{rot} and the mean values of those classes $\mu_{\ell_{trans}}$ and $\mu_{\ell_{trans}}$, along with the mean of the ground truth classes $\mu_{y_{trans}}$ and $\mu_{y_{trans}}$, we can write an initial loss function as

$$\mathcal{L}_0 = |\mu_{\ell_{trans}} - \mu_{y_{trans}}|^2 + |\mu_{\ell_{rot}} - \mu_{y_{rot}}|^2 \tag{7.3}$$

Here, because the classes represent regions of the non-uniform error distributions, we do not take the midpoint of the bins. Instead we take the mean of the segment of the distribution weighted by the relative probabilities of achieving the different possible error values along that segment, as shown in Figure 7.1.

Second, we need the loss function to be asymmetric, since in some applications it is more important to conservatively estimate the localization error than to be as accurate as possible. That is, we would rather minimize a weighted notion of least squared error where weights are higher for overly optimistic predictions. To represent this in the loss function, we define

$$\mathcal{L}_1 = \lambda(\mu_{\ell_{trans}}, \mu_{y_{trans}})|\mu_{\ell_{trans}} - \mu_{y_{trans}}|^2 + \lambda(\mu_{\ell_{rot}}, \mu_{y_{rot}})|\mu_{\ell_{rot}} - \mu_{y_{rot}}|^2 \tag{7.4}$$

where

$$\lambda(\mu_\ell, \mu_y) = \begin{cases} 1 & \text{if } \mu_\ell \leq \mu_y \\ w & \text{otherwise} \end{cases} \tag{7.5}$$

and $w > 1$. Note that functions that minimize $\mathcal{L}_0$ also minimize $\mathcal{L}_1$. However, given that even with large data sets performance predictors may still be imperfect, $\mathcal{L}_1$ biases errors that do occur to be, generally speaking, less risky for the overall system.

The third and final consideration is that these performance predictions may be used as inputs to planners, as discussed in chapter 8, and thus some notion of the quality or uncertainty regarding the prediction could be beneficial. Models that output a distribution over class labels that accurately reflects the probability of each label being the true label are said to be calibrated [119]. For example, if on some subset of the data our model predicts errors of a given magnitude $y = \xi$ with probability 0.1, then if it is well-calibrated, roughly one out of 10 such labels will be $y = \xi$. There is significant work on calibration, and although in practice many times simply applying a softmax activation function to the final layer and interpreting the output as a probability distribution over layers can provide reasonable calibration [131, 277], we opt to apply a more explicit cross-entropy loss.

However, we also want to weight our loss by the magnitude of the error, similar in spirit to other weighted cross-entropy losses [136]. Thus, we have a third possible loss function

$$\mathcal{L}_2 = -\sum_{i=1}^{5} \mathcal{L}_1(\ell_i, y_i)[y_i \neq c_i] \ln(\vec{\ell_i}) \tag{7.6}$$

where $\vec{\ell_i}$ is the ith entry in the softmax output vector and square brackets represent Iverson brackets, evaluating to 1 if the inner statement is true and 0 if it is false.

A priori, it is not obvious which loss function will lead to the best performance within the larger multi-SLAM system, but given that the decision-making model (a partially observable Markov decision process) has the ability to model and reason about the reliability of its observations, then it is likely that as long as there is a baseline of reasonable performance, having a well-calibrated performance model will allow the higher level planner to most reliably exploit the performance models predictions.

7.1.4 Training Procedure and Hyperparameters

Given that we use pre-trained networks, our focus is on fine-tuning to our performance prediction task. The main hyperparameters we might consider changing from their original settings during fine-tuning are the learning rate, learning schedule, batch size, and momentum. For a more in-depth treatment of different considerations for various hyperparameters during fine-tuning, see Li et al. [190] and the references therein.

To simplify this process, we use the gradient descent method ADAM [170], which automatically adjusts learning rate and operates with a similar concept to momentum, subsuming these components of the hyperparameter search. It is possible that individually setting these hyperparameters can achieve superior performance. Last, as our fine-tuning data is somewhat limited, we set the batch size to be 16.

When conducting our experiments we train several instances of our models over different subsets of the KITTI data using k-fold cross validation where $k = 5$. In this setup, of the 11 trajectories, in each fold 2 are held out for testing, 3 used for validation, and the remaining 6 used for training. Thus, all numbers reported below are the average of evaluations across all 5 models, unless otherwise noted.

One drawback of training models for each SLAM system independently is that to support a large portfolio of SLAM front-ends, we would need a large number performance prediction models. One topic for future consideration is whether or not we could reduce this burden by training models that can generalize across multiple feature (or descriptor, or detector) types, essentially reducing the space our performance models need to cover from all sensor $\times$ feature combinations to just the set of different modalities, which is a much more limited domain. Similarly, models learned with respect to specific sensor suites may be able to be generalized to similar suites, all of which may be used with a particular SLAM algorithm. For example, a camera-based system may use many different types of cameras without requiring a different front-end system.

7.2 Results

At a high level this is a relatively straightforward classification task, in which we expect the learned function to accurately predict the approximate magnitude of error in location estimates. Therefore we evaluate our set of SLAM performance predictors primarily by understanding their accuracy in multi-class classification. However, there are also several additional axes of analysis that are important beyond raw accuracy. First, the magnitude of the mis-classification is still important, so in some sense the original regressive nature of the task cannot be ignored. Second, it is typically more costly to underestimate localization error than to overestimate it, since the robot will often have the option to take additional measurements to gain information about its location, or have auxiliary methods or resources it can draw on to reduce uncertainty, such as human monitors or backup localization methods.

To address these complicating factors, we also introduce modified definitions of precision and recall, where, for each sample, classes representing the correct error magnitudes and those representing larger errors are aggregated to create the 'positive' class, and classes representing smaller errors are aggregated into a 'negative' class, creating a binary classification problem. The typical definitions of precision and recall are then applied to these new class labels, thus there are no 'true negatives'. Finally, we compute the correlation of the of the performance predictions with the actual errors by considering each class label as indicating the weighted mean-value of the range of errors that class represents.

We test the efficacy of these performance models using two open-source SLAM systems: ORB-SLAM2 [243] which uses stereo camera data, and CAE-LO [400] which uses LiDAR data. While ORB-SLAM2 and CAE-LO are not the leading edge of state-of-the-art systems, ranked 65th and 41st on the KITTI benchmark, respectively, they do provide reasonably performant open-source systems. Moreover, both methods are indirect, meaning that they have explicit feature extraction steps and thus provide more easily accessible memoized data for use in learning performance models. Critically, they are also relatively

Model	Accuracy	Precision	Recall
ORB-SLAM2 (AlexNet)	**0.62** $\pm$ 0.10	0.75 $\pm$ 0.10	0.91 $\pm$ 0.04
ORB-SLAM2 (Inception)	0.57 $\pm$ 0.08	0.71 $\pm$ 0.06	0.88 $\pm$ 0.03
CAE-LO (AlexNet)	0.42 $\pm$ 0.13	0.66 $\pm$ 0.11	0.72 $\pm$ 0.06
CAE-LO (Inception)	**0.55** $\pm$ 0.08	0.68 $\pm$ 0.05	0.74 $\pm$ 0.06

Table 7.1: Model performance when predicting translation errors.

Model	Accuracy	Precision	Recall
ORB-SLAM2 (AlexNet)	0.69 $\pm$ 0.08	0.87 $\pm$ 0.07	0.89 $\pm$ 0.04
ORB-SLAM2 (Inception)	**0.70** $\pm$ 0.07	0.90 $\pm$ 0.03	0.88 $\pm$ 0.03
CAE-LO (AlexNet)	**0.61** $\pm$ 0.11	0.78 $\pm$ 0.06	0.82 $\pm$ 0.05
CAE-LO (Inception)	**0.61** $\pm$ 0.10	0.77 $\pm$ 0.03	0.83 $\pm$ 0.04

Table 7.2: Model performance when predicting rotation errors.

close in overall performance, and use different modalities. While it is not always the case that systems differ in performance most when they rely of different sensors, it is nonetheless a very common occurrence in practice.

7.2.1 Performance Prediction Accuracy, Precision, and Recall

Tables 7.1 and 7.2 summarize the main results on accuracy, precision, and recall of performance prediction for translation and rotation errors of different SLAM systems using different modalities and neural architectures. A random classifier would have accuracy 0.2. Numbers and bounds given in the tables are averages and standard deviations over all k folds of the data set. There are several key takeaways.

First, different SLAM algorithms with different modalities do exhibit some differences with respect to how accurately we can predict their errors. This could be due to several factors, including the reliability of the underlying SLAM algorithm or the variety of sensing conditions present for each modality in the training data set. Our best hypothesis for this gap is that the pre-trained networks were trained specifically for images similar to what is observed by the stereo pair used in ORB-SLAM2 and that although we transform data from CAE-LO into image format, the pre-trained networks are naturally better suited to find patterns in raw image data moreso than images constructed from memoized data. Second, we find, somewhat surprisingly, that both networks perform relatively equally, with the most noticeable difference being the slightly lower variance in performance (generally speaking) of the Inception architecture.

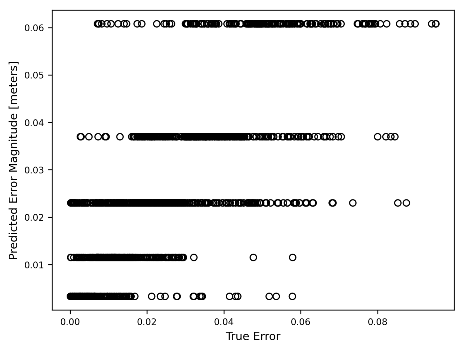
Third, rotation errors seem easier to predict across both algorithms, although this effect is especially pronounced for ORB-SLAM2. This is encouraging since much of the absolute error in pose estimates comes from errors in rotation estimates that are then compounded into location estimate errors as the robot continues traveling. Eliminating, or at least being aware of these points in the trajectory could greatly enhance SLAM outcomes. That said, though the trajectories in the KITTI data set average roughly 5 right-angle or sharper turns, much of the data set involves driving relatively straight which typically gives rise to lower rotation error estimates. Finally, recall seems generally higher, which is expected since the loss function penalizes overly optimistic error estimates asymmetrically.

7.2.2 Correlation with Actual Error

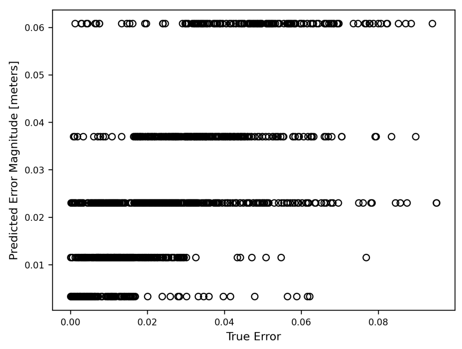
While some minimal level of accuracy is required to be able to use such performance models, for example within multi-SLAM systems, here we offer some additional metrics regarding the magnitude of errors in performance prediction. Figure 7.2 shows how predicted and actual errors correlate on one set of test data. Note that we did not specifically train any networks for continuous-valued regression. Sub-figure (f) shows the best performing model: the Inception network predicting rotation errors in ORB-SLAM2. What is particularly compelling in this instance is relatively distinct groupings of actual error across the different predicted classes compared to other models.

7.2.3 Architecture and Input Ablation Studies

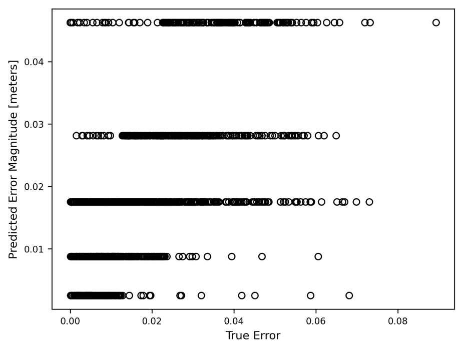
While the space of all possible convolutional network architectures is far too large to test exhaustively, we do investigate performance on several alternate training setups. In particular, we run ablation tests where models are trained using A) different information,



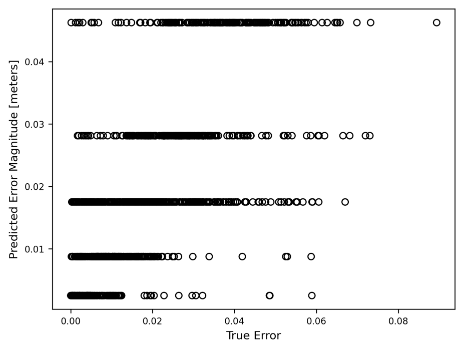
(a) ORB-SLAM2 (AlexNet)



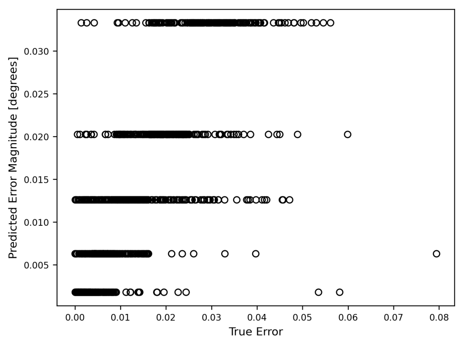
(b) ORB-SLAM2 (Inception)



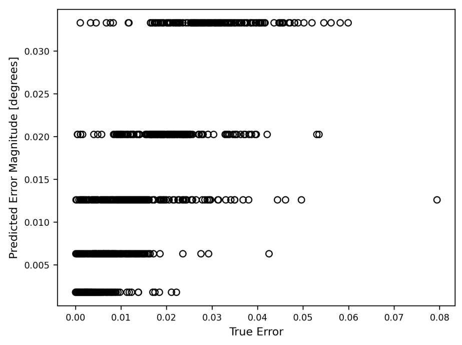
(c) CAE-LO (AlexNet)



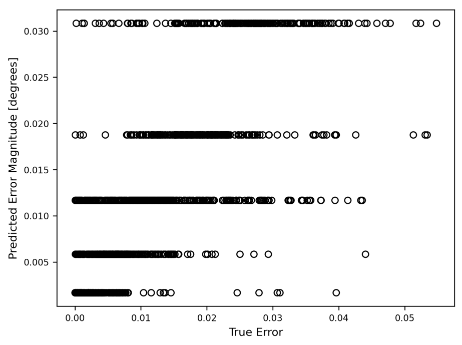
(d) CAE-LO (Inception)



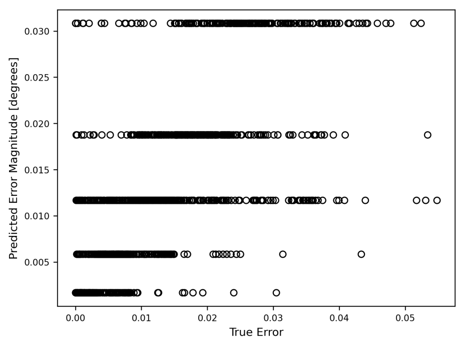
(e) ORB-SLAM2 (AlexNet)



(f) ORB-SLAM2 (Inception)



(g) CAE-LO (AlexNet)



(h) CAE-LO (Inception)

Figure 7.2: Scatter plots of actual and predicted errors in translation (a)-(d) and rotation (e)-(h).

Ablation Test	Acc. (t)	Prec. (t)	Recall (t)	Acc. (r)	Prec. (r)	Recall (r)
Unmodified	0.62	0.75	0.91	0.69	0.87	0.89
Raw Data Only	0.51	0.62	0.65	0.57	0.64	0.77
Features Only	0.38	0.53	0.55	0.43	0.58	0.73
Info Mat Only	0.33	0.50	0.57	0.24	0.49	0.69
No Raw Data	0.44	0.52	0.60	0.49	0.63	0.72
No Features	0.50	0.59	0.61	0.66	0.71	0.81
No Info Mat	0.57	0.67	0.77	0.67	0.70	0.88
No Hidden Layer	0.41	0.51	0.59	0.55	0.72	0.70
No Cross Entropy	0.64	0.74	0.79	0.70	0.90	0.91

Table 7.3: Model performance when predicting translation and rotation errors under different ablation tests. The *Unmodified* row is copied here for convenience from Tables 7.1 and 7.2. The ** *Only* rows correspond to models fine-tuned using only one of the three possible inputs (raw data, feature locations/descriptors, information matrix). The *No* ** rows correspond to models fine-tuned without one of the three inputs. The *No Hidden Layer* row represents models fine-tuned with architectures that go directly from 6144-dimensional vectors to the final 10-dimensional output, with no intermediate 2048-dimensional fully connected hidden layer. Lastly. the *No Cross Entropy* row provides metrics for models trained using loss function $\mathcal{L}_1$ rather than $\mathcal{L}_2$.

B) different fine-tuning architectures, and C) different loss functions. In these experiments, we limit the focus to just predicting ORB-SLAM2 performance using pre-trained AlexNet. The results of these experiments are summarized in Table 7.3.

Here, we can see a few trends emerging. First, raw data seems to be the most important input. We do observe substantial improvements when adding feature characterizations and some marginal improvements when including information matrix structure, but raw image data is still by far the most important input. Second, the effect of the additional hidden layer during fine-tuning is surprisingly large. This may be due to large difference in dimensionality between output layers from AlexNet (6144) and the size of the final output (10), which perhaps makes it more difficult for the network to learn and condense effective abstractions from AlexNet outputs. Finally, applying $\mathcal{L}_1$ actually slightly increases performance. This is not expected, since cross-entropy loss is occasionally *more* susceptible to overfitting, but in the absence of calibration experiments, it is not clear yet whether this result constitutes a strict improvement or a tradeoff.

7.3 Conclusion

In this chapter we showed how certain convolutional architectures are well-suited to predicting the magnitude of localization error online in SLAM systems. We also showed the utility of incorporating both memoized data as well as part of the solver's internal state of computation within the input to learning algorithms for modeling performance. While not a wholly new setup, the application of deep learning over data internal to the solver in addition to raw data represents a new approach for predicting SLAM algorithm performance. Moreover, although not conclusive, we also present some initial experiments and lay the groundwork for important future hypotheses regarding the natural synergy of various hyperparameter choices (architecture, loss) when modeling SLAM errors.

As in previous chapters, we try to exploit as much existing computation within the SLAM system as possible. Although we use several different sources of data to make predictions, including intermediate representations, all of these pieces of data are natively computed by most SLAM systems. However, while supervised learning methods are incredibly useful in many areas of robotics, there is one aspect of the supervised learning pipeline that is generally very expensive and no less so in SLAM. This is labeling. Unlike other common tasks, crowd-sourcing labels is not an option for obtaining ground truth SLAM errors. Although in the last several years reducing reliance on labeling has generally not been the focus of SLAM researchers, it is an area where innovation in unsupervised or self-supervised learning could have a profound impact, as it would allow the application of many more powerful machine learning techniques directly to many of the central subprocesses within SLAM.

There are many competing definitions of robustness or reliability in the machine learning literature, each motivated by specific use cases and deployment applications. While not necessary for all AI systems in general, for most robots, pessimistically estimating performance and including uncertainty in these estimates is critical. Pessimistic estimates help keep the robot operating in safe, predictable regions of input space and reduce the likelihood that erroneously positive predictions will lead to unsafe behavior. The focus is on maintaining system function rather than optimizing system performance while functioning. Additionally, being able to estimate the uncertainty in predictions allows the output to be used much more effectively down stream when considering other sub-tasks in the robotics stack.

CHAPTER 8

CHOOSING THE RIGHT TOOL FOR THE JOB: ONLINE DECISION MAKING OVER SLAM ALGORITHMS

While accurate location estimates are not universally required for functional robots, the overwhelming majority of mobile robotics applications rely on them. Over the last several decades, there have been hundreds of different SLAM algorithms proposed, with many designed for mutually exclusive operating conditions and sensing payloads. However, many applications for robotic systems demand operating in a wide variety of environments that afford vastly different forms of stable features. Often these sets of operating conditions cannot all be adequately covered by a single SLAM algorithm. In order to meet this need, roboticists have two options: either design a single "one-size-fits-all" SLAM system, or design methods to leverage existing SLAM algorithms in combination such that each individual algorithm can be applied when it is most relevant. Here, we argue for the latter approach, motivated chiefly by the following observations.

First, there are already hundreds of well-documented, open source SLAM algorithms. Second, these algorithms cover a wide range of sensing payloads and are designed to exploit a variety of different environmental characteristics. Together, these observations suggest that, were there a system capable of choosing the best performing system for a given situation from a pool of *already developed* SLAM systems, then the performance of these best systems would be adequate in a large number of scenarios.

Third, most of the computation in SLAM systems occurs in the back-end during the optimization process. Moreover, most state-of-the-art solvers handle essentially the same problem, regardless of whether the problem is created using data from a camera, laser, IMU, or other sensor. So, running several different front-ends in parallel is relatively cheap. Last, many SLAM algorithms rely on several different sensors with their own custom front-ends already, and having the ability to filter out a defective sensor from the data stream is already a task many SLAM researchers design for, although not in a decision theoretic way.

Thus, the problem we want to solve is to select, at each time-step, which SLAM front-end results, if any, we should pass on to the back-end optimizer. Naturally, the outcome of these actions has a degree of uncertainty with respect to the localization error. Moreover, our true state in the decision-making sense, that is, exactly how well our current algorithm will perform next time step and exactly how accurate our current location estimate is, is not fully observable. Therefore, we will model this decision-making problem using a well-known formalism for modeling such decisions: the partially observable Markov decision process (POMDP). POMDPs offer several advantages, including many approximate solutions with good performance in practice, as well as supporting reinforcement learning, although we do not propose learning the parameters of this POMDP at this time.

Nearly all state-of-the-art SLAM algorithms are designed to exploit patterns in data from specific sensing modalities, such as time-of-flight and structured light depth sensors, or RGB cameras. This specialization increases localization accuracy in domains where the given modality detects many high-quality features, but comes at the cost of decreasing performance in other, less favorable environments. For robotic systems that may experience a wide variety of sensing conditions, this difficulty in generalization presents a significant challenge. Deployed robotic systems are complex, often comprised of dozens or even hundreds of sub-systems and algorithms, each designed for a specific purpose and specific operating conditions. For example, different methods for simultaneous localization and mapping (SLAM) may be designed for different sensors (camera [242], LiDAR [255]), extrinsic sensor calibrations (front facing [139], top facing [150]), or environmental structures or affordances [258]. While such specialization often allows greater performance by leveraging structure in data to perform more efficient or accurate computation, it comes

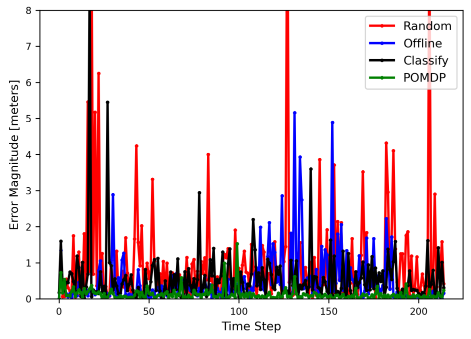


Figure 8.1: Time series of localization errors for several approaches to the SLAM ASP. The POMDP-based approach is the only one capable of reasoning about both the immediate suitability of particular sensors and the long-term effects of constructing optimization problems using these sensors.

at the cost of restricting the set of operating conditions in which the system can perform reliably. Moreover, outputs of these sub-systems are often inputs to other sub-systems and thus affect the quality of future computation in ways that are frequently too uncertain or too complex to model.

One strategy to combat these complexities is modularity. Roboticists have identified common processes (e.g. SLAM) where similar data (e.g. RGB or depth images) are processed to produce similar outputs (e.g. pose estimates). Here many algorithms may be interchanged without affecting the module's interface with other parts of the system. However, as robot deployments encounter greater variety in operational conditions, it becomes increasingly difficult to design singular, one-size-fits-all modules (algorithms) that can perform reliably under all possible conditions. Moreover, as sensors become cheaper, and sensor suites larger, it may be difficult for robots deployed in demanding or dangerous environments to incorporate all possible strategies for contingencies with different operational sensor suites within a single module. We term the development of single modules the 'one-size-fits-all' approach to system design, and our hypothesis is that this is not always the best approach to developing robust, broadly capable robotic systems.

This paper presents an alternative to the 'one-size-fits-all' architecture. Instead, we propose storing several redundant algorithms in memory, each of which may be substituted within a particular module in the stack, and then selecting the most appropriate instance of that module online. This approach essentially replaces the task of designing singular modules that operate reliably under all possible conditions with the task of efficiently identifying which existing algorithm is most reliable in the current situation. This architecture relieves the tension between specialization and generalization inherent in many robotics choices, since if a reliable algorithm exists for the current conditions and we can identify it, the system can 'generalize' to that situation without compromising performance in other situations.

Our primary contributions are (1) a formal definition of the Algorithm Selection Problem (ASP) extension this architecture presents that highlights the fundamentally sequential nature of this task in robotics, (2) a solution concept using a combination of classification and belief-space planning, and (3) a detailed simulation and set of experiments showing the potential benefit (see Figure 8.1) of online decision making over SLAM algorithms and in particular the effectiveness of belief-space planning.

The "Algorithm Selection Problem" (ASP) was first outlined by Rice in 1976 [307], and has since been studied more closely in several sub-fields of computer science. In particular, combinatorial search [175, 166] and optimization [229, 174, 241] have been among the most prolific adopters, with the SAT solver SATzilla [394] likely one of the most successful applications of ASP methods to-date. Systems that solve ASPs vary substantially in how they operate. For example, they may select a single algorithm at the beginning of the problem [394], select a schedule of multiple algorithms to run sequentially [296], select a subset of algorithms to run in parallel [147], or monitor progress and revisit these decisions during the solve [7]. Because our goal is to choose between SLAM front ends in order to maintain highly accurate location estimates *indefinitely*, we focus on the online, or dynamic, version of this problem.

Many ASP methods employ machine learning techniques to either evaluate potential algorithm performance or select algorithms directly. This application has strong ties with meta-learning [334, 167]. There have also been proposals to use sequential decision making in the form of reinforcement learning (RL) for algorithm selection when the algorithms have a recursive nature [180, 261, 259]. In some cases meta-RL systems, such as RL^3 [36], may eventually allow a form of automatic adaptation to new ASPs. However, in this chapter we focus on understanding the unique challenges of robotics ASPs, particularly for SLAM, and thus we are concerned more with formalizing the decision-making problem and developing a performant decision-making model.

Despite applications of ASP methods to other hard problems there have been few serious efforts to bring insights from ASP research into practice in robotics. The idea of using portfolios of models has been explored for localization [256], and the closely related problems of hyperparameter optimization and online hyperparameter tuning have been studied in the context of motion planning [236, 37]. However, engagement with formal ASP constructs has been, to the best of our knowledge, limited to the study of decentralized heuristic selection for coordination [310], and some computer vision tasks with relevance to robotic perception [207, 208]. In summary, this work offers the first formalisms, models, and solution methods that address the unique and fundamental challenges of applying the spirit of algorithm selection to SLAM and similar robotic perception problems. While not theoretically limited to perception systems, we see SLAM as a natural and important application of this formalism.

8.1 Algorithm Selection Problems for Robotics

Most applications of algorithm selection have been able to use essentially the original formalism from Rice with little modification. In this section we will briefly introduce this formalism, explain how robotics applications such as SLAM require more complex considerations to solve optimally due to their recursive nature and potential indefinite and reactive operation requirements, and then present an extended formalism that captures these aspects of the ASP for robotics.

8.1.1 The Algorithm Selection Problem

Formally, the ASP considers a set of algorithms $\mathcal{A}$, sometimes called a *portfolio*, and a problem instance x drawn from a some space of possible problems $\mathcal{P}$. Solving problem instance $x \in \mathcal{P}$ with algorithm $A \in \mathcal{A}$ results in a performance (often time or cost), which in our SLAM application we will call solution error $\varepsilon(A, x)$. The original ASP is thus to design or learn some selector function $S : \mathcal{P} \to \mathcal{A}$ such that $\forall x \in \mathcal{P}$, $S(x) = A^*$, where $A^* \in \text{argmin}_{A \in \mathcal{A}} \varepsilon(A, x)$.

In general, there may be other notions of maximizing performance, such as minimizing processor time or minimizing the cost of a plan for solving x generated by A. Additionally, in many cases it is not possible to guarantee $S(x) = A^* \forall x \in \mathcal{P}$, and in practice this depends on the quality of the features that can be derived from x in order to inform $S(x)$. Common extensions include selecting a set of algorithms $A \subset \mathcal{A}$ to run on x in parallel, i.e. $S(x) = \{A_1, A_4, A_{11}\}$, selecting a sequence or schedule of algorithms to run, i.e. $S(x) = \{A_1 : (t_0, t_4), A_4 : (t_4, t_{11}), A_{11} : (t_{11}, t_f)\}$, or dynamically adjusting the algorithm selected online, i.e. $S(x_t) = S(S(x_{t-1})(x_{t-1}))$.

8.1.2 Additional Robotics Considerations

There are four properties of SLAM systems that preclude some of the most common ASP approaches, as well as make the base objective a less accurate descriptor of success. First, most of the time, even after the algorithm has run, a SLAM system will not know the true value of $\varepsilon(A, x)$. This is in contrast to most other applications where, for example, the total processing time or the cost of the resultant plan is available. This makes dynamic selector functions, typically implemented as classifiers or regressors that rely specifically on fully observable features or feedback, less applicable. We could of course still apply such methods, but as we will see there are alternative frameworks that more naturally handle this partial observability constraint and do so while maintaining decision-theoretic optimality.

Second, all SLAM systems, whether Kalman filters, particle filters, or pose-graph optimizers, are *recursive*, where the state estimates produced at time t are used as inputs at time $t+1$. Thus, achieving particular intermediate data values or representations is to some degree a function of the choice of solution strategy, which is true for many complex problems but not frequently modeled explicitly in ASP applications. As shown in Figure 8.2, instances of x encountered at different points in the SLAM problem are generally not independent. Moreover, most state-of-the-art SLAM systems use sliding window pose-graph optimization [328], meaning that optimization problems become generally more stable as time progresses [200] and that switching modalities essentially reduces the active optimization window size back to just the current time step. Thus, there is a cost to the stability, and therefore accuracy, associated with switching between sensing modalities online in the context of a SLAM system.

Third, by convention, we consider so-called passive SLAM systems that can only react to changes in data quality or data quantity rather than preemptively act to affect sensing conditions. At each time step, at least some component of the data required for localization is provided in a manner beyond the control of the agent. For this reason, selecting a single algorithm at the outset is only robust if the deployment is very constrained. Moreover, robotic systems are often highly compute bound, and SLAM optimization is a notoriously computationally expensive process. While it is possible to run multiple SLAM front end feature extraction routines for various sensing modalities in parallel, it is not possible to run multiple full SLAM systems in parallel.

Fourth, indefinite simply indicates that total the size of the problem or number of steps is not known. As SLAM algorithms operate indefinitely while the robot is deployed, selecting a schedule of different algorithms to run during the deployment is not possible due to the unknown duration. We now formalize the robotics ASP, designed to capture the *recursive*, *reactive*, and *indefinite* properties specifically for SLAM and similar robotics problems. Selecting an algorithm to sort a list [180], for example, is recursive but nei-

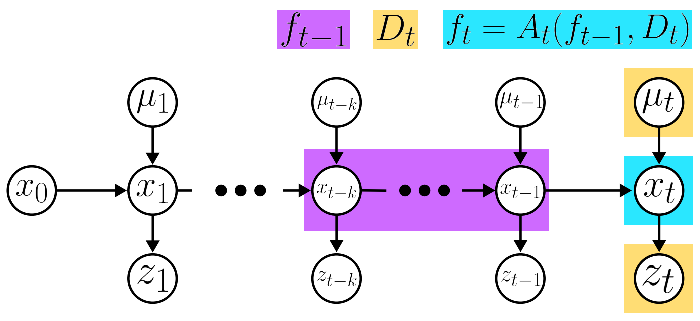


Figure 8.2: A dynamic Bayesian network (DBN) representation of SLAM inference. As new data (D_t, yellow) from proprioception (μ_t) and exteroception (z_t) is observed in an uncontrolled process, it is used alongside one or more previous location estimates (f_{t-1}, purple), corresponding to the DBN nodes $\{x_{t-k}, \ldots, x_{t-1}\}$, to estimate the current pose x_t (more generally, f_t, blue). Here, μ_t and z_t represent geometric constraints on the transformation of the robot's position over time. In practice, these geometric constraints must be derived from raw data, and it is precisely this process which may produce large errors when sensing conditions diverge from expectations. While many SLAM systems use different optimization procedures, including loop-closures and other explicit references to a persistent map or other model of the world, we note that the recursive, reactive, and indefinite characteristics of the problem remain the case in all SLAM systems.

ther reactive nor indefinite. Together, these properties create a more challenging problem requiring sequential reasoning.

We retain the notation for the portfolio of algorithms $\mathcal{A}$. Instead of a problem $x \in \mathcal{P}$, we must represent data processed in a sequence, part of which is recursive and influenced by the algorithm selected in the previous time step and part of which is generated independently by the environment and only available to the robot incrementally. We represent the latter data at time t as D_t, and the former as $f_t = A_t(f_{t-1}, D_t)$; see Figure 8.2 for more details. Both f_{t-1} and D_t affect the performance of A_t. We will denote the entire, unbounded sequence of data as $\tau = D_0, \ldots, D_\infty$, and let $\tau \in \mathcal{P}$. Of course, τ is not known at run time, but in some cases we may have domain knowledge that indicates some τ are more likely than others.

Because we care about the output quality at every intermediate time step in addition to the final time step, our objective is instead to minimize the sum $\sum_{t=t_0}^{t_\infty} \varepsilon(A_t, f_{t-1}, D_t)$. However, as the actual error is not fully observable, $\varepsilon(A_t, f_{t-1}, D_t)$ is instead a probability distribution and our objective is to minimize this sum in expectation: $\sum_{t=t_0}^{t_\infty} \mathbb{E}[\varepsilon(A_t, f_{t-1}, D_t)]$.

Finally, we see that since $f_t = S(f_{t-1}, D_t)(f_{t-1}, D_t)$, S must represent a sequence of dependent decisions with stochastic outcomes, which are naturally expressed by the notion of a policy from sequential decision making. We now give a formal definition of this problem using the notation developed above.

Definition 2. Given an unbounded data stream τ from the set of streams $\mathcal{P}$, the *Perception Algorithm Selection Problem* is to, at each time step, select using selector S an algorithm A_t from a portfolio of algorithms $\mathcal{A}$ such that the cumulative expected error $\sum_{t=t_0}^{t_\infty} \mathbb{E}[\varepsilon(A_t, f_{t-1}, D_t)]$ is minimized.

This formulation generalizes several variants. For example, if we relax the partial obervability condition on the data quality or error, we no longer need to maintain belief over the error and thus minimize the object $\sum_{t=t_0}^{t_\infty} \varepsilon(A_t, f_{t-1}, D_t)$, which we can see is solvable using an MDP. If we further relax the recursive condition, we get an objective minimizing $\sum_{t=t_0}^{t_\infty} \varepsilon(A_t, D_t)$ which is solvable via repeated classification. Last, if we relax the reactive (streaming) data condition, we minimize $\sum_{t=t_0}^{t_f} \varepsilon(A_t, D_t)$, which could then be solved using existing schedule building techniques from the ASP literature since τ is known completely in advance.

8.2 Choosing SLAM Algorithms Online

Given the key differences in robotics versions of the ASP compared to the more traditional combinatorial search applications, it is clear that dynamic ASP methods that treat repeated algorithm selection independently, and thus employ classification or regression techniques alone, will not maximize the updated objective. In this section we will first describe a POMDP decision-making model we developed to solve this problem and cover some key design choices. We will then discuss a method for generating simulations of SLAM systems across a range of sensing conditions which we use to conduct our experiments.

8.2.1 Sequential Algorithm Selection as a Partially Observable Markov Decision Process

Below, we denote members of the POMDP model with an overbar to distinguish them from other variables. We propose the following POMDP model as a promising first step towards decision-theoretic solutions to the ASP for robotics sub-systems.

- $\bar{S}$: $D_l \times D_c \times D_k \times f \times A$, where A is the previous algorithm $\{\text{LASER}, \text{CAMERA}, \text{KINECT}\}$, f is the quality (conditioning, number of correct data associations) of the previous solve $\{1, \ldots 5\}$, and D_l, D_c, and D_k, all drawn from $\{1, \ldots, 3\}$ represent the partially observable data quality of the currently streamed data. Thus there are a total of 405 states.
- $\bar{A}$: Our set of actions is the set of algorithms, $\mathcal{A}$.
- $\bar{T}$: There is no uncertainty in algorithm execution. If algorithm A is selected, algorithm A is executed. However, there is uncertainty in the quality of the next data D_{t+1} with respect to each modality and uncertainty with respect to the quality of the output f_t with respect to subsequent solve attempts. We implement a small bias towards sensing conditions remaining the same or similar since on balance this is most likely, and we also encode a small chance of increasing the quality of f_{t+1}, provided the same algorithm is used and the current optimization problem maintained.
- $\bar{R}$: Reward is the negative expected error of Equation (1), given α, β, and δ, which are functions of either D_l, D_c, or D_k, depending on the which algorithm is currently selected. We also include a mitigating factor of quality$(f_t)^{-\frac{1}{2}}$.
- $\bar{\Omega}$: $D_l \times D_c \times D_k$, where D_l, D_c, and D_k represent the quality estimates output by the classifier.
- $\bar{O}$: The observation function encodes the noise characteristics (roughly 80% accurate) of the classifiers evaluating the suitability of each SLAM front end on the current data D_t.

This POMDP is too large to solve exactly, so we use an approximate technique based on value iteration, implemented in the pomdp_py library [415], which we represent compactly as a finite state controller[46]. There are many possible extensions to this model, including using more basic information, such as the residuals from the optimizer after each step, analyzing the density of information matrix, or employing other, more complicated forms of error estimation or correction prediction [9]. Moreover, using value approximators, such as neural networks, in the context of reinforcement learning could help make this formulation tractable for higher dimensional variants, such as adding the ability to control external entities in some capacity [398].

Overall, the key idea is to exploit the fact that most SLAM back ends are modality agnostic, and thus provide substantial opportunity to intercede between raw data acquisition and factor graph construction. Such interventions may take several forms though in this case we focus on ignoring or filtering out low-quality data that may be potentially misleading or distracting. Thus, we can avoid singular, monolithic front ends that in spite of their complexity are often still susceptible to individual sensor failures.

8.2.2 Modeling SLAM Systems

While most empirical results are established for entire SLAM systems, we use the fact that many back end pose-graph solvers are modality agnostic to focus our modeling efforts on the effects of swapping out front ends only and not the entire pipeline. Empirical measures of SLAM system performance, characterized by the probability density function (PDF) of the magnitude of their location estimate errors, rarely coincide exactly with any known parametric distribution. Most commonly, such error distributions are modeled as half-normal distributions [369], or mixed distributions that include the half-normal distribution [120]. This approach is adequate although not perfect if the algorithms operate in the sensing regimes for which they were designed. However, as sensing conditions (localization affordances) change due to the passage of time or the motion of the robot in the

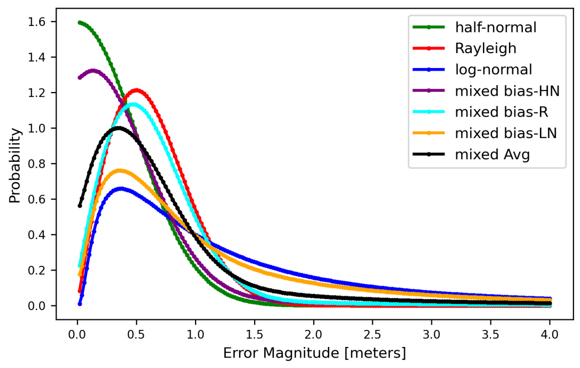


Figure 8.3: Error distributions used in the simulator. Half-normal, Rayleigh, and log-normal distributions are shown in green, red, and blue respectively, while several mixed distributions (bias-HM, bias-R, bias-LN) correspond to $\alpha = 0.8$, $\beta = 0.8$, and $\delta = 0.8$ (others set to 0.1), respectively. Mixed Avg has $\alpha = \beta = \delta = 0.\bar{3}$.

world, these error distributions also shift in accordance with how well the given SLAM algorithm can accurately estimate state given the current quality of the sensing data. Since there are virtually no models for SLAM algorithm performance in *unintended* deployment conditions, we propose a mixed distribution composed of a variable linear combination of half-normal, log-normal, and Rayleigh distributions in order to model empirically observed error distributions more realistically. This also allows more control over the shape of the error PDF depending on the simulated sensing conditions. More formally, we simulate the error distribution of SLAM algorithm A_t on data D_t as

$$\begin{aligned}\mathrm{P}(\varepsilon|A_t,D_t)=&\alpha\Big(\frac{\sqrt{2}}{\sigma_{\mathcal{N}}\sqrt{\pi}}e^{\frac{-\varepsilon^2}{2\sigma_{\mathcal{N}}^2}}\Big)+\beta\Big(\frac{\varepsilon}{\sigma_{\mathcal{R}}^2}e^{\frac{-\varepsilon^2}{2\sigma_{\mathcal{R}}^2}}\Big)\\&+\delta\Big(\frac{1}{\varepsilon\sigma_{\mathcal{L}}\sqrt{2\pi}}e^{\frac{-\ln(\varepsilon)^2}{2\sigma_{\mathcal{L}}^2}}\Big),\end{aligned} \tag{8.1}$$

where α, β, and δ are functions of A_t and D_t, and $\alpha + \beta + \delta = 1$. $\sigma_{\mathcal{N}}$, $\sigma_{\mathcal{R}}$, and $\sigma_{\mathcal{L}}$ are the standard deviation values for the half-normal, log-normal, and Rayleigh distributions, respectively. They may be tuned to provide even finer control although in our simulator and experiments we use constant values of $\sigma_{\mathcal{N}} = 0.5$, $\sigma_{\mathcal{R}} = 0.5$, and $\sigma_{\mathcal{L}} = 1$. Figure 8.3 shows several example error distributions.

In addition to sensing conditions we also consider the amount of data currently represented in the sliding window of the optimizer. Changing modalities means that recent features cannot be loop-closed against, for example because ORB [311] features from RGB images and FLIRT [362] features from lasers have no meaningful correspondence. Thus, there exists a trade off between switching front end algorithms immediately upon receiving bad data to avoid a bad location estimate, and maintaining a fully populated local map in order to be more robust to future bad inputs. Though there has been work on optimizing the size of sliding windows in SLAM solvers [200], there has been less on characterizing the effect of window size on error distributions. The established wisdom is that window size offers diminishing returns, reducing errors by a factor of roughly $1/n$ after n repeated observations [328]. Thus, after drawing an error from the distribution in (1), we apply a slightly more conservative reduction of $1/\sqrt{n}$, up to $1/\sqrt{n_{max}}$, where $n_{max} = 5$ is the maximum sliding window size.

The agent moves continuously through the world between waypoints, shown in Figure 8.4, according to some maximum linear and angular velocities. It has a bounded field of view and range for each sensor, roughly in accordance with real-world parameters. The agent may view more than one type of environment if accessible to its sensors (it cannot see through walls), and multiple environments may simultaneously affect the classifier predictions regarding the localization affordances of the current location. For example, camera-based SLAM systems operate well in cluttered, well-lit environments, but struggle in highly aliased, high contrast, or very low light settings. If the agent is viewing two areas which each have these characteristics, the incoming data D_t may be classified differently than if it was viewing an obviously poor or obviously well-suited scene. We do not include the effects of error accumulation over time or the loop-closure detection process.

8.3 Results

The primary measure of efficacy for SLAM systems, independent of a particular downstream task, is localization error. In the following experiments, we measure the effect of different modality selection strategies on the average error magnitude, the distribution of

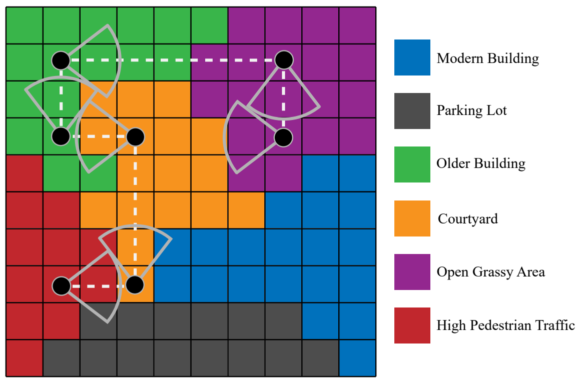


Figure 8.4: Example path consisting of 7 total waypoints an agent may take in an environment with 6 sub-environments, each of which have potentially unique localization affordances. These affordances are affected by 6 parameters: Level of ambient light, amount of clutter, level of dynamics, amount of perceptual aliasing, amount of empty space, and natural versus artificial light. Each parameter can take two possible values for a total of 64 unique possible environments. In this example, the high pedestrian traffic area may have a high level of dynamics, the courtyard may have a large amount of open space and natural light, and the older building may have low ambient light and no natural light.

errors, and the robustness of the system to sensor failures. We also show that certain strategies become relatively more effective as the space of operating environments becomes less homogeneous. In particular, we compare the following four strategies, each of which select from the same set of 3 sensors at each time-step: RANDOM, OFFLINE, CLASSIFY, and POMDP.

RANDOM selects a sensor randomly with uniform probability. OFFLINE first analyzes the entire map and estimates the suitability of each sensor for each part of the map. The sensor with the overall highest average suitability is then selected and used exclusively for the entire deployment. This method does not use information about the planned trajectory of the robot, which may visit some regions of the environment more often than others. Although this may seem like a relatively weak baseline, this is in fact essentially the current state-of-the-art approach, except that humans are the ones typically doing the pre-deployment evaluation of the environment and matching it with a SLAM front-end.

CLASSIFY runs an imperfect classifier at each time step, and selects the highest scoring modality to use that frame. This is equivalent to acting in a greedy manner exclusively on the POMDP observations, which are generated via a simulated classification process. POMDP selects an algorithm based on a policy representing an approximate solution of a POMDP modeling the problem. The primary qualitative difference between CLASSIFY and POMDP is that the POMDP represents and reasons about the effects of its current algorithm selection on the *quality of future inputs*, whereas CLASSIFY does not.

For all experiments, data was collected by randomly generating an environment, establishing several waypoints for the robot to navigate to, and then simulating and recording the localization errors. For a given number of sub-environments (2-10), all methods were run a total of 10 times over the same 10 randomly generated maps.

8.3.1 Minimizing Cumulative Error

Unsurprisingly, the POMDP method accumulates the least localization error in simulation,[1] with an average per-pose error across all 90 trails of 0.27m and a standard deviation of 0.12m. CLASSIFY, OFFLINE, and RANDOM obtained averages of 0.41 ± 0.14, 0.44 ± 0.15, and 0.93 ± 0.10 meters, respectively. The differentiating factor seems to be the approach employed during transitions in sensing conditions. While the CLASSIFY method always selects the highest scoring method regardless of history, and is thus susceptible to noise, the belief dynamics within the POMDP act as a sort of low-pass filter on the noisy sensor suitability observations, enabling the POMDP to continue using high-quality optimization problem initializations (f_{t-1}) when the lapse in reported signal quality is transient. In 81% of sequences where the robot moves from one modality to another, it took more than two consecutive observations where the current choice was ranked lower than the sensor it eventually selected.

[1]When interpreting the simulation results, we focus on the relative performance of different methods rather than their absolute performance, since the latter has little meaning in the isolation of simulation.

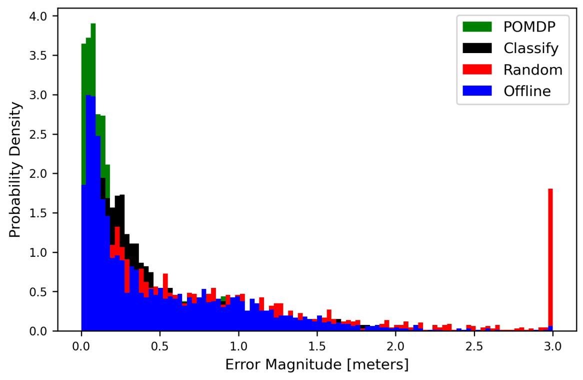


Figure 8.5: Distribution of localization errors for all trails with a total of 4 sub-environment types. Other numbers of sub-environments show a similar trend, although their means shift slightly. Note that errors exceeding 3m were capped at 3m for the purposes of visualization.

8.3.2 Avoiding Catastrophic Failures

Perhaps the most important characteristic for deployed SLAM systems is to avoid large errors. Some large errors can not only cause immediate problems in terms of trajectory following or route planning, but they can also cause difficulty when trying to re-localize or detect loop closures. Therefore, techniques that can reduce such catastrophic state estimation errors are employed frequently, and include a wide array of methods from simple thresholds to complex graph optimization. Here, we show the distribution of errors produced by each approach (Figure 8.5).

There are two key takeaways. First, the POMDP-based solution clearly enables the most robust data conditions for localization. Second, we see that the OFFLINE method has 2 modes. The first ($\mu \approx 0.1$) results from localization when it is operating in the environment for which its chosen sensor is well-suited, and the second ($\mu \approx 0.8$) occurs when it is operating in unfavorable sensing conditions. This set of events represents types of deployments roboticists may currently elect to avoid due to lack of generalizability.

8.3.3 Fault Tolerance

One key aspect of this system is that it can also deal with unexpected sensor failures. In fact, there is no need to model this event differently than, for example, entering a very dark area while previously using a camera to localize. As long as the meta-data or classifier for a given sensor can reliably detect an abnormality in sensor function, it can output an assessment of its suitability as it would for any other frame. This suitability is then used as an observation as it would be if the sensor was functioning nominally. Most powerfully, this allows systems to employ multi-modal SLAM systems and still have the opportunity to fall back on algorithms designed for a subset of modalities.

Table 8.1 shows the results of a set of experiments designed to test this capability. In these experiments, we artificially restrict the set of sensors available to the RANDOM, OFFLINE, and CLASSIFY methods, while leaving the POMDP policy unchanged. We then run simulations (10 trials with 5 different sub-environments) as before, but generate *noisy* observations that reflect the ground truth that one sensor is malfunctioning. Here, we can see that even without this prior information the policy is able to reason about the available options online and apply them at least as effectively as the other baseline methods, which were designed specifically leveraging this information.

Table 8.1: Robustness to Sensor Failure

Method	Working Sensors	Mean Error	Standard Deviation
RANDOM	2/3	0.87	0.13
OFFLINE	2/3	0.43	0.09
CLASSIFY	2/3	0.39	0.14
POMDP	2/3	0.32	0.11

8.3.4 Effect of Environment Heterogeneity

While the OFFLINE method works well in cases where we can predict pre-deployment how likely different sensing conditions will be during deployment, we can see that the more heterogeneous or unpredictable the operating environment becomes, the more challenging it is for non-reactive systems to perform well. Figure 8.6 shows a graph of average localization error across all trials as a function of the number of sub-environments present in the map (number of distinct regions with different sensing characteristics, or colored regions in Figure 8.4). Large values on the x-axis indicate a less homogeneous operational environment.

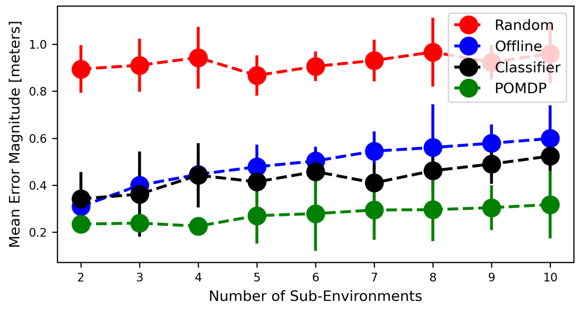


Figure 8.6: Average localization error for all trials as a function the number of sub-environments in the map. Vertical bars represent one standard deviation.

Because OFFLINE selects one sensor, a uniform traversal of the map provides a lower bound on the rate of optimal sensor selection of just $|\mathcal{A}|^{-1}$. Moreover, if we consider arbitrary multimodal systems, this becomes roughly $2^{-|\mathcal{A}|}$. Therefore, we expect the performance of OFFLINE to decrease initially before converging as environment diversity increases.

8.4 Conclusion

This chapter outlined a new type of algorithm selection problem for robotic perception, devised a solution to this problem using partially observable Markov decision processes, and detailed several experiments showing the effectiveness of online decision making and sequential reasoning for such problems. We also presented results from a detailed simulation of SLAM algorithms based on multiple modalities which show that SLAM systems with the ability to modulate reliance on front-end data resources experience less localization error on average as well as significantly fewer catastrophic localization errors.

Robot architectures are full of feedback loops, which makes planning about future computation complicated. For example, sensor data affects state estimate, which affects route planning, which then affects the stimuli available for sensing. In addition, new data is constantly being incorporated into partially solved problems. These two complications have been largely ignored by the AI community, presumably due to more immediate concerns about the function of individual nodes in the architecture. Considered together, they make obvious the importance of sequential decision making under uncertainty when trying to maintain operating conditions within nominal regimes across all nodes in the architecture.

In addition to all robots operating in partially observable physical environments, it is also the norm for individual algorithms to operate in partially observable computational environments. Typically, each node of computation within the stack has a very limited (partial) view of the state of computation in the rest of the system, which can in some situations dramatically affect the quality of the overall behavior. In some sense, keeping a robot operating normally is itself a control problem due to the robot's ability to affect the data it receives and saves for future computation. While many roboticists have recognized the need to address the partially observable nature of operating in the real world and the suitability of POMDPs for modeling such problems, relatively less attention has been paid to the (also partially observable) meta problem of maintaining computational state (in this chapter we focus on localization) within the robotics stack.

CHAPTER 9
AN INTEGRATED ARCHITECTURE FOR MULTI-SLAM SYSTEMS

As has been demonstrated throughout the previous chapters, SLAM systems are complex, with many components that must operate together in order to get accurate location estimates. Moreover, these components all have different ways of processing raw or intermediate information and parameters that must be adjusted. While this is already a challenging aspect of most SLAM systems, it is even more so for Multi-SLAM systems. While incorporating additional SLAM algorithms increases planning complexity and the demand for training predictive performance models, the core topics of this chapter are not fundamentally dependent on the number of SLAM algorithms within the Multi-SLAM portfolio.

Planners that reason about on-line SLAM algorithm execution and the predictive models of SLAM algorithm performance that inform them are the major building blocks of multi-SLAM systems, however, there are many other auxiliary problems that must be solved before these systems may be deployed. Such problems largely fall into two categories: 1) issues that must be solved in order for the system to run at all, and 2) issues that must be addressed for the system to reach something near its maximum potential. This chapter will focus on topics within both categories.

First, several issues arise when using asynchronous exteroceptive sensing. This is not a completely novel problem, but the vast majority of SLAM systems use one or more very high frequency proprioceptive sensors and just one, lower frequency exteroceptive sensor. The problem arises when choosing discrete points in time at which to represent the robot's pose in the problem. Most of the time, this problem is hidden within a subscript. $\mathbf{x}_1$ is not "the robot's pose at time $t = 1.000$s"; it is instead the first point in time we would like to solve for the robot's pose exactly. This may be any time with respect to a wall clock. Thus, the act of creating variables (poses) $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, whether within a Kalman filter, particle filter, or pose-graph, is a form of discretization. In most systems, there is an obvious choice for how this should be done, which is to create poses for every exteroceptive sensor reading, or some subset of them defined by simple rules like minimum estimated displacement or minimum elapsed time. However, these choices are not at all straightforward when the robot is receiving information from different exteroceptive sources that are similarly frequent, but not synchronized, either together or in some constant, known phase.

As with typical systems, we can still represent this problem as a dynamic Bayesian network (DBN). However, this DBN becomes significantly more complicated for two reasons. First, simply adding a second set of exteroceptive readings at arbitrary times and with an independent generative model (map) adds some complexity. Second, although it is easy to just add additional nodes (represented as variables in subsequent solves) to the DBN, in practice we do not want to solve this problem exactly since, for the same time period, the optimization problem it creates is roughly twice as expensive. Thus, we will need a method for reasoning about which sub-problem from within the DBN we should represent at a given time, with the primary challenges for this decision coming at points when the system decides to switch SLAM front-ends from one modality to another.

The second issue we will tackle is the following: It seems plausible that even if features extracted from raw sensor data are not used immediately, for example, because they were from a laser and the camera-based system was chosen at the time, these features may have value in mapping and loop-closure processes, either later in the same deployment or in subsequent deployments. Moreover, the addition of reliable features within the map, even if not originally used for their own registration, may alter the relative reliability of different SLAM systems that encounter the same location again in the future, possibly under different sensing conditions. For example, if the robot is operating outside in relatively low light and thus uses its laser to localize, it may still observe salient features with its camera. Upon revisiting that location in daylight, having access to these features within a map such that it may find correspondences within the current frame will likely reduce localization error.

This chapter introduces several data structures and algorithms already used within SLAM systems and extends them for use in Multi-SLAM systems. Specifically, we will focus on trajectory estimation and representation using asynchronous sensors, and multi-modal map representations. The realities of asynchronous sensing are one of the most commonly overlooked or assumed-away challenges in SLAM. However, in highly dynamic applications such as driving or flying, these issues are unavoidable. There are several related but distinct phenomena variously referred to as synchronization problems in robotics. Here, we are interested in asynchronous exteroceptive signals, which have roughly the same rate, but are not time synchronized, even if they are time calibrated [354]. Thus the times at which each sensor captures the state of the outside world may not be the same, regardless of whether the time at which the signals are received by the computer estimating state. We are also considering only single-agent problems, and the asynchronous label does not indicate problems or constraints on inter-agent communication [219]. Finally, we are not necessarily addressing problems of synchronizing maps, as may be the case when sensing configuration or robot operations require turn-taking or non-overlapping signals [61].

We are also restricting our discussion to a certain subset of sensor types and SLAM system designs. Problems with asynchronicity may be addressed through the use of fundamentally different types of sensors that provide more or less streaming data, such as event cameras [267], or by changing the nature of the SLAM optimization problem (find the MLE trajectory) from a non-parametric problem to a parametric one, where time is a continuous-valued parameter. So-called continuous-time SLAM systems represent trajectories as parametric equations, where the optimization process finds parameters rather than locations at fixed times [71, 396]. For example, they may represent the robot's trajectory as a polynomial or other type of spline. This does elegantly solve many synchronization problems but also requires selecting the proper parametric form, which may or may not be harder than solving synchronization problems in the discrete formulation.

With respect to discrete-time SLAM systems, the most common synchronization approaches use the high rate of most IMU signals a type of quasi-continuous signal that can be aligned or integrated to reduce errors from out-of-sync sensors [149]. This concept can be extended with the addition of buffers for signals from one sensor waiting to be fused with other data [184], or to cases where the asynchronization takes one of a finite number of known forms [341]. Other approaches mitigate the effect of late, delayed, or low-rate data by changing optimization window size [165].

Multi-SLAM systems have an additional requirement in that we want to be able to construct and initialize optimization problems in a flexible manner that may use or represent different, potentially disjoint, subsets of information. For example, if the POMDP decides to switch front-ends, then we need to construct and initialize a new optimization problem with factors related to the chosen set of modalities. There have been proposals to essentially interpolate between relative transform factors of different modalities in SE(3) in order to define single points in time to instantiate pose estimates [110], but it is unclear if this approach can handle switching between problems or retroactive solving, which are the primary complications in Multi-SLAM.

As outlined in chapter 2, there are many different representations of a robot's operating environment, and these representations are heavily contingent on both the robot's expected tasks as well as its sensor suite. Since, Multi-SLAM systems are designed to be task agnostic, we are mostly interested in how to support storage, access, and manipulation of map information across many different modalities and intermediate representations. Here, we will restrict our attention to metric maps, or maps that at least contain some metric information, although many points also apply to topological maps [253]. In general metric maps may be represented in a number of different ways, including landmarks and keypoints [403], dense raw data such as RGB-D images [127], dense parametric representations like line segments and planar facets [248, 258], object-level entities or other semantic labels to groups of data [224], or latent within a neural network [25]. Neural network maps have become popular in the computer vision community recently, but they generally not practical since they require a large amount of labeled data from the area that needs to be mapped before being able to localize. That is, they cannot realistically be applied to SLAM problems, only to localization problems after the fact. For more on different map representations and their pros and cons, see the excellent survey from Cadena et al. [50].

At a high level, maps facilitate two (sometimes more) behaviors: (1) localization, and (2) navigation. Depending on the environment, available sensors, and task, the data structures that support localization algorithms may be different than those that support navigation, both route planning and trajectory planning. For example, a robot that localizes by matching keypoints from an RGB camera cannot navigate safely using those keypoints alone since there is no guarantee that all physical obstacles in its environment will be represented as keypoints. Maps that facilitate additional high-level planning beyond localization and navigation are often quite complex in order to be effective [295].

There have also been proposals to build general libraries that try to cover a range of possible representations [368], although it is unclear how often these types of generic packages are adopted in practice. Recently, there has been some more targeted work aimed specifically at map reuse and flexibility within a small subset of sensor choices. ReSLAM [308] is a method for map storage that uses a multi-resolution reconstruction to support different possible cameras. We propose a family of representations and map building algorithms that target a middle ground between these approaches using clustering to adaptively and conservatively produce useful, stable, and informative artifacts from multiple modalities. Theoretically, these representations can support large number of different sensor-task combinations without exhaustively populating irrelevant data structures.

9.1 Multi-SLAM Systems

The two problems outlined above have solutions that are largely independent, and we will cover them in the following two subsections.

9.1.1 Dynamic Bayesian Networks for Multi-SLAM Systems

Dealing with asynchronous data can be modeled as a dynamic Bayesian network (DBN) like that in Figure 9.1. This is not a wholly new model, but does represent a level of complexity not typically expressed in SLAM problems. Systems that do use multiple exteroceptive sensors often fuse them into a single time stamp before adding them to the DBN, though in our case we would like the ability to completely and independently switch between sensor streams. The crucial problem surfaces when a switch between localization methods is suggested. At this point, a new optimization problem must be built using a different subset of the DBN. Moreover, this new problem will likely not share any variables with the problem solved at the last time step. Without modifying the problem setting by maintaining some factors for continuity, this is likely to cause instability or poor localization estimates. However, this problem may be mitigated by initializing the problem with mixed sets of variables representing multiple modalities as a way of "stitching" the two independent estimation problems together in a smooth way. Moreover, we want to avoid adding or keeping constraints to the problem that are possibly erroneous or of otherwise low quality.

For example, consider the factor graph in Figure 9.2. Here, factors labeled u are from an IMU, odometer, or other high-frequency, proprioceptive sensor, and factors z^1 and z^2 are from exteroceptive sensors — in this case a camera and LiDAR, respectively. In general, there may be arbitrarily many sensors, each returning measurements $z^1, \ldots, z^k$. Subscripts simply indicate the temporal order in which signals were received. As the robot operates, it receives signals at discrete points in time from all sensors. The goal of whichever SLAM front-end is selected is to calculate accurate factors given the most recent perception data

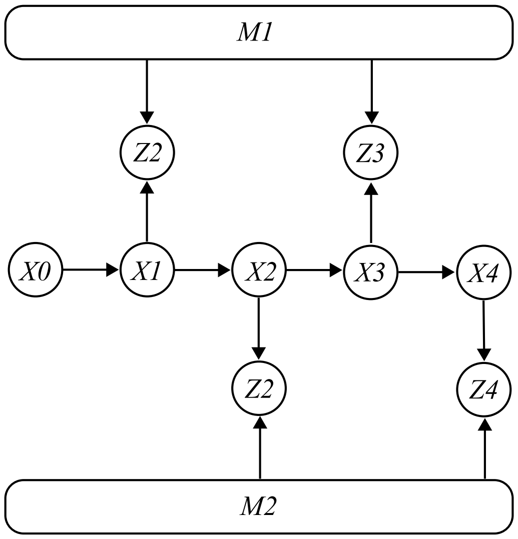


Figure 9.1: A dynamic Bayesian network modeling the location estimate of a robot using two asynchronous exteroceptive sensors.

and then add them to the graph. Factors u only exist between subsequent poses because they arise from sensing the robot's own motion and cannot be related to more distant poses except through constraining intermediate poses. Thus, conditioned on the previous pose, the current pose is independent of all but the most recent ego-measurement u. Factors z, since they come from measuring the outside world, may theoretically relate any two poses.

The process of adding factors at specific points in time that coincide with particular measurements essentially discretizes the trajectory estimation problem. It is possible to add new pose estimates whenever *any* measurement is recorded, but this often results in graphs (and thus optimization problems) that are quickly far too large to solve online. Thus, a common strategy to reduce the number of variables in the optimization problem, and one that we adopt here, is to pre-integrate inertial measurements before creating pose variables [210, 106]. We do this by starting with the equations for the incremental change in orientation R, velocity v, and position p recorded between each inertial measurement:

$$\begin{aligned}
R(t+\Delta t) &= R(t)\text{Exp}(\omega(t)\Delta t) \\
v(t+\Delta t) &= v(t) + a(t)\Delta t \\
p(t+\Delta t) &= p(t) + v(t)\Delta t + \frac{1}{2}a(t)\Delta t^2.
\end{aligned} \tag{9.1}$$

Given the update rule for each timestep, we can combine them to form a single measurement over many timesteps as

$$\begin{aligned}
R(t+1) &= R(t)\prod_{k=0}^{\Delta^{-1}-1}\text{Exp}(\omega(t+k\Delta t)\Delta t) \\
v(t+1) &= v(t) + \sum_{k=0}^{\Delta^{-1}-1} R(t+k\Delta t)a(t+k\Delta t)\Delta t \\
p(t+1) &= p(t) + \sum_{k=0}^{\Delta^{-1}-1} v(t+k\Delta t)\Delta t + \frac{1}{2}\sum_{k=0}^{\Delta^{-1}-1} R(t+k\Delta t)a(t+k\Delta t)\Delta t^2,
\end{aligned} \tag{9.2}$$

where a is the linear acceleration, ω is the angular velocity, Exp() is the exponential map. For simplicity here, we have omitted explicit notation representing the IMU biases, gravity terms, and noise terms, which are occasionally helpful in making explicit models, but in the case of the noise are not directly observable and in the case of the gravity and bias terms, may be rolled into the final measurement prior to pre-integration.

Given a pre-integrated inertial measurement u_t, we can derive a cost function, represented by a factor in the factor graph, that represents the negative log-likelihood of the expression $\text{P}(x_t|x_{t-1}, u_t)$. Similarly, factors can be derived for exteroceptive measurements by minimizing reprojection errors of salient features observed in different frames, leading to the expression $\text{P}(x_t|x_{t-1}, u_t, z_t)$. If we consider a fixed window of past measurements of size n, we get $\text{P}(x_{t-n:t}|x_{t-n-1}, u_{t-n:t}, z_{t-n:t})$. The naive extension of this expression to include multiple, asynchronous sensors is $\text{P}(x_{t-n:t}|x_{t-n-1}, u_{t-n:t}, z^1_{t-n:t}, \ldots, z^m_{t-n:t})$, which produces the objective function

$$
\begin{aligned}
X^*_{t-n:t} = \operatorname{argmin} & \sum_{i=t-n}^{t} ||(x_{i-1} - x_i) - u_i||^2 \\
& + \sum_{z_{ij} \in C_1} ||(x_i - x_j) - z_{ij}||^2 \\
& + \dots \\
& + \sum_{z_{ij} \in C_m} ||(x_i - x_j) - z_{ij}||^2.
\end{aligned}
\tag{9.3}
$$

While potentially expensive to evaluate, and potentially containing inaccurate or erroneous constraints, this cost function is fairly straightforward to construct. If we identify some particular modality z^l as being low quality, we can simply omit the corresponding factors from the objective.

Occasionally however, it may be advantageous to actually delay including new factors in the graph in order to either determine their quality more reliably by gathering more data or minimize the risk of getting low quality solutions due to having small, poorly conditioned problems. Moreover, if a maximally accurate pose estimate is not immediately required, the robot may be able to save computation by foregoing a full solve and relying on methods like dead reckoning for short periods of time. Sometimes it may also be better to delay inference in order to see what kind of signal will be available from the next asynchronous reading before committing to solving a problem using the current modality. In all of these cases, this creates a higher level decision problem of which data to use and which to either delay using or discard altogether. Here, we do not provide a solution to this problem, but we do show how the desired factor graphs can be constructed straightforwardly.

For example, consider Figure 9.2. If the robot began operating using front end 1, and at time t switched to front end 2, there are several plausible factor graphs that could represent the new inference problem, depending on exactly what information was deemed reliable. The first option is to make a hard switch (Figure 9.2(a)), immediately removing all data from z^1 occurring before t from the problem, and leading to the probability expression $\mathrm{P}(x_{t+\phi} | x_{t-1}, u_{t-\phi:t+\phi}, z^2_{t-\phi}, z^2_{t+\phi})$, where ϕ represents a variable offset in time between when signals from z^1 and z^2 are captured. Note, ϕ is not necessarily a constant throughout operation.

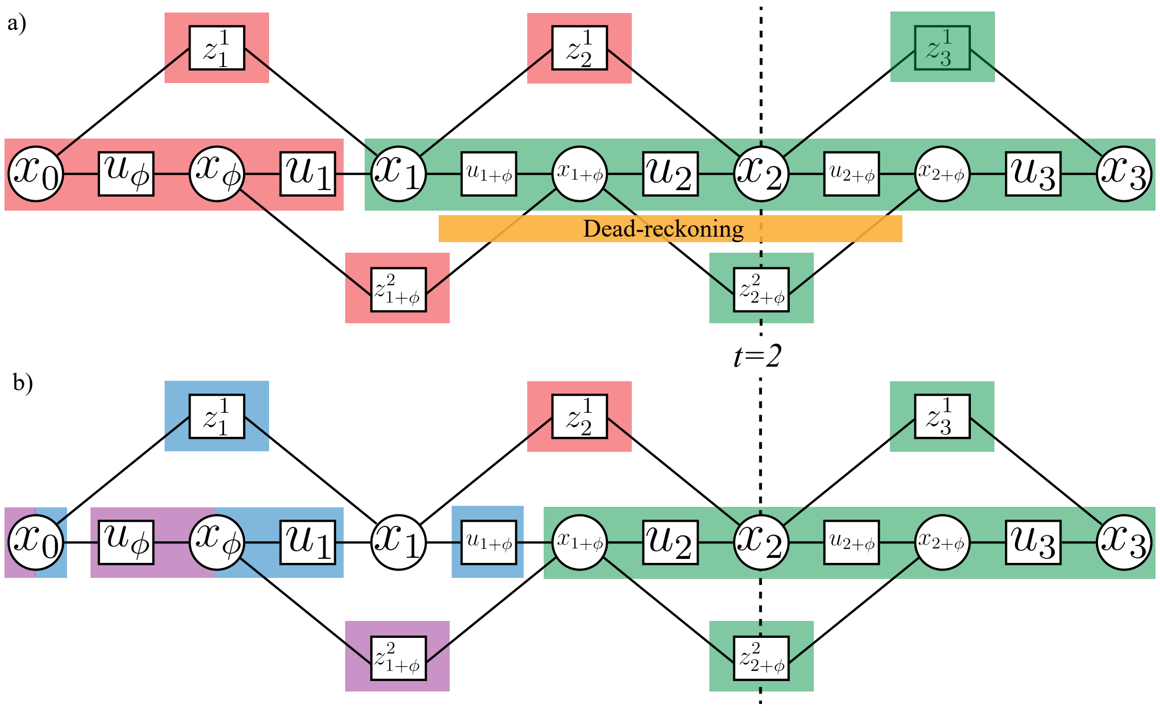


Figure 9.2: Full multi-SLAM factor graphs. Circles represent potential variables in the problem. Squares represent constraints between variables based on either matching between measurements (z) or integration of high-frequency measurements (u).

The second option is to try to stitch the two problems together by continuing to include factors from the previous front-end. Here, we want to avoid instantiating poses for both the set of timesteps when z^1 captures data and the set of timestamps when z^2 captures data since this will make our problem more expensive to solve. To do this, we can use the fact that for any given timestep we have multiple *independent* paths through the factor graph which lead to the same pose variable, regardless of which front end is synchronized with that particular pose. Thus, we can construct *non-redundant* factors that combine previous proprioceptive and exteroceptive factors in a manner that summarizes the existing information in the factor graph without double counting. For example, in Figure 9.2(b), in addition to the new set of variables (green), we can instantiate two different constraints (blue and purple) composed of completely separate data that both link the initial pose x_0 to subsequent poses discretized at times $x_{k+\phi}$ which align with sensor 2, and thus avoid a prolonged period of dead-reckoning.

Last, the case of delaying the creation of poses is also straightforward. Dead-reckoning alone requires no new optimization since we are using only one source of information. Thus, it is possible to integrate IMU or odometry indefinitely. If, after some delay, it is determined that past sensor data should be used, and thus new pose variables and factors are required, we can simply partition the proprioceptive data at the corresponding points and construct a factor graph that is identical to the one that would have been constructed without delay.

These variations are in fact all special cases of a more general, meta-problem of deciding what information to use when estimating a trajectory. In the past, significant attention has been paid to so-called 'robust SLAM' methods that detect and ignore, through various mechanisms [343, 181, 141] potential bad loop closures. That is, they reduce the impact of, or remove altogether, certain factors z^i if it is determined later on that they are erroneous or inaccurate enough to degrade the quality of the trajectory estimate. This problem is also related to work on determining the size of sliding windows during sliding window optimization [200]. The proposed graphical structure is also highly reminiscent of multi-robot SLAM systems. Given the potential need to manage a heterogeneous fleet of robots with different sensors and SLAM algorithms, the proposed model may provide a reasonable starting point. Given the complexities of these problems and the likely challenges of analytical models, this may be a good candidate for machine learning algorithms in the future.

9.1.2 Maps for Multi-SLAM: Storage, Access, and Manipulation

We propose a combination of approaches for map storage and representation that trades memory and computation, some of it offline, for a more flexible and hi-fidelity model of the environment. First, we maintain independent maps for each sensor type that are registered using trajectories copied from the Multi-SLAM system output, so there is no feedback between different trajectories updating, creating updated maps, and then creating other updated trajectories. This, of course, limits some of the ability of this representation represent alternative features, but it is also substantially less complicated. Creating and maintaining independent maps also allows these maps to be used by other robots that may not possess the same exact sensor suite, since they can access feature types independently.

However, registering these features accurately is not straightforward due to their non-alignment in time with respect to other sensors. We solve this problem by simply using the high-frequency proprioceptive data to interpolate between the pose estimates of the selected front-end at any given time. At a high level, this strategy leaves a lot of room for flexibility since the underlying data may be anything from raw images to ORB features to semantic objects like vehicles or doorways. During deployments, there will also be many instance where observations of the world reveal some useful information that determines, for example, the existence of a reliable low-level feature, but are inconclusive with respect to a higher-level question, such as the existence of a particular semantic class or the co-occurrence of features of multiple types.

One possible solution is to spend time offline to cluster data in a hierarchical, agglomerative manner, where lower level representations are iteratively combined into higher-level entities [250]. For example, as shown in Figure 9.3, individual laser returns can form line segments, line segments can form objects, and different collections of object classes can form different types of places. The benefits are 3-fold. First, if run offline, the algorithm has the advantage of a complete global perspective on all modalities of data and relatively unlimited compute time, both of which are not available during runtime. Second, clustering naturally deals well with partial observability and low-quality data since it has the option to simply omit the current data from being used in the next level of analysis. For example, if it is not clear whether a particular line segment can be conclusively attributed to a particular object, it can remain in its current representation. Last, it allows the use of other models for localization that require a latent model of the environment and then attempts to match currently observable features. Such models may be very expensive to compute online.

Hierarchical agglomerative clustering (HAC) essentially takes all the products of each front end and iteratively combines them into higher level representations. Here, we use a technique from so-called ‘infallible’ classification, where objects may remain partially clustered (classified) if there is not enough data to continue combining them. In our application, we use this technique to form multi-modal features and high-confidence regions within the map by spatially clustering features that were originally independently detected by RGB data or LiDAR data. Such features and regions can then be used later to perform more robust feature selection and matching during subsequent localization events, even if the modality used to localize is different than the modality used to originally register the features during map construction.

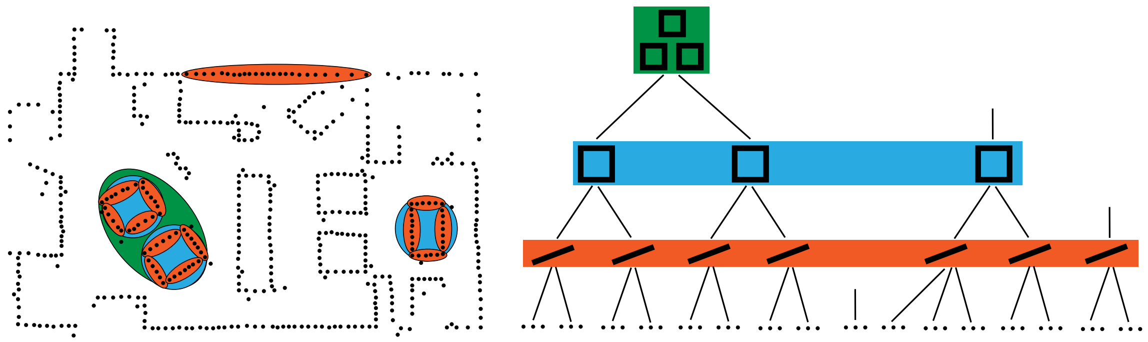


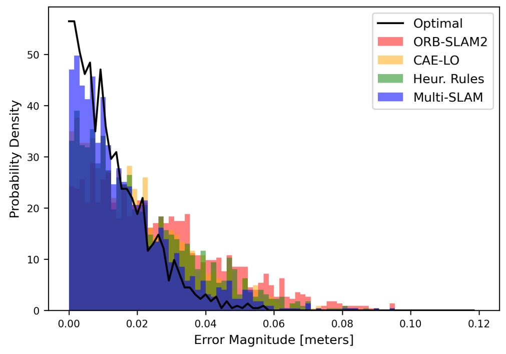
Figure 9.3: Right: an example of partially complete clustering of depth data. Black dots are individual laser observations; orange ellipses represent line segments identified via clustering; blue ellipses represent closed objects; green ellipses represent groups of objects. Left: an example dendrogram. Note that many objects do not connect all the way to the top layer, since the map lacks the data to conclusively determine their membership in a larger group. However, they can persist in the map and be clustered later should that data become available.

9.2 Results

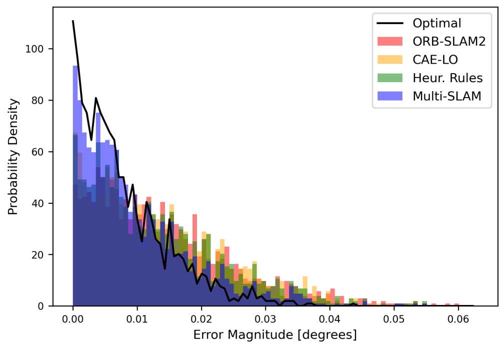
The goals of the following experiments are relatively simple. We would like to test if multi-SLAM systems allow a greater range of operating environments than SLAM systems that rely on a single front-end, or systems that use rules, rather than learning, to select SLAM front-ends. Here, we will look at the distribution of localization errors with respect to ground truth as the primary measure of success. However, there are other measures, such as computation, that could be additionally considered in future studies. As before we will use the KITTI data set, since we require ground truth and multiple modalities simultaneously, whereas in the previous chapter, training could be split by modality across datasets. This is because the robot needs to have the option to use or ignore data from any modality at any point, and we need to be able to measure its performance against some ground truth value, regardless of its choice.

In addition to accuracy, we also test the robustness of the multi-SLAM system with respect to sensor degradation. We do this by artificially altering the raw data online, before it is sent to the SLAM front end. As before, we use the ORB-SLAM2 and CAE-LO algorithms, in addition to allowing the multi-SLAM system to also choose to use neither and instead rely only on inertial sensors, or so-called dead reckoning. We also implement a method for selecting front ends based on heuristics rather than learning or planning, which we call Multi-SLAM HEURISTIC and use as a baseline. This model simply looks at the overall intensity (or depth, for LiDAR), and the number of features. If either quantity is too high or too low for a given modality, it prevents that data from being used. If all modalities fail, it selects dead-reckoning. A more performant overall system could in theory be constructed using state-of-the-art SLAM solutions for each modality [74, 412, 70].

Our last set of experiments investigate the potential quality of maps constructed from one modality which are registered using localization estimates from an independent modality. One experiment simply tests the reuse of independent maps constructed using multi-



(a) Translation Error



(b) Rotation Error

Figure 9.4: Distributions of localization error. It is encouraging, although perhaps not surprising, that simply choosing between several sub-optimal solutions can in fact improve performance significantly. Note the optimal curve is computed by taking the minimum error between all systems available (ORB-SLAM2 and CAE-LO) at each time step. The optimal curve of course does not represent a zero-error solution.

SLAM, and the second uses infallible-HAC to post-process multi-modal maps and tests their localization performance.

9.2.1 Multi-SLAM Localization Accuracy

Here, we compare the accuracy performance of multi-SLAM on KITTI data against the constituent SLAM algorithms and Multi-SLAM HEURISTIC. Figure 9.4 shows the distribution of translation and rotation errors for each approach. The histograms are aggregated across all 11 trajectories with ground truth from the KITTI data set. Here, we see that Multi-SLAM performs significantly better than either individual algorithm, and also significantly outperforms the rule-based heuristic selector.

However, we should note several important caveats. First, these experiments were performed using SLAM algorithms that are already relatively performant, due to their open-source implementations. It is an open question if benefits of the same magnitude extend to systems with much different performance profiles. Moreover, the data set used, although it contains some challenging aspects, is meant to be more or less ‘in distribution’ for most SLAM systems, meaning that it does not contain many of the most challenging sensing edge cases. Furthermore, the sensors used to collect this data are exceptionally high quality, and we ran these experiments without significant compute constraints or real-time constraints. Thus, there is still significant empirical work to be done to validate this result on a broader spectrum of potential applications.

Last, we must highlight that the ‘optimal’ curve shown in the figures is not a theoretical limit, even given the underlying algorithms Multi-SLAM is selecting from. This is because Multi-SLAM may solve distinct optimization problems that are not a subset of the problems that ORB-SLAM2 or CAE-LO would solve. Thus, further improvements to the problem of which features to include could theoretically create optimization problems whose solutions result in even lower levels of error than even the best possible solutions from systems that use all features.

9.2.2 Robustness to Sensor Degradation

To test robustness to sensor degradation, we apply several different treatments to raw camera and LiDAR data. For camera data, we apply Gaussian blur, simulated occlusion, intensity shifting, and salt and pepper noise. For Gaussian blur, we convolve a 5×5 Gaussian kernel withthe raw image. To simulate occlusion, we randomly sample a contiguous patch of the image that contains between 25% and 50% of the image and set all pixel values to the mean intensity of the image. To shift intensity, we randomly choose a number between 40 and 80 and add or subtract that number from all pixel values, capped at 0 and 255. Last, for salt and pepper noise, we randomly select 10% of pixels and set the to either white (max value) or black (min value) with equal probability.

For LiDAR data, we also apply blur, occlusion, and noise. We do not implement an analog for intensity shifting. Blur is applied in the same fashion with a Guassian kernel of size 3×3 due to the lower resolution of the depth image, although for rays that return nothing, we omit them from the calculation and re-normalize the remaining kernel weights. We model occlusion by also through a similar process, but instead set values to represent no

Treatment	Multi-SLAM	ORB-SLAM2	CAE-LO
Blur (Camera)	0.018	0.112	-
Occlusion (Camera)	0.021	0.033	-
S&P Noise (Camera)	0.022	0.043	-
Intensity (Camera)	0.018	0.076	-
Blur (LiDAR)	0.023	-	0.099
Occlusion (LiDAR)	0.022	-	0.054
S&P Noise (LiDAR)	0.013	-	0.020
No Treatment	0.012	0.023	0.018

Table 9.1: Mean translation error in meters for different SLAM systems under different adverse input treatments.

	Multi-SLAM	ORB-SLAM2	CAE-LO
Blur (Camera)	0.0051	0.1906	-
Occlusion (Camera)	0.0096	0.0683	-
S&P Noise (Camera)	0.0051	0.0091	-
Intensity (Camera)	0.0070	0.0252	-
Blur (LiDAR)	0.0055	-	0.1295
Occlusion (LiDAR)	0.0054	-	0.0778
S&P Noise (LiDAR)	0.0044	-	0.0063
No Effect	0.0041	0.0055	0.0051

Table 9.2: Mean rotation error in degrees for different SLAM systems under different adverse input treatments.

return rather than a specific depth. Last, to apply salt and pepper noise, we again randomly sample 10% of rays and replace the given value with either the no return value or the minimum range value, with equal probability.

Tables 9.1 and 9.2 show the effect of different signal perturbations on SLAM algorithm performance. Overall, the performance seems to be much more robust to individual signal corruption, which is to be expected since this is what we designed for. In some cases, such as signal blurring, it seems that the system essentially used only the non-affected sensor input for the entire duration. It is also clear that the performance predictions are not uniformly good over all treatments. For example, even when the camera suffered substantial occlusion, ORB-SLAM2 was still selected enough of the time to significantly degrade performance below what would have been possible using CAE-LO alone. This is perhaps due to the greater feature density from the camera images making up for a relatively limited field of view. Regardless, more work is necessary to understand this particular case.

9.2.3 Repeated Localization

In this experiment, we want to test two related hypotheses. First, that maps populated with features from one front-end but registered using the pose estimates from multi-SLAM are at least as reliable for localization as maps populated by the same features used to register poses. And second, maps containing only features identified through clustering are more reliable for localization than maps containing all features.

To test the first hypothesis, for each trajectory we record which SLAM front-end (ORB2 or CAE-LO) was invoked the least. After making a map of feature registered using Multi-SLAM, we play back the trajectory again and this time force localization to be done with the lesser-used SLAM algorithm. Overall, across all 11 trajectories, we find that localization accuracy contains roughly 1.17 times as much error with respect to translation estimates and 1.52 times as much error with respect to orientation estimates. We find this disparity larger than anticipated, although rotation is notoriously more difficult to estimate accurately. One possible reason for this is that some features that are ignored during the operation of Multi-SLAM should never have been detected in the first place. For example, if ORB features are not used in a given frame, it is not known whether they are only marginally less reliable or are completely erroneous. In the latter case, relying on them in the future could create significant error.

To test the second hypothesis, we collect all features from all 11 trajectories using Multi-SLAM and register all features to the same global map. We then run clustering using simple Euclidean distance threshold for determining cluster membership. There are 9 total cluster types, based on size: singletons, groups of 2-4, and groups of 5 or more, and present modes: RGB only, LiDAR only, and both. Finally, given these clusters and each feature's membership within some cluster, we run ORB-SLAM2 again over the 11 trajectories. However, when localizing, we weight the cost of each feature by looking up the type of cluster it belongs to, where larger clusters that include both depth and RGB data are given higher weight compared to smaller clusters of those that include only RGB or only depth. This method results in a 4.1% reduction in average translation error and a 6.5% reduction in rotation error, which although not large, is promising for the future utility of such clustering methods in map reuse and flexibility.

9.3 Conclusion

In this chapter we presented two different problems of data organization and relation, one related to constructing the proper inference problem and one related to constructing the best model of the world, and presented some possible solutions. We then tested those solutions and found that although it is difficult to prove from first principles that they are optimal in some sense, they nevertheless allow us to use multi-SLAM systems effectively and flexibly. We also discussed the implications of some of our design choices on the overall life cycle of robots that implement multi-SLAM systems. Specifically, we investigate how such systems may allow a greater spectrum of tradeoffs between development and deployment costs, and how practitioners may navigate them.

One difference between this chapter and most previous chapters is that the problems discussed here are largely the result of design decisions rather than a priori questions about accurate inference or optimal behavior, much like other problems that exist only because of the way a system is architected [246]. While easy to express, it is often much more challenging to convincingly and definitively answer and these questions or evaluate their solutions. This difficulty might be due to the relevance of both quantitative attributes like efficiency or memory footprint and qualitative attributes like flexibility, generality, or intuitiveness, some of which are functions of the mathematical formulation of the data structures and algorithms and some of which are tied to the role a given system plays within the given robotics architecture.

When we look at the extent to which such tradeoffs exist, we can see there is an important crossover point in robotics where design decisions have mathematical consequences in terms of the problems they create. Moreover, when design or architecture decisions are viewed systematically with respect to their effects on the entire system we may be able to apply mathematical models and analyses, in addition to qualitative evaluations, to help make design decisions. Multi-SLAM systems represent a compelling case of algorithmic development potentially driving research on robotic architecture and subsequent analysis of the science of integration — understanding how how, why, and to what effect systems may be composed of other systems, and introducing systematic, mathematical analyses of decisions that are currently challenging to predict and evaluate.

CHAPTER 10

CONCLUSION

10.1 Summary of Contributions

This thesis has touched on several problems relating to different components of typical SLAM systems, and the solutions proposed for these problems have relied on a variety of mathematical techniques and design philosophies common in robotics. These include outlier rejection, portfolios of models, abstraction of constraint types within pose-graph optimization, specialized feature detection, supervised learning, and sequential decision making under uncertainty. Chapters 3-6 focus on SLAM *algorithms* for robust perception, dealing with imperfect information and noisy sensor readings. Several different mathematical models are proposed that allow better filtering of outliers (Chapter 3), better management and application of portfolios of models (Chapter 4), augmentation of graphical models to include information originating from multiple sources (Chapter 5), and specialization of several SLAM algorithms for regular, indoor environments with low quality sensors and very limited computation (Chapter 6).

Specifically, in Chapter 3 we discuss an approach to creating blue-print style maps of permanent features of an environment from data collected over multiple deployments. The key benefit of this approach is that it constructs maps that allow more reliable localization since observations from transient objects in the environment are less likely to be erroneously matched to features in the map. Thus, with the proper scheme for rejecting observations when computing $p(z|x)$, the robot is less likely to fundamentally misinterpret its observations. We also take care to minimize the memory footprint of the map and make it scale with area explored rather than duration of deployment, all while retaining machine-level precision and uncertainty estimates.

In Chapter 4 we discuss an approach for lane identification of autonomous vehicles under topological uncertainty, where the main innovation is to reduce reliance on maps of the roadway system, which for large-scale operations are difficult to keep up-to-date. We do this by removing the assumption that the given map is perfect, and instead use the map as a prior belief about the topology of the road, over which there is some uncertainty. This uncertainty is handled by employing a population of models that reason about whether the current observations contradict the topology implied by the model and then use the best fitting model to represent the map at any given moment. Another innovation presented is that, in addition to observations of the environment such as lane markings, we also use observations of other vehicles on the road under the assumption that their behavior is also conditioned on the topological structure of the road. This allows further reduction of reliance on expensive maps and global sensors like GPS since observations of other agents often compliment lane marking data.

In Chapter 5 we discuss an approach to augmenting typical graph-SLAM with data from a human, which we term Human-in-the-Loop SLAM. It is relatively easy for even non-experts to identify errors maps, but it is not obvious how to translate map errors identified by humans into additional constraints on the robot's trajectory. Human-in-the-Loop SLAM takes a small amount of human input, creates new, potentially rank-deficient constraints, adds them to the original optimization problem, and re-solves the problem to generate a more accurate map without the need to re-collect data or increase the solver's compute budget. This not only drastically reduces the cost of generating an initially poor map, but it also allows for types of constraints, between parts of the environment that may have only been observed once, that are normally impossible to generate even with perfect data association.

In Chapter 6 we discuss an entire SLAM system designed for detecting, matching, optimizing, and compressing features easily detected with very inexpensive depth sensors. Due to the limitations of such sensors, many of the most reliably detectable features do not fully constrain a robot's relative motion from frame to frame are thus rank-deficient. Rank deficiency (in 2D and 3D) poses challenges both for calculating feature correspondences as well as formulating robust optimization processes. We provide methods for both challenges as well as methods for storage and manipulation of map items, similar to methods discussed in Chapter 3, but with the added difficulty of doing so incrementally online rather than all at once during post-processing.

Throughout, a key focus has been on using readily available or existing resources, often in the form of robustly detectable information, easily supplied human feedback, or existing code to overcome challenging perception problems. Many of these projects could be categorized as benefiting from different forms of specialization, and while specialization is not the only way to improve robotic perception performance, there have already been many works on the core mathematical concepts underpinning state estimation, making further research relatively less likely to offer substantial marginal benefit. In addition, the fact that robotic perception systems so clearly benefit from relatively minor additional assumptions about their operating environments makes understanding patterns and synergies within and between these specialized methods an important goal in the realization of performant robotic systems; and this thesis represents a small step towards these goals. There is perhaps no better class of SLAM algorithms that represent these truths than those that compose SLAM front ends, particularly feature detection and correspondence calculation.

Chapters 7-9, motivated by insights from work on the systems presented Chapters 3-6, present several key building blocks in the construction of Multi-SLAM systems that allow effective dynamic integration of multiple SLAM front ends. These include predictive front end performance models (Chapter 7), belief-space planning for online SLAM front end selection (Chapter 8), and graphical modeling and factor graph construction that handles multiple, asynchronous, multi-modal front end outputs (Chapter 9). Together, these components represent a novel and effective strategy for robust SLAM.

In Chapter 7, we discuss how to learn performance models of SLAM algorithms that can predict the magnitude of error in the rotation and translation estimates of SLAM systems conditioned on both the current sensor input as well as the features extracted from the input and a representation of the state of the solver prior to the additional of the newest information. They key insight is that such a model is not only learnable using deep convolutional neural networks, but also that even imperfect performance models, if their uncertainty estimates are well-calibrated, can supply a useful signal to a planner. Various other ablation studies regarding architecture and loss functions, and issues with creating balanced training data sets, are also presented.

In Chapter 8, we discuss the formulation of a belief-space planner for online SLAM front end selection. Key complexities are identified with respect to the well-studied algorithm selection problem, including recursion, reactivity, and indefiniteness, that necessitate an updated problem formulation for which belief-space planning within a POMDP is a natural solution. The resulting novel decision-making problem captures trade offs when choosing between SLAM front end algorithms online and allows us to solve the problem in a manner that, in expectation, minimizes the total error in the trajectory. One particularly nice property of this method is that regardless of the quality of the underlying data, front ends, or performance models, as long as the uncertainty estimate from the performance models is well-calibrated, the planner will make the optimal choices.

In Chapter 9, we discuss two key issues related to the implementation of Multi-SLAM Systems. While the abstract dynamic selection of algorithms may in theory reduce trajectory estimate error, this technique creates several complex modeling and inference problems. This chapter describes new variations of dynamic Bayesian networks needed to model the process of switching between SLAM front end algorithms and new techniques for stitching together maps made using different modalities.

10.2 Future Work

If mobile robotic systems of the future will require SLAM capabilities, which at the moment seems far more likely than not, then these systems will either A) be monolithic in nature, drawing on a fixed set of sensor inputs combined according to a fixed algorithm determined pre-deployment or B) they will not be monolithic, instead adapting, combining, or selecting methods of signal processing and inference dynamically as the robot operates. How exactly they may do this is not fully constrained at present, for example systems may perform 'hard' switches between algorithms, as proposed in this thesis, they may include and reject subsets of outputs from different front-ends, or they may employ a dynamic weighting scheme over front-end data. However, one fact is certain: there is no third option, middle ground, or hybrid solution; SLAM systems either have the capability to dynamically change their reliance on the present data, or they do not.

The overwhelming majority of SLAM research over the last 4 decades, indeed all but a handful of papers, primarily on robust optimization, have investigated hypotheses related solely to scenario A. The work and ideas presented in this thesis, to the best of my knowledge, represent a significant fraction of the academic literature regarding the feasibility of option B. Though there are many other works that use POMDPs to reason about different control options for other robotic tasks, the new class of SLAM systems proposed, which we call 'Multi-SLAM' systems, offers a distinctly new paradigm of research in SLAM and begets several promising research directions in both SLAM and robotics at large.

Future research on Multi-SLAM systems has many possible avenues, with several in particular roughly corresponding to the problems outlined in chapters 7-9. Better performance prediction, better planners, and better integration between individual SLAM systems will each significantly increase the capability of Multi-SLAM systems. Multi-SLAM systems also stand to benefit from an ever increasing volume of open-source SLAM software, and efforts to organize these disparate code bases for use within Multi-SLAM systems could be very impactful. More robust SLAM systems also have knock on positive effects on the study of other robotics problems, since SLAM capable robots can be deployed under a larger number of conditions and thus allow us to access new experimental domains in order to test new hypotheses.

Furthermore, many of the general principles presented here need not be unique to SLAM systems, and a hypothesis for future consideration is whether this general approach, where decisions are made over libraries of highly specialized, off-the-shelf algorithms, can be applied effectively to other quintessential robotics problems for which definitive, universal solutions do not seem near at hand. The work presented in the latter chapters of this thesis may provide a foundation for the study and development of many other versions of 'Multi-X' systems targeting different modules within the robotics stack. The idea of leveraging the redundancy that modularity and specialization produce within robotics, along with decision making over the space of existing algorithms, may be extended to a variety of robotics sub-tasks as a realistic option for greater robustness and generalization. There are many other perception, planning, and control tasks which, much like SLAM, share a simple mathematical formulation but may be solved in a variety of operational contexts, thus producing a vast number of different problem instances with different, specialized solutions. All such problems, such as grasping, motion planning, or object tracking, may in theory be target applications for the same type of multi-algorithm system that generalizes Multi-SLAM. Although systems for these problems that 'just work' may not be imminent, the time for seriously considering such systems in our collective research has certainly arrived.

10.3 Final Thoughts

There are very few low-level problems unique to robotics; most have origins in other fields, like computer vision, planning, or control. There are also very few high-level tasks. In fact, some may argue that there are only two: getting from point A to point B, and manipulating objects. However, the incredible range of different conditions under which these tasks may be performed and the unpredictability of operating in the physical world have, rightly so, lead to a large number of specialized modules that each take advantage of structures within different types of data. As a core component of getting from point A to point B, the SLAM problem, whether as currently formulated or re-envisioned, will need to be solved.

With respect to localization and state estimation we already have many powerful tools, and while no solution method is strictly best, their trade offs are relatively well understood and quantified. However, although not new, the mapping component of SLAM has proven to be even more formidable and has captivated me from the beginning of my interest in robotics. In my opinion, this difficulty stems from the fact that the generative models that explain the dynamics of the world and how it appears require knowledge at many levels of abstraction to describe correctly. Without such models the ‘data association problem’, the problem of reasoning about whether or not a given observation originates from a previously observed object or place in the world, is extremely challenging and SLAM as a whole becomes much more precarious. Currently, I consider the data association problem to be the most immediate challenge for deploying robots reliably in the open world.

While AI and robotics researchers are often very good at choosing a single level of abstraction for a given problem, having to build data structures and update models that support a wide range of phenomena is an area where there is still a lot of work to be done. Moreover, maps generated via SLAM will likely eventually go beyond just supporting state estimation, and so far researchers are generally still searching for the most appropriate data to store and the best representations for that data. This is most apparent in many recent threads of work on using multi-modal data, particularly text, RGB, haptic, and depth, to learn, build, or encode, knowledge about how stimuli from these different modes might co-occur. These attempts are still in the relative early stages and the effect of multi-modal data on modeling for robotics remains to be seen. Learning and representing types of knowledge about the world other than spatial and topological have drawn attention from many fields beyond robotics and while this is good in that it raises the probability of useful discovery, I also understand it as an indication of the fundamental difficulty of this problem.

An additional, perhaps more mundane, challenge on the horizon is that of experimentation. Most other fields of AI, like planning or learning, have access to optimal solutions, simulators, or labeled data sets. If the goal of a SLAM system is state estimation in the real world and kilometer scale, then getting ground truth often requires prohibitively expensive sensors or external systems. Moreover, as SLAM systems become more advanced they become harder to program and this is exacerbated when there are other constraints such a computation, memory, or sensor quality, all of which are very natural extensions of understandable cost constraints that virtually every deployed system will experience. It is possible that, together, these challenges are producing an environment where it is increasingly difficult for small-scale experiments or those localized to a specific SLAM algorithm to be impactful. Because there are a huge number of different ways SLAM algorithms can be connected and integrated to form different SLAM system architectures, the importance of a particular algorithm is limited to scenarios where that algorithm is selected by a developer for a particular SLAM system. What these changes are producing, however, may be the forerunner to a more formal study of SLAM system architecture in which Multi-SLAM systems could be an archetype, or possibly even larger robotic architectures more generally.

The study of robotics, as I have come to understand it, is largely a science of *integration*. Understanding how to get a distributed system of sensors, computers, and motors to understand and interact with the world is largely a problem of understanding how to compose these many algorithms together in a manner that reduces the system's susceptibility to the perturbations in input that come with operating in a complex and unpredictable environment. This type of meta-level question is itself probably the most unique aspect of robotics, and to me is one of the most compelling reasons why approaches like Multi-SLAM are worth investigating; they represent a microcosm of some of the most important questions roboticists need to answer in order to achieve reliable systems.

Moreover, this problem formulation offers a unique and convenient level of abstraction for reasoning about tasks at many points in the robotics stack, and takes advantage of the rich tradition of specialization and collective desire for modularity within engineering and robotics systems. Rather than needing to invent a unifying theory of a task (e.g. localization, grasp planning, etc.) for every task we can instead embrace the heterogeneity of existing algorithms and capabilities and unify only the process of reasoning about which one is appropriate for the current task. Or, more broadly, we can study how to compose solutions to many small problems into a larger architecture that leads to far more robust system behavior.

BIBLIOGRAPHY

[1] Abd Almisreb, Ali, Jamil, Nursuriati, and Din, N Md. Utilizing AlexNet deep transfer learning for ear recognition. In *International Conference on Information Retrieval and Knowledge Management* (2018), pp. 1–5.

[2] Agarwal, Pratik, and Olson, Edwin. Variable reordering strategies for SLAM. In *International Conference on Intelligent Robots and Systems (IROS)* (2012), pp. 3844–3850.

[3] Agarwal, Sameer, and Mierle, Keir. Ceres solver: Tutorial & reference. *Google Inc 2* (2012), 8.

[4] Akhlaghi, Shahrokh, Zhou, Ning, and Huang, Zhenyu. Adaptive adjustment of noise covariance in Kalman filter for dynamic state estimation. In *IEEE Power & Energy Society General Meeting* (2017), pp. 1–5.

[5] Alahi, Alexandre, Ortiz, Raphael, and Vandergheynst, Pierre. FREAK: Fast retina keypoint. In *Conference on Computer Vision and Pattern Recognition (CVPR)* (2012), pp. 510–517.

[6] Alcantarilla, Pablo F, and Solutions, T. Fast explicit diffusion for accelerated features in nonlinear scale spaces. *IEEE Transactions on Pattern Analysis and Machince Intelligence 34* (2011), 1281–1298.

[7] Alejandro, Arbelaez, Youssef, Hamadi, and Michele, Sebag. Online heuristic selection in constraint programming. In *International Symposium on Combinatorial Search (SOCS)* (2009).

[8] Ali, Islam, Wan, Bingqing (Selina), and Zhang, Hong. Prediction of SLAM ATE using an ensemble learning regression model and 1-D global pooling of data characterization. *arXiv preprint arXiv:2303.00616* (2023).

[9] Alsayed, Zayed, Bresson, Guillaume, Verroust-Blondet, Anne, and Nashashibi, Fawzi. 2D SLAM correction prediction in large scale urban environments. In *International Conference on Robotics and Automation (ICRA)* (2018), pp. 5167–5174.

[10] Alshawa, Majd. ICL: Iterative closest line a novel point cloud registration algorithm based on linear features. *Ekscentar* (2007), 53–59.

[11] Alspach, Daniel, and Sorenson, Harold. Nonlinear Bayesian estimation using Gaussian sum approximations. *Transactions on Automatic Control 17* (1972), 439–448.

[12] Altman, Eitan. *Constrained Markov decision processes*, vol. 7. 1999.

[13] An, Su-Yong, Kang, Jeong-Gwan, Lee, Lae-Kyoung, and Oh, Se-Young. SLAM with salient line feature extraction in indoor environments. In *International Conference on Control Automation Robotics & Vision* (2010), pp. 410–416.

[14] Andreasson, Henrik, and Duckett, Tom. Topological localization for mobile robots using omni-directional vision and local features. In *International Federation of Automatic Control* (2004).

[15] Arasaratnam, Ienkaran, and Haykin, Simon. Cubature Kalman filters. *IEEE Transactions on Automatic Control 54* (2009), 1254–1269.

[16] Arasaratnam, Ienkaran, Haykin, Simon, and Elliott, Robert J. Discrete-time nonlinear filtering algorithms using Gauss–Hermite quadrature. *Proceedings of the IEEE 95* (2007), 953–977.

[17] Arbelaez, Pablo, Maire, Michael, Fowlkes, Charless, and Malik, Jitendra. Contour detection and hierarchical image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 33* (2010), 898–916.

[18] Arbuckle, Daniel, Howard, Andrew, and Mataric, Maja. Temporal occupancy grids: A method for classifying the spatio-temporal properties of the environment. In *International Conference on Intelligent Robots and Systems (IROS)* (2002).

[19] Atanasov, Nikolay, Le Ny, Jerome, Daniilidis, Kostas, and Pappas, George J. Decentralized active information acquisition: Theory and application to multi-robot SLAM. In *International Conference on Robotics and Automation (ICRA)* (2015), pp. 4775–4782.

[20] Aulinas, Josep, Petillot, Yvan R, Salvi, Joaquim, and Lladó, Xavier. The SLAM problem: A survey. In *Artificial Intelligence Research and Development* (2008), pp. 363–371.

[21] Awrangjeb, Mohammad, Lu, Guojun, and Fraser, Clive S. Performance comparisons of contour-based corner detectors. *IEEE Transactions on Image Processing 21* (2012), 4167–4179.

[22] Bai, Fang, Vidal-Calleja, Teresa, and Grisetti, Giorgio. Sparse pose graph optimization in cycle space. *IEEE Transactions on Robotics 37* (2021), 1381–1400.

[23] Bailey, Tim, and Durrant-Whyte, Hugh. Simultaneous localization and mapping (SLAM): Part II. *IEEE Robotics & Automation Magazine 13* (2006), 108–117.

[24] Balch, Tucker, and Arkin, Ronald C. Behavior-based formation control for multi-robot teams. *IEEE Transactions on Robotics and Automation 14* (1998), 926–939.

[25] Balntas, Vassileios, Li, Shuda, and Prisacariu, Victor. Relocnet: Continuous metric learning relocalisation using neural nets. In *European Conference on Computer Vision (ECCV)* (2018), pp. 751–767.

[26] Balntas, Vassileios, Riba, Edgar, Ponsa, Daniel, and Mikolajczyk, Krystian. Learning local feature descriptors with triplets and shallow convolutional neural networks. In *British Machine Vision Conference* (2016).

[27] Bay, Herbert, Tuytelaars, Tinne, and Van Gool, Luc. SURF: Speeded up robust features. In *European Conference on Computer Vision (ECCV)* (2006), pp. 404–417.

[28] Behzadian, Bahram, Agarwal, Pratik, Burgard, Wolfram, and Tipaldi, Gian Diego. Monte Carlo localization in hand-drawn maps. In *International Conference on Intelligent Robots and Systems (IROS)* (2015), pp. 4291–4296.

[29] Bellman, Richard. Dynamic programming. *Science* (1966).

[30] Ben-Ari, Moti. A tutorial on Euler angles and quaternions. *Weizmann Institute of Science 524* (2014).

[31] Bengtsson, Thomas, Bickel, Peter, Li, Bo, et al. Curse-of-dimensionality revisited: Collapse of the particle filter in very large scale systems. *Probability and statistics: Essays in honor of David A. Freedman 2* (2008), 316–334.

[32] Berler, Ami, and Shimony, Solomon Eyal. Bayes networks for sensor fusion in occupancy grids. In *Conference on Uncertainty in Artificial Intelligence (UAI)* (1997).

[33] Bernstein, Daniel S, Givan, Robert, Immerman, Neil, and Zilberstein, Shlomo. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research 27* (2002), 819–840.

[34] Bertsekas, Dimitri P. Approximate policy iteration: A survey and some new methods. *Journal of Control Theory and Applications 9* (2011), 310–335.

[35] Besl, Paul J, and McKay, Neil D. Method for registration of 3-D shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures* (1992), vol. 1611, International Society for Optics and Photonics.

[36] Bhatia, Abhinav, Nashed, Samer B, and Zilberstein, Shlomo. RL^3: Boosting meta reinforcement learning via RL inside RL^2. *arXiv preprint arXiv:2306.15909* (2023).

[37] Bhatia, Abhinav, Svegliato, Justin, Nashed, Samer B, and Zilberstein, Shlomo. Tuning the hyperparameters of anytime planning: A metareasoning approach with deep reinforcement learning. In *International Conference on Automated Planning and Scheduling (ICAPS)* (2022), pp. 556–564.

[38] Biber, Peter. Dynamic maps for long-term operation of mobile service robots. In *Robotics: Science and Systems (RSS)* (2005).

[39] Biswas, Joydeep, and Veloso, Manuela. Planar polygon extraction and merging from depth images. In *International Conference on Intelligent Robots and Systems (IROS)* (2012).

[40] Biswas, Joydeep, and Veloso, Manuela. Episodic non-Markov localization: Reasoning about short-term and long-term features. In *International Conference on Robotics and Automation (ICRA)* (2014).

[41] Biswas, Joydeep, and Veloso, Manuela M. Episodic non-Markov localization. *Robotics and Autonomous Systems 87* (2017), 162–176.

[42] Biswas, Rahul, Limketkai, Benson, Sanner, Scott, and Thrun, Sebastian. Towards object mapping in non-stationary environments with mobile robots. In *International Conference on Intelligent Robots and Systems (IROS)* (2002).

[43] Biza, Ondrej, and Platt, Robert. Online abstraction with MDP homomorphisms for deep learning. *arXiv preprint arXiv:1811.12929* (2018).

[44] Blackwell, David. Conditional expectation and unbiased sequential estimation. *The Annals of Mathematical Statistics* (1947), 105–110.

[45] Boniardi, Federico, Valada, Abhinav, Burgard, Wolfram, and Tipaldi, Gian Diego. Autonomous indoor robot navigation using a sketch interface for drawing maps and routes. In *International Conference on Robotics and Automation (ICRA)* (2016), pp. 2896–2901.

[46] Braziunas, Darius. POMDP solution methods. *University of Toronto* (2003).

[47] Bromley, Jane, Guyon, Isabelle, LeCun, Yann, Säckinger, Eduard, and Shah, Roopak. Signature verification using a "Siamese" time delay neural network. In *Advances in Neural Information Processing Systems* (1993).

[48] Bruno, Hudson Martins Silva, and Colombini, Esther Luna. LIFT-SLAM: A deep-learning feature-based monocular visual SLAM method. *Neurocomputing 455* (2021), 97–110.

[49] Bylow, Erik, Sturm, Jurgen, Kerl, Christian, Kahl, Fredrik, and Cremers, Daniel. Real-time camera tracking and 3D reconstruction using signed distance functions. In *Robotics: Science and Systems (RSS)* (2013).

[50] Cadena, Cesar, Carlone, Luca, Carrillo, Henry, Latif, Yasir, Scaramuzza, Davide, Neira, José, Reid, Ian, and Leonard, John J. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics 32* (2016), 1309–1332.

[51] Calonder, Michael, Lepetit, Vincent, Strecha, Christoph, and Fua, Pascal. BRIEF: Binary robust independent elementary features. In *European Conference on Computer Vision (ECCV)* (2010), pp. 778–792.

[52] Carlevaris-Bianco, Nicholas, and Eustice, Ryan M. Generic factor-based node marginalization and edge sparsification for pose-graph SLAM. In *International Conference on Robotics and Automation (ICRA)* (2013), pp. 5748–5755.

[53] Carlone, Luca, Aragues, Rosario, Castellanos, José A, and Bona, Basilio. A fast and accurate approximation for planar pose graph optimization. *International Journal of Robotics Research 33* (2014), 965–987.

[54] Carlone, Luca, Du, Jingjing, Kaouk Ng, Miguel, Bona, Basilio, and Indri, Marina. Active SLAM and exploration with particle filters using Kullback-Leibler divergence. *Journal of Intelligent & Robotic Systems 75* (2014), 291–311.

[55] Carson, Helen, Ford, Jason J, and Milford, Michael. Predicting to improve: Integrity measures for assessing visual localization performance. *Robotics and Automation Letters 7* (2022), 9627–9634.

[56] Chaplot, Devendra Singh, Gandhi, Dhiraj, Gupta, Saurabh, Gupta, Abhinav, and Salakhutdinov, Ruslan. Learning to explore using active neural SLAM. *arXiv preprint arXiv:2004.05155* (2020).

[57] Chatila, Raja, and Laumond, J. Position referencing and consistent world modeling for mobile robots. In *International Conference on Robotics and Automation (ICRA)* (1985), vol. 2, pp. 138–145.

[58] Chaves, Stephen M, Kim, Ayoung, Galceran, Enric, and Eustice, Ryan M. Opportunistic sampling-based active visual SLAM for underwater inspection. *Autonomous Robots 40* (2016), 1245–1265.

[59] Chen, SY. Kalman filter for robot vision: A survey. *IEEE Transactions on Industrial Electronics 59* (2011), 4409–4420.

[60] Chen, Tao, Gupta, Saurabh, and Gupta, Abhinav. Learning exploration policies for navigation. *arXiv preprint arXiv:1903.01959* (2019).

[61] Chen, Weinan, Zhu, Lei, Gu, Shichao, and Zhang, Hong. CAMs-SLAM: Cloud-based multi-submap VSLAM for multi-source asynchronous sensing of biped climbing robots. *Sensors* (2023).

[62] Chen, Xieyuanli, Lu, Huimin, Xiao, Junhao, Zhang, Hui, and Wang, Pan. Robust relocalization based on active loop closure for real-time monocular SLAM. In *International Conference on Computer Vision Systems (ICVS)* (2017), pp. 131–143.

[63] Chen, Yang, and Medioni, Gérard. Object modelling by registration of multiple range images. *Image and Vision Computing 10* (1992), 145–155.

[64] Chetverikov, Dmitry, Svirko, Dmitry, Stepanov, Dmitry, and Krsek, Pavel. The trimmed iterative closest point algorithm. In *International Conference on Pattern Recognition (ICPR)* (2002).

[65] Chiou, Manolis, Stolkin, Rustam, Bieksaite, Goda, Hawes, Nick, Shapiro, Kimron L, and Harrison, Timothy S. Experimental analysis of a variable autonomy framework for controlling a remotely operating mobile robot. In *International Conference on Intelligent Robots and Systems (IROS)* (2016), pp. 3581–3588.

[66] Cho, HyunGi, Yeon, Suyong, Choi, Hyunga, and Doh, Nakju. Detection and compensation of degeneracy cases for IMU-Kinect integrated continuous SLAM with plane features. *Sensors 18* (2018), 935.

[67] Cho, Kyunghyun, van Merriënboer, Bart, Gulcehre, Caglar, Bahdanau, Dzmitry, Bougares, Fethi, Schwenk, Holger, and Bengio, Yoshua. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing* (2014), pp. 1724–1734.

[68] Choi, Young-Ho, Lee, Tae-Kyeong, and Oh, Se-Young. A line feature based SLAM with low grade range sensors using geometric constraints and active exploration for mobile robot. *Autonomous Robots 24* (2008), 13–27.

[69] Choset, Howie, and Nagatani, Keiji. Topological simultaneous localization and mapping (SLAM): Toward exact localization without explicit localization. *IEEE Transactions on Robotics and Automation 17* (2001).

[70] Chou, Chih-Chung, and Chou, Cheng-Fu. Efficient and accurate tightly-coupled visual-lidar SLAM. *IEEE Transactions on Intelligent Transportation Systems 23* (2021), 14509–14523.

[71] Cioffi, Giovanni, Cieslewski, Titus, and Scaramuzza, Davide. Continuous-time vs. discrete-time vision-based SLAM: A comparative study. *Robotics and Automation Letters 7* (2022), 2399–2406.

[72] Concha Belenguer, Alejo, and Civera Sancho, Javier. DPPTAM: Dense piecewise planar tracking and mapping from a monocular sequence. In *International Conference on Intelligent Robots and Systems (IROS)* (2015).

[73] Curless, Brian, and Levoy, Marc. A volumetric method for building complex models from range images. In *ACM* (1996).

[74] Cvišić, Igor, Marković, Ivan, and Petrović, Ivan. SOFT2: Stereo visual odometry for road vehicles based on a point-to-epipolar-line metric. *IEEE Transactions on Robotics 39* (2022), 273–288.

[75] Daftry, Shreyansh, Zeng, Sam, Bagnell, J Andrew, and Hebert, Martial. Introspective perception: Learning to predict failures in vision systems. In *International Conference on Intelligent Robots and Systems (IROS)* (2016).

[76] Dalal, Navneet, and Triggs, Bill. Histograms of oriented gradients for human detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)* (2005), pp. 886–893.

[77] Daniilidis, Konstantinos. Hand-eye calibration using dual quaternions. *International Journal of Robotics Research 18* (1999), 286–298.

[78] Datta Gupta, Syamantak. A comparative study of the particle filter and the ensemble Kalman filter. Master's thesis, University of Waterloo, 2009.

[79] Dayoub, Feras, and Duckett, Tom. An adaptive appearance-based map for long-term topological localization of mobile robots. In *International Conference on Intelligent Robots and Systems (IROS)* (2008).

[80] Dean, Thomas L, Givan, Robert, and Leach, Sonia. Model reduction techniques for computing approximately optimal solutions for Markov decision processes. *arXiv preprint arXiv:1302.1533* (1997).

[81] Degerman, Johan, Pernstål, Thomas, and Alenljung, Klas. 3D occupancy grid mapping using statistical radar models. In *Intelligent Vehicles Symposium (IV)* (2016), pp. 902–908.

[82] Del Moral, Pierre. Nonlinear filtering: Interacting particle resolution. *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics 325* (1997), 653–658.

[83] Del Moral, Pierre. Measure-valued processes and interacting particle systems. application to nonlinear filtering problems. *The Annals of Applied Probability 8* (1998), 438–495.

[84] Dellaert, Frank, and Kaess, Michael. Square root SAM: Simultaneous localization and mapping via square root information smoothing. *International Journal of Robotics Research 25* (2006), 1181–1203.

[85] Dempster, A. P., Laird, N. M., and Rubin, D. B. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological) 39* (1977), 1–38.

[86] DeTone, Daniel, Malisiewicz, Tomasz, and Rabinovich, Andrew. Toward geometric deep SLAM. *arXiv preprint arXiv:1707.07410* (2017).

[87] DeTone, Daniel, Malisiewicz, Tomasz, and Rabinovich, Andrew. SuperPoint: Self-supervised interest point detection and description. In *Conference on Computer Vision and Pattern Recognition Workshops* (2018).

[88] Diosi, Albert, Taylor, Geoffrey, and Kleeman, Lindsay. Interactive SLAM using laser and advanced sonar. In *International Conference on Robotics and Automation (ICRA)* (2005), pp. 1103–1108.

[89] Diryag, Ali, Mitić, Marko, and Miljković, Zoran. Neural networks for prediction of robot failures. *Journal of Mechanical Engineering Science 228* (2014), 1444–1458.

[90] Dissanayake, Gamini, Huang, Shoudong, Wang, Zhan, and Ranasinghe, Ravindra. A review of recent developments in simultaneous localization and mapping. In *International Conference on Industrial and Information Systems* (2011), pp. 477–482.

[91] Dissanayake, MWM Gamini, Newman, Paul, Clark, Steve, Durrant-Whyte, Hugh F, and Csorba, Michael. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation 17* (2001), 229–241.

[92] Dollar, Piotr, Tu, Zhuowen, and Belongie, Serge. Supervised learning of edges and object boundaries. In *Conference on Computer Vision and Pattern Recognition (CVPR)* (2006).

[93] Dollár, Piotr, and Zitnick, C Lawrence. Structured forests for fast edge detection. In *International Conference on Computer Vision (ICCV)* (2013).

[94] Doroodgar, Barzin, Ficocelli, Maurizio, Mobedi, Babak, and Nejat, Goldie. The search for survivors: Cooperative human-robot interaction in search and rescue environments using semi-autonomous robots. In *International Conference on Robotics and Automation (ICRA)* (2010), pp. 2858–2863.

[95] Dubbelman, G., and Browning, B. COP-SLAM: Closed-form online pose-chain optimization for visual SLAM. *IEEE Transactions on Robotics 31* (2015).

[96] Dusmanu, Mihai, Rocco, Ignacio, Pajdla, Tomas, Pollefeys, Marc, Sivic, Josef, Torii, Akihiko, and Sattler, Torsten. D2-Net: A trainable CNN for joint description and detection of local features. In *Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).

[97] Elfes, A. Using occupancy grids for mobile robot perception and navigation. *Computer* (1989).

[98] Elfes, Alberto. Sonar-based real-world mapping and navigation. *IEEE Journal on Robotics and Automation 3* (1987), 249–265.

[99] Elman, Jeffrey L. Finding structure in time. *Cognitive Science 14* (1990), 179–211.

[100] Eustice, Ryan, Singh, Hanumant, Leonard, John J, Walter, Matthew R, and Ballard, Robert. Visually navigating the RMS Titanic with SLAM information filters. In *Robotics: Science and Systems (RSS)* (2005), pp. 57–64.

[101] Farid, Alec, Snyder, David, Ren, Allen Z, and Majumdar, Anirudha. Failure prediction with statistical guarantees for vision-based robot control. *arXiv preprint arXiv:2202.05894* (2022).

[102] Ferns, Norm, Panangaden, Prakash, and Precup, Doina. Metrics for finite Markov decision processes. In *Conference on Uncertainty in Artificial Intelligence (UAI)* (2004), pp. 162–169.

[103] Fine, Shai, Singer, Yoram, and Tishby, Naftali. The hierarchical hidden Markov model: Analysis and applications. *Machine learning 32* (1998), 41–62.

[104] Fischler, Martin A., and Bolles, Robert C. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *ACM* (1981).

[105] Folkesson, John, and Christensen, Henrik. Graphical SLAM-a self-correcting map. In *International Conference on Robotics and Automation (ICRA)* (2004), pp. 383–390.

[106] Forster, Christian, Carlone, Luca, Dellaert, Frank, and Scaramuzza, Davide. IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. Tech. rep., 2015.

[107] Franke, Uwe, Pfeiffer, David, Rabe, Clemens, Knoeppel, Carsten, Enzweiler, Markus, Stein, Fridtjof, and Herrtwich, Ralf. Making bertha see. In *International Conference on Computer Vision Workshops* (2013).

[108] Galindo, Cipriano, Saffiotti, Alessandro, Coradeschi, Silvia, Buschka, Pär, Fernandez-Madrigal, Juan-Antonio, and González, Javier. Multi-hierarchical semantic maps for mobile robotics. In *International Conference on Intelligent Robots and Systems (IROS)* (2005), pp. 2278–2283.

[109] Geiger, Andreas, Lenz, Philip, and Urtasun, Raquel. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)* (2012).

[110] Geneva, Patrick, Eckenhoff, Kevin, and Huang, Guoquan. Asynchronous multi-sensor fusion for 3D mapping and localization. In *International Conference on Robotics and Automation (ICRA)* (2018), pp. 5994–5999.

[111] Giamou, Matthew, Khosoussi, Kasra, and How, Jonathan P. Talk resource-efficiently to me: Optimal communication planning for distributed loop closure detection. In *International Conference on Robotics and Automation (ICRA)* (2018), pp. 1–9.

[112] Givan, Robert, Dean, Thomas, and Greig, Matthew. Equivalence notions and model minimization in Markov decision processes. *Artificial Intelligence 147* (2003), 163–223.

[113] González, R, Rodrıguez, F, Guzman, JL, and Berenguel, M. Comparative study of localization techniques for mobile robots based on indirect Kalman filter. In *IFR International Symposium on Robotics* (2009), pp. 253–258.

[114] Gordon, Neil J, Salmond, David J, and Smith, Adrian FM. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. In *IEE Proceedings F (Radar and Signal Processing)* (1993), pp. 107–113.

[115] Gough, Julian, Karplus, Kevin, Hughey, Richard, and Chothia, Cyrus. Assignment of homology to genome sequences using a library of hidden Markov models that represent all proteins of known structure. *Journal of Molecular Biology 313* (2001), 903–919.

[116] Grisetti, Giorgio, Kümmerle, Rainer, Strasdat, Hauke, and Konolige, Kurt. g2o: A general framework for (hyper) graph optimization. In *International Conference on Robotics and Automation (ICRA)* (2011), pp. 9–13.

[117] Gu, Xuefeng, Wang, Yafei, and Ma, Taiyuan. DBLD-SLAM: A deep-learning visual SLAM system based on deep binary local descriptor. In *International Conference on Control, Automation and Information Sciences* (2021).

[118] Guestrin, Carlos, Koller, Daphne, Parr, Ronald, and Venkataraman, Shobha. Efficient solution algorithms for factored MDPs. *Journal of Artificial Intelligence Research 19* (2003), 399–468.

[119] Guo, Chuan, Pleiss, Geoff, Sun, Yu, and Weinberger, Kilian Q. On calibration of modern neural networks. In *International Conference on Machine Learning (ICML)* (2017), pp. 1321–1330.

[120] Guo, Jiadong, Borges, Paulo VK, Park, Chanoh, and Gawel, Abel. Local descriptor for robust place recognition using lidar intensity. *Robotics and Automation Letters 4* (2019), 1470–1477.

[121] Hahnel, D., Schulz, D., and Burgard, W. Map building with mobile robots in populated environments. In *International Conference on Intelligent Robots and Systems (IROS)* (2002).

[122] Hammersley, John M, and Morton, K William. Poor man's Monte Carlo. *Journal of the Royal Statistical Society: Series B (Methodological) 16* (1954), 23–38.

[123] Han, Xiao, Tao, Yulin, Li, Zhuyi, Cen, Ruping, and Xue, Fangzheng. SuperPointVO: A lightweight visual odometry based on CNN feature extraction. In *International Conference on Automation, Control and Robotics Engineering* (2020).

[124] Han, Xufeng, Leung, Thomas, Jia, Yangqing, Sukthankar, Rahul, and Berg, Alexander C. Matchnet: Unifying feature and metric learning for patch-based matching. In *Conference on Computer Vision and Pattern Recognition (CVPR)* (2015).

[125] Hart, Peter E, Nilsson, Nils J, and Raphael, Bertram. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics 4* (1968), 100–107.

[126] Hartmann, Jan, Klüssendorff, Jan Helge, and Maehle, Erik. A comparison of feature descriptors for visual SLAM. In *European Conference on Mobile Robots* (2013).

[127] Hastürk, Özgür, and Erkmen, Aydan M. DUDMap: 3D RGB-D mapping for dense, unstructured, and dynamic environment. *International Journal of Advanced Robotic Systems 18* (2021).

[128] He, Ruijie, Prentice, Sam, and Roy, Nicholas. Planning in information space for a quadrotor helicopter in a GPS-denied environment. In *International Conference on Robotics and Automation (ICRA)* (2008), pp. 1814–1820.

[129] He, Xudong, Zhao, Junqiao, Sun, Lu, Huang, Yewei, Zhang, Xinglian, Li, Jun, and Ye, Chen. Automatic vector-based road structure mapping using multi-beam LiDAR. In *Remote Sensing* (2018), pp. 417–422.

[130] Hemachandra, Sachithra, Walter, Matthew R, and Teller, Seth. Information theoretic question asking to improve spatial semantic representations. In *AAAI Fall Symposium* (2014).

[131] Hendrycks, Dan, and Gimpel, Kevin. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136* (2016).

[132] Hening, Sebastian, Ippolito, Corey A, Krishnakumar, Kalmanje S, Stepanyan, Vahram, and Teodorescu, Mircea. 3D LiDAR SLAM integration with GPS/INS for UAVs in urban GPS-degraded environments. In *AIAA Information Systems-AIAA Infotech Aerospace*. 2017.

[133] Hensel, Stefan, Hasberg, Carsten, and Stiller, Christoph. Probabilistic rail vehicle localization with eddy current sensors in topological maps. *IEEE Transactions on Intelligent Transportation Systems* (2011).

[134] Hinduja, Akshay, Ho, Bing-Jui, and Kaess, Michael. Degeneracy-aware factors with applications to underwater SLAM. In *International Conference on Intelligent Robots and Systems (IROS)* (2019), pp. 1293–1299.

[135] Ho, Bing-Jui, Sodhi, Paloma, Teixeira, Pedro, Hsiao, Ming, Kusnur, Tushar, and Kaess, Michael. Virtual occupancy grid map for submap-based pose graph SLAM and planning in 3d environments. In *International Conference on Intelligent Robots and Systems (IROS)* (2018), pp. 2175–2182.

[136] Ho, Yaoshiang, and Wookey, Samuel. The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling. *IEEE Access 8* (2019), 4806–4813.

[137] Hochreiter, Sepp, and Schmidhuber, Jürgen. Long short-term memory. *Neural Computation 9* (1997), 1735–1780.

[138] Hoffer, Elad, and Ailon, Nir. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition* (2015).

[139] Holder, Martin, Hellwig, Sven, and Winner, Hermann. Real-time pose graph SLAM based on radar. In *Intelligent Vehicles Symposium (IV)* (2019), pp. 1145–1151.

[140] Holmes, Steven A, Klein, Georg, and Murray, David W. An O(N^2) square root unscented Kalman filter for visual simultaneous localization and mapping. *IEEE Transactions on Pattern Analysis and Machine Intelligence 31* (2008), 1251–1263.

[141] Hong, Seonghun, Kim, Jinwhan, Pyo, Juhyun, and Yu, Son-Cheol. A robust loop-closure method for visual SLAM in unstructured seafloor environments. *Autonomous Robots 40* (2016), 1095–1109.

[142] Hosseinzadeh, Mehdi, Latif, Yasir, Pham, Trung, Suenderhauf, Niko, and Reid, Ian. Structure aware SLAM using quadrics and planes. In *Asian Conference on Computer Vision* (2018).

[143] Hou, Yi, Zhang, Hong, and Zhou, Shilin. Convolutional neural network-based image representation for visual loop closure detection. In *International Conference on Information and Automation* (2015).

[144] Hu, Haohao, Sackewitz, Lukas, and Lauer, Martin. Joint learning of feature detector and descriptor for visual SLAM. In *Intelligent Vehicles Symposium (IV)* (2021).

[145] Huang, Guoquan P, Mourikis, Anastasios I, and Roumeliotis, Stergios I. Analysis and improvement of the consistency of extended Kalman filter based SLAM. In *International Conference on Robotics and Automation (ICRA)* (2008), pp. 473–479.

[146] Huang, Guoquan P, Mourikis, Anastasios I, and Roumeliotis, Stergios I. A quadratic-complexity observability-constrained unscented Kalman filter for SLAM. *IEEE Transactions on Robotics 29* (2013), 1226–1243.

[147] Huberman, Bernardo A, Lukose, Rajan M, and Hogg, Tad. An economics approach to hard computational problems. *Science 275* (1997), 51–54.

[148] Hwang, Seo-Yeon, and Song, Jae-Bok. Monocular vision-based SLAM in indoor environment using corner, lamp, and door features from upward-looking camera. *IEEE Transactions on Industrial Electronics 58* (2011), 4804–4812.

[149] Indelman, Vadim, Williams, Stephen, Kaess, Michael, and Dellaert, Frank. Information fusion in navigation systems via factor graph based incremental smoothing. *Robotics and Autonomous Systems 61* (2013), 721–738.

[150] Jeong, WooYeon, and Lee, Kyoung Mu. CV-SLAM: A new ceiling vision-based SLAM technique. In *International Conference on Intelligent Robots and Systems (IROS)* (2005).

[151] Jessup, James, Givigi, Sidney N, and Beaulieu, Alain. Robust and efficient multirobot 3-D mapping merging with octree-based occupancy grids. *IEEE Systems Journal 11* (2015), 1723–1732.

[152] Jiao, Jianhao, Zhu, Yilong, Ye, Haoyang, Huang, Huaiyang, Yun, Peng, Jiang, Linxin, Wang, Lujia, and Liu, Ming. Greedy-based feature selection for efficient LiDAR SLAM. In *International Conference on Robotics and Automation (ICRA)* (2021).

[153] Julier, Simon J, and Uhlmann, Jeffrey K. New extension of the Kalman filter to nonlinear systems. In *Signal Processing, Sensor Fusion, and Target Recognition* (1997), vol. 3068, pp. 182–193.

[154] Kaelbling, Leslie Pack, Littman, Michael L., and Cassandra, Anthony R. Planning and acting in partially observable stochastic domains. *Journal of Artificial Intelligence Research* (1998).

[155] Kaess, Michael. Simultaneous localization and mapping with infinite planes. In *International Conference on Robotics and Automation (ICRA)* (2015), pp. 4605–4611.

[156] Kaess, Michael, Ila, Viorela, Roberts, Richard, and Dellaert, Frank. The Bayes tree: An algorithmic foundation for probabilistic robot mapping. In *Algorithmic Foundations of Robotics IX* (2010), pp. 157–173.

[157] Kaess, Michael, Johannsson, Hordur, Roberts, Richard, Ila, Viorela, Leonard, John J, and Dellaert, Frank. iSAM2: Incremental smoothing and mapping using the Bayes tree. *International Journal of Robotics Research 31* (2012), 216–235.

[158] Kaess, Michael, Ranganathan, Ananth, and Dellaert, Frank. iSAM: Incremental smoothing and mapping. *IEEE Transactions on Robotics 24* (2008), 1365–1378.

[159] Kallasi, Fabjan, Rizzini, Dario Lodi, and Caselli, Stefano. Fast keypoint features from laser scanner for robot localization and mapping. *Robotics and Automation Letters 1* (2016), 176–183.

[160] Kaneko, Masaya, Iwami, Kazuya, Ogawa, Toru, Yamasaki, Toshihiko, and Aizawa, Kiyoharu. Mask-SLAM: Robust feature-based monocular SLAM by masking using semantic segmentation. In *Conference on Computer Vision and Pattern Recognition Workshops* (2018).

[161] Kang, Rong, Shi, Jieqi, Li, Xueming, Liu, Yang, and Liu, Xiao. DF-SLAM: A deep-learning enhanced visual SLAM system based on deep local features. *arXiv preprint arXiv:1901.07223* (2019).

[162] Karpyshev, Pavel, Kruzhkov, Evgeny, Yudin, Evgeny, Savinykh, Alena, Potapov, Andrei, Kurenkov, Mikhail, Kolomeytsev, Anton, Kalinov, Ivan, and Tsetserukou, Dzmitry. MucaSLAM: CNN-based frame quality assessment for mobile robot with omnidirectional visual SLAM. In *International Conference on Automation Science and Engineering* (2022), pp. 368–373.

[163] Kavraki, Lydia E, Svestka, Petr, Latombe, J-C, and Overmars, Mark H. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation 12* (1996), 566–580.

[164] Kegeleirs, Miquel, Grisetti, Giorgio, and Birattari, Mauro. Swarm SLAM: Challenges and perspectives. *Frontiers in Robotics and AI 8* (2021).

[165] Keivan, Nima, and Sibley, Gabe. Asynchronous adaptive conditioning for visual–inertial SLAM. *International Journal of Robotics Research 34* (2015), 1573–1589.

[166] Kerschke, Pascal, Hoos, Holger H, Neumann, Frank, and Trautmann, Heike. Automated algorithm selection: Survey and perspectives. *Evolutionary Computation 27* (2019), 3–45.

[167] Khan, Irfan, Zhang, Xianchao, Rehman, Mobashar, and Ali, Rahman. A literature survey and empirical study of meta-learning for classifier selection. *IEEE Access 8* (2020), 10262–10281.

[168] Khotanzad, Alireza, and Hong, Yaw Hua. Invariant image recognition by Zernike moments. *IEEE Transactions on Pattern Analysis and Machine Intelligence 12* (1990), 489–497.

[169] Kim, Soohwan, Cheong, Howon, Park, Ju-Hong, and Park, Sung-Kee. Human augmented mapping for indoor environments using a stereo camera. In *International Conference on Intelligent Robots and Systems (IROS)* (2009), pp. 5609–5614.

[170] Kingma, Diederik P, and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[171] Kleiner, Alexander, Dornhege, Christian, and Dali, Sun. Mapping disaster areas jointly: RFID-coordinated SLAM by hurnans and robots. In *IEEE International Workshop on Safety, Security and Rescue Robotics* (2007), pp. 1–6.

[172] Kollar, Thomas, and Roy, Nicholas. Trajectory optimization using reinforcement learning for map exploration. *International Journal of Robotics Research 27* (2008), 175–196.

[173] Kosecká, Jana, and Li, Fayin. Vision based topological Markov localization. In *International Conference on Robotics and Automation (ICRA)* (2004), pp. 1481–1486.

[174] Kostovska, Ana, Jankovic, Anja, Vermetten, Diederick, de Nobel, Jacob, Wang, Hao, Eftimov, Tome, and Doerr, Carola. Per-run algorithm selection with warm-starting using trajectory-based features. In *International Conference on Parallel Problem Solving from Nature* (2022), pp. 46–60.

[175] Kotthoff, Lars. Algorithm selection for combinatorial search problems: A survey. *Data Mining and Constraint Programming: Foundations of a Cross-disciplinary Approach* (2016), 149–190.

[176] Krajnik, Tomas, Fentanes, Jaime Pulido, Cielniak, Grzegorz, Dondrup, Christian, and Duckett, Tom. Spectral analysis for long-term robotic mapping. In *International Conference on Robotics and Automation (ICRA)* (2014).

[177] Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems* (2012), pp. 1097–1105.

[178] Kuipers, Benjamin. Modeling spatial knowledge. *Cognitive Science 2* (1978), 129–153.

[179] Labbe, Mathieu, and Michaud, Francois. Appearance-based loop closure detection for online large-scale and long-term operation. *IEEE Transactions on Robotics 29* (2013), 734–745.

[180] Lagoudakis, Michail G, Littman, Michael L, et al. Algorithm selection using reinforcement learning. In *International Conference on Machine Learning (ICML)* (2000), pp. 511–518.

[181] Latif, Yasir, Cadena, César, and Neira, José. Robust graph slam back-ends: A comparative analysis. In *International Conference on Intelligent Robots and Systems (IROS)* (2014), pp. 2683–2690.

[182] Laugier, Christian, and Chatila, Raja. *Autonomous navigation in dynamic environments*. Springer, 2007.

[183] LeCun, Yann, Boser, Bernhard, Denker, John S, Henderson, Donnie, Howard, Richard E, Hubbard, Wayne, and Jackel, Lawrence D. Backpropagation applied to handwritten zip code recognition. *Neural Computation 1* (1989), 541–551.

[184] Lee, Changyo, Peng, Jichao, and Xiong, Zhenhua. Asynchronous fusion of visual and wheel odometer for SLAM applications. In *International Conference on Advanced Intelligent Mechatronics (AIM)* (2020), pp. 1990–1995.

[185] Lee, Gim Hee, Fraundorfer, Friedrich, and Pollefeys, Marc. Robust pose-graph loop-closures with expectation-maximization. In *International Conference on Intelligent Robots and Systems (IROS)* (2013), pp. 556–563.

[186] Lee, Kwang Wee, Wijesoma, Sardha, and Guzmán, Javier Ibanez. A constrained SLAM approach to robust and accurate localisation of autonomous ground vehicles. *Robotics and Autonomous Systems 55* (2007), 527–540.

[187] Leonardi, Marco, Fiori, Luca, and Stahl, Annette. Deep learning based keypoint rejection system for underwater visual ego-motion estimation. *International Federation of Automatic Control 53* (2020), 9471–9477.

[188] Leutenegger, Stefan, Chli, Margarita, and Siegwart, Roland Y. BRISK: Binary robust invariant scalable keypoints. In *International Conference on Computer Vision (ICCV)* (2011), pp. 2548–2555.

[189] Li, Dongjiang, Shi, Xuesong, Long, Qiwei, Liu, Shenghui, Yang, Wei, Wang, Fangshi, Wei, Qi, and Qiao, Fei. DXSLAM: A robust and efficient visual SLAM system with deep features. In *International Conference on Intelligent Robots and Systems (IROS)* (2020).

[190] Li, Hao, Chaudhari, Pratik, Yang, Hao, Lam, Michael, Ravichandran, Avinash, Bhotika, Rahul, and Soatto, Stefano. Rethinking the hyperparameters for fine-tuning. *arXiv preprint arXiv:2002.11770* (2020).

[191] Li, Lihong, Walsh, Thomas J, and Littman, Michael L. Towards a unified theory of state abstraction for MDPs. In *International Symposium on Artificial Intelligence and Mathematics* (2006).

[192] Li, Qingde, and Griffiths, JG. Iterative closest geometric objects registration. *Computers & Mathematics with Applications 40* (2000).

[193] Li, X Rong, and Zhang, Youmin. Multiple-model estimation with variable structure. v. likely-model set algorithm. *IEEE Transactions on Aerospace and Electronic Systems 36* (2000), 448–466.

[194] Li, Xiao-Rong, and Bar-Shalom, Yaakov. Multiple-model estimation with variable structure. *IEEE Transactions on Automatic Control* (1996).

[195] Li, Yali, Wang, Shengjin, Tian, Qi, and Ding, Xiaoqing. A survey of recent advances in visual feature detection. *Neurocomputing 149* (2015), 736–751.

[196] Li, Ye, Ma, Teng, Chen, Pengyun, Jiang, Yanqing, Wang, Rupeng, and Zhang, Qiang. Autonomous underwater vehicle optimal path planning method for seabed terrain matching navigation. *Ocean Engineering 133* (2017), 107–115.

[197] Li, You, and Ruichek, Yassine. Building variable resolution occupancy grid map from stereoscopic system—a quadtree based approach. In *Intelligent Vehicles Symposium (IV)* (2013), pp. 744–749.

[198] Li, Zewen, Liu, Fan, Yang, Wenjie, Peng, Shouheng, and Zhou, Jun. A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems* (2021).

[199] Lim, Joseph J, Zitnick, C Lawrence, and Dollár, Piotr. Sketch tokens: A learned mid-level representation for contour and object detection. In *Conference on Computer Vision and Pattern Recognition (CVPR)* (2013).

[200] Lim, Seungwook, Lee, Tae-kyeong, Lee, Seongsoo, An, Shounan, and Oh, Se-young. Adaptive sliding window for hierarchical pose-graph-based SLAM. In *International Conference on Control, Automation and Systems* (2012), pp. 2153–2158.

[201] Littman, Michael Lederman. *Algorithms for sequential decision-making*. Brown University, 1996.

[202] Liu, Haomin, Chen, Mingyu, Zhang, Guofeng, Bao, Hujun, and Bao, Yingze. ICE-BA: Incremental, consistent and efficient bundle adjustment for visual-inertial slam. In *Conference on Computer Vision and Pattern Recognition (CVPR)* (2018), pp. 1974–1982.

[203] Liu, Huaran, Liu, Zhengyu, and Lu, Feiyu. A systematic comparison of particle filter and EnKF in assimilating time-averaged observations. *Journal of Geophysical Research: Atmospheres 122* (2017), 13–155.

[204] Liu, Yang, and Zhang, Hong. Indexing visual features: Real-time loop closure detection using a tree structure. In *International Conference on Robotics and Automation (ICRA)* (2012), pp. 3613–3618.

[205] Lowe, David G. Object recognition from local scale-invariant features. In *International Conference on Computer Vision (ICCV)* (1999), pp. 1150–1157.

[206] Lu, Feng, and Milios, Evangelos. Globally consistent range scan alignment for environment mapping. *Autonomous Robots 4* (1997), 333–349.

[207] Lukac, Martin, and Kameyama, Michitaka. Adaptive functional module selection using machine learning: Framework for intelligent robotics. In *SICE Annual Conference* (2011), pp. 2480–2483.

[208] Lukac, Martin, Zhurtanov, Almas, and Ospanova, Aizhan. High-level verification of multi-object segmentation. In *International Conference on Information and Digital Technologies (IDT)* (2016), pp. 173–179.

[209] Luperto, Matteo, Castelli, Valerio, and Amigoni, Francesco. Predicting performance of SLAM algorithms. *arXiv preprint arXiv:2109.02329* (2021).

[210] Lupton, Todd, and Sukkarieh, Salah. Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions. *IEEE Transactions on Robotics 28* (2011), 61–76.

[211] Madani, Omid, Hanks, Steve, and Condon, Anne. On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems. In *Innovative Applications of Artificial Intelligence Conference* (1999), pp. 541–548.

[212] Magill, David Thomas. Optimal adaptive estimation of sampled stochastic processes. *IEEE Transactions on Automatic Control 10* (1965), 434–439.

[213] Mahendran, Siddharth, Ali, Haider, and Vidal, René. 3D pose regression using convolutional neural networks. In *International Conference on Computer Vision Workshops* (2017), pp. 2174–2182.

[214] Mahmud, Saaduddin, Nashed, Samer B, Goldman, Claudia V, and Zilberstein, Shlomo. Estimating causal responsibility for explaining autonomous behavior. In *International Workshop on Explainable, Transparent Autonomous Agents and Multi-Agent Systems* (2023), pp. 78–94.

[215] Mairal, Julien, Leordeanu, Marius, Bach, Francis, Hebert, Martial, and Ponce, Jean. Discriminative sparse image models for class-specific edge detection and image interpretation. In *European Conference on Computer Vision (ECCV)* (2008).

[216] Makarenko, Alexei A, Williams, Stefan B, Bourgault, Frederic, and Durrant-Whyte, Hugh F. An experiment in integrated exploration. In *International Conference on Intelligent Robots and Systems (IROS)* (2002).

[217] Malek, Alan, Abbasi-Yadkori, Yasin, and Bartlett, Peter. Linear programming for large-scale Markov decision problems. In *International Conference on Machine Learning (ICML)* (2014), pp. 496–504.

[218] Manne, Alan S. Linear programming and sequential decisions. *Management Science* (1960).

[219] Maragliano, Matteo, Ahmed, Muhammad Farhan, Recchiuto, Carmine Tommaso, Sgorbissa, Antonio, and Fremont, Vincent. Collaborative active SLAM: Synchronous and asynchronous coordination among agents. *arXiv preprint arXiv:2310.01967* (2023).

[220] Martin, David R, Fowlkes, Charless C, and Malik, Jitendra. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence 26* (2004), 530–549.

[221] Martinez-Cantin, Ruben, De Freitas, Nando, Brochu, Eric, Castellanos, José, and Doucet, Arnaud. A Bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot. *Autonomous Robots 27* (2009), 93–103.

[222] Maybeck, Peter S. *Stochastic models, estimation, and control.* Academic press, 1982.

[223] Maybeck, Peter S. Moving-bank multiple model adaptive estimation and control algorithms: An evaluation. *Control and Dynamic Systems: Advances in Theory and Applications* (2012), 1–31.

[224] McCormac, John, Clark, Ronald, Bloesch, Michael, Davison, Andrew, and Leutenegger, Stefan. Fusion++: Volumetric object-level SLAM. In *International Conference on 3D Vision* (2018), pp. 32–41.

[225] Mehra, Raman. On the identification of variances and adaptive Kalman filtering. *IEEE Transactions on automatic control 15* (1970), 175–184.

[226] Mehra, Raman K. On-line identification of linear dynamic systems with applications to Kalman filtering. *IEEE Transactions on Automatic Control 16* (1971), 12–21.

[227] Meng, Jie, Wang, Shuting, Xie, Yuanlong, Jiang, Liquan, Li, Gen, and Liu, Chao. Efficient re-localization of mobile robot using strategy of finding a missing person. *Measurement 176* (2021).

[228] Merali, Rehman S, and Barfoot, Timothy D. Optimizing online occupancy grid mapping to capture the residual uncertainty. In *International Conference on Robotics and Automation (ICRA)* (2014), pp. 6070–6076.

[229] Meunier, Laurent, Rakotoarison, Herilalaina, Wong, Pak Kan, Roziere, Baptiste, Rapin, Jeremy, Teytaud, Olivier, Moreau, Antoine, and Doerr, Carola. Black-box optimization revisited: Improving algorithm selection wizards through massive benchmarking. *IEEE Transactions on Evolutionary Computation 26* (2021), 490–500.

[230] Michael, Nathan, Zavlanos, Michael M, Kumar, Vijay, and Pappas, George J. Distributed multi-robot task assignment and formation control. In *International Conference on Robotics and Automation (ICRA)* (2008), pp. 128–133.

[231] Milella, Annalisa, Dimiccoli, Carmine, Cicirelli, Grazia, and Distante, Arcangelo. Laser-based people-following for human-augmented mapping of indoor environments. In *Artificial Intelligence and Applications* (2007), pp. 169–175.

[232] Miró, Jaime Valls, Dissanayake, Gamini, and Zhou, Weizhen. Vision-based SLAM using natural features in indoor environments. In *International Conference on Intelligent Sensors, Sensor Networks and Information Processing* (2005), pp. 151–156.

[233] Mitlin, Anatoly, and Nashed, Samer. Autonomous machine motion planning in a dynamic environment, Nov. 15 2022. US Patent 11,498,587.

[234] Mohamed, AH, and Schwarz, KP. Adaptive Kalman filtering for INS/GPS. *Journal of Geodesy 73* (1999), 193–203.

[235] Mokhtarian, Farzin, and Mohanna, Farahnaz. Performance evaluation of corner detectors using consistency and accuracy measures. *Computer Vision and Image Understanding 102* (2006), 81–94.

[236] Moll, Mark, Chamzas, Constantinos, Kingston, Zachary, and Kavraki, Lydia E. Hyperplan: A framework for motion planning algorithm selection and parameter optimization. In *International Conference on Intelligent Robots and Systems (IROS)* (2021), pp. 2511–2518.

[237] Montemerlo, Michael, Thrun, Sebastian, Koller, Daphne, Wegbreit, Ben, et al. Fast-SLAM: A factored solution to the simultaneous localization and mapping problem. In *Innovative Applications of Artificial Intelligence Conference* (2002), pp. 593–598.

[238] Moras, Julien, Cherfaoui, Véronique, and Bonnifait, Philippe. Credibilist occupancy grids for vehicle perception in dynamic environments. In *International Conference on Robotics and Automation (ICRA)* (2011), pp. 84–89.

[239] Moravec, Hans P., and Cho, Dong Woo. A Bayesian method for certainty grids. In *AAAI Spring Symposium on Robot Navigation* (1989).

[240] Mourikis, Anastasios I, and Roumeliotis, Stergios I. Predicting the performance of cooperative simultaneous localization and mapping (C-SLAM). *International Journal of Robotics Research 25* (2006), 1273–1286.

[241] Müller, David, Müller, Marcus G, Kress, Dominik, and Pesch, Erwin. An algorithm selection approach for the flexible job shop scheduling problem: Choosing constraint programming solvers through machine learning. *European Journal of Operational Research 302* (2022), 874–891.

[242] Mur-Artal, Raul, Montiel, Jose Maria Martinez, and Tardos, Juan D. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics 31* (2015), 1147–1163.

[243] Mur-Artal, Raul, and Tardós, Juan D. ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Transactions on Robotics 33* (2017), 1255–1262.

[244] Murphy, Kevin P. Dynamic Bayesian networks. *Probabilistic Graphical Models 7* (2002).

[245] Murphy, Robin R. Human-robot interaction in rescue robotics. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 34* (2004), 138–153.

[246] Nakamura, Mason, Svegliato, Justin, Nashed, Samer B, Zilberstein, Shlomo, and Russell, Stuart. Formal composition of robotic systems as contract programs. In *International Conference on Intelligent Robots and systems (IROS)* (2023).

[247] Naseer, Tayyab, Ruhnke, Michael, Stachniss, Cyrill, Spinello, Luciano, and Burgard, Wolfram. Robust visual SLAM across seasons. In *International Conference on Intelligent Robots and Systems (IROS)* (2015).

[248] Nashed, Samer, and Biswas, Joydeep. Curating long-term vector maps. In *International Conference on Intelligent Robots and Systems (IROS)* (2016), pp. 4643–4648.

[249] Nashed, Samer, and Ilstrup, David. Localization determination for vehicle operation, Sept. 7 2021. US Patent 11,112,259.

[250] Nashed, Samer, Park, Jong Jin, and Durham, Joseph. Physical models for hierarchical clustering and symbolic inference, Jan. 2 2024. US Patent 11,860,278.

[251] Nashed, Samer, Svegliato, Justin, and Zilberstein, Shlomo. Ethically compliant planning within moral communities. In *AAAI/ACM Conference on AI, Ethics, and Society* (2021), pp. 188–198.

[252] Nashed, Samer, and Zilberstein, Shlomo. A survey of opponent modeling in adversarial domains. *Journal of Artificial Intelligence Research 73* (2022), 277–327.

[253] Nashed, Samer B. A brief survey of loop closure detection: A case for rethinking evaluation of intelligent systems. In *NeurIPS 2020 Workshop: ML Retrospectives, Surveys Meta-Analyses (ML-RSA)* (2020).

[254] Nashed, Samer B. Laser2Vec: Similarity-based retrieval for robotic perception data. In *International Conference on Intelligent Robots and Systems (IROS)* (2020), pp. 10657–10662.

[255] Nashed, Samer B, and Biswas, Joydeep. Human-in-the-loop SLAM. In *AAAI Conference on Artificial Intelligence* (2018).

[256] Nashed, Samer B, Ilstrup, David M, and Biswas, Joydeep. Localization under topological uncertainty for lane identification of autonomous vehicles. In *International Conference on Robotics and Automation (ICRA)* (2018), pp. 6000–6005.

[257] Nashed, Samer B, Mahmud, Saaduddin, Goldman, Claudia V, and Zilberstein, Shlomo. Causal explanations for sequential decision making under uncertainty. In *International Conference on Autonomous Agents and Multiagent Systems* (2023), pp. 2307–2309.

[258] Nashed, Samer B, Park, Jong Jin, Webster, Roger, and Durham, Joseph W. Robust rank deficient SLAM. In *International Conference on Intelligent Robots and Systems (IROS)* (2021).

[259] Nashed, Samer B, Svegliato, Justin, Bhatia, Abhinav, Russell, Stuart, and Zilberstein, Shlomo. Selecting the partial state abstractions of mdps: A metareasoning approach with deep reinforcement learning. In *International Conference on Intelligent Robots and Systems (IROS)* (2022).

[260] Nashed, Samer B, Svegliato, Justin, and Blodgett, Su Lin. Fairness and sequential decision making: Limits, lessons, and opportunities. *arXiv preprint arXiv:2301.05753* (2023).

[261] Nashed, Samer B, Svegliato, Justin, Brucato, Matteo, Basich, Connor, Grupen, Rod, and Zilberstein, Shlomo. Solving Markov decision processes with partial state abstractions. In *International Conference on Robotics and Automation (ICRA)* (2021), pp. 813–819.

[262] Nguyen, Thien-Nghia, Michaelis, Bernd, Al-Hamadi, Ayoub, Tornow, Michael, and Meinecke, Marc-Michael. Stereo-camera-based urban environment perception using occupancy grid and object tracking. *IEEE Transactions on Intelligent Transportation Systems 13* (2011), 154–165.

[263] Nguyen, Viet, Harati, Ahad, and Siegwart, Roland. A lightweight SLAM algorithm using orthogonal planes for indoor mobile robotics. In *International Conference on Intelligent Robots and Systems (IROS)* (2007), pp. 658–663.

[264] Nguyen, Viet, Martinelli, Agostino, Tomatis, Nicola, and Siegwart, Roland. A comparison of line extraction algorithms using 2D laser rangefinder for indoor mobile robotics. In *International Conference on Intelligent Robots and Systems (IROS)* (2005).

[265] Ni, Kai, Steedly, Drew, and Dellaert, Frank. Tectonic SAM: Exact, out-of-core, submap-based SLAM. In *International Conference on Robotics and Automation (ICRA* (2007), pp. 1678–1685.

[266] Nieto-Granda, Carlos, Rogers, John G, Trevor, Alexander JB, and Christensen, Henrik I. Semantic map partitioning in indoor environments using regional analysis. In *International Conference on Intelligent Robots and Systems (IROS)* (2010), pp. 1451–1456.

[267] Nong, Xiaoqi, and Hadfield, Simon. ASL-SLAM: An asynchronous formulation of lines for SLAM with event sensors. In *International Conference on Industrial Engineering and Applications* (2022), pp. 84–91.

[268] Nourbakhsh, Illah R, Sycara, Katia, Koes, Mary, Yong, Mark, Lewis, Michael, and Burion, Steve. Human-robot teaming for search and rescue. *IEEE Pervasive Computing 4* (2005), 72–79.

[269] Olson, Edwin, Strom, Johannes, Goeddel, Rob, Morton, Ryan, Ranganathan, Pradeep, and Richardson, Andrew. Exploration and mapping with autonomous robot teams. *Communications of the ACM 56* (2013), 62–70.

[270] Olson, Edwin B. Real-time correlative scan matching. In *International Conference on Robotics and Automation (ICRA)* (2009), pp. 4387–4393.

[271] Ong, Sylvie CW, Png, Shao Wei, Hsu, David, and Lee, Wee Sun. Planning under uncertainty for robotic tasks with mixed observability. *International Journal of Robotics Research 29* (2010), 1053–1068.

[272] Ono, Yuki, Trulls, Eduard, Fua, Pascal, and Yi, Kwang Moo. LF-Net: Learning local features from images. In *Advances in Neural Information Processing Systems* (2018).

[273] Ossenkopf, Marie, Castro, Gastón, Pessacg, Facundo, Geihs, Kurt, and De Cristóforis, Pablo. Long-horizon active SLAM system for multi-agent coordinated exploration. In *European Conference on Mobile Robots* (2019), pp. 1–6.

[274] Parasuraman, Raja, Barnes, Michael, Cosenzo, Keryl, and Mulgund, Sandeep. Adaptive automation for human-robot teaming in future command and control systems. Tech. rep., DTIC Document, 2007.

[275] Pasetto, Damiano, Camporese, Matteo, and Putti, Mario. Ensemble Kalman filter versus particle filter for a physically-based coupled surface–subsurface model. *Advances in Water Resources 47* (2012), 1–13.

[276] Paz, Lina María, Jensfelt, Patric, Tardos, Juan D, and Neira, José. EKF SLAM updates in O(n) with divide and conquer SLAM. In *International Conference on Robotics and Automation (ICRA)* (2007), pp. 1657–1663.

[277] Pearce, Tim, Brintrup, Alexandra, and Zhu, Jun. Understanding softmax confidence and uncertainty. *arXiv preprint arXiv:2106.04972* (2021).

[278] Pedraza, Luis, Rodriguez-Losada, Diego, Matia, Fernando, Dissanayake, Gamini, and Miró, Jaime Valls. Extending the limits of feature-based SLAM with B-splines. *IEEE Transactions on Robotics 25* (2009), 353–366.

[279] Penney, Graeme P, Edwards, Philip J, King, Andrew P, Blackall, Jane M, Batchelor, Philipp G, and Hawkes, David J. A stochastic iterative closest point algorithm (stochasticp). In *International Conference on Medical Image Computing and Computer-Assisted Intervention* (2001), pp. 762–769.

[280] Petrik, Marek, and Zilberstein, Shlomo. Constraint relaxation in approximate linear programs. In *International Conference on Machine Learning (ICML)* (2009), pp. 809–816.

[281] Pfingsthorn, Max, and Birk, Andreas. Generalized graph SLAM: Solving local and global ambiguities through multimodal and hyperedge constraints. *International Journal of Robotics Research 35* (2016), 601–630.

[282] Pfingsthorn, Max, Birk, Andreas, and Bülow, Heiko. An efficient strategy for data exchange in multi-robot mapping under underwater communication constraints. In *International Conference on Intelligent Robots and Systems (IROS)* (2010), pp. 4886–4893.

[283] Pfister, Samuel T., Roumeliotis, Stergios I., and Burdick, Joel W. Weighted line fitting algorithms for mobile robot map building and efficient data representation. In *International Conference on Robotics and Automation (ICRA)* (2003).

[284] Piazza, Enrico, Lima, Pedro U, and Matteucci, Matteo. Performance models in robotics with a use case on SLAM. *Robotics and Automation Letters 7* (2022), 4646–4653.

[285] Pineau, Joelle, Gordon, Geoff, Thrun, Sebastian, et al. Point-based value iteration: An anytime algorithm for POMDPs. In *International Joint Conference on Artificial Intelligence* (2003), pp. 1025–1032.

[286] Pineda, Luis Enrique, Wray, Kyle Hollins, and Zilberstein, Shlomo. Fast SSP solvers using short-sighted labeling. In *AAAI Conference on Artificial Intelligence* (2017).

[287] Placed, Julio A, Strader, Jared, Carrillo, Henry, Atanasov, Nikolay, Indelman, Vadim, Carlone, Luca, and Castellanos, José A. A survey on active simultaneous localization and mapping: State of the art and new frontiers. *IEEE Transactions on Robotics* (2023).

[288] Pomerleau, François, Colas, Francis, Siegwart, Roland, and Magnenat, Stéphane. Comparing icp variants on real-world data sets. *Autonomous Robots 34* (2013).

[289] Poupart, Pascal, and Boutilier, Craig. Bounded finite state controllers. *Advances in Neural Information Processing Systems 16* (2003).

[290] Poupart, Pascal, Malhotra, Aarti, Pei, Pei, Kim, Kee-Eung, Goh, Bongseok, and Bowling, Michael. Approximate linear programming for constrained partially observable Markov decision processes. In *AAAI Conference on Artificial Intelligence* (2015), pp. 3342–3348.

[291] Powell, Warren B. Perspectives of approximate dynamic programming. *Annals of Operations Research 241* (2016), 319–356.

[292] Prentice, Sam, and Roy, Nicholas. The belief roadmap: Efficient planning in linear POMDPs by factoring the covariance. In *International Symposium on Robotics Research* (2011), pp. 293–305.

[293] Procopiuc, Octavian, Agarwal, Pankaj K, Arge, Lars, and Vitter, Jeffrey Scott. Bkd-tree: A dynamic scalable kd-tree. In *International Symposium on Advances in Spatial and Temporal Databases* (2003), pp. 46–65.

[294] Proença, Pedro F, and Gao, Yang. Fast cylinder and plane extraction from depth cameras for visual odometry. In *International Conference on Intelligent Robots and Systems (IROS)* (2018).

[295] Pronobis, Andrzej, and Jensfelt, Patric. Large-scale semantic mapping and reasoning with heterogeneous modalities. In *International Conference on Robotics and Automation (ICRA)* (2012), pp. 3515–3522.

[296] Pulina, Luca, and Tacchella, Armando. A self-adaptive multi-engine solver for quantified boolean formulas. *Constraints 14* (2009), 80–116.

[297] Rabiee, Sadegh, and Biswas, Joydeep. IV-SLAM: Introspective vision for simultaneous localization and mapping. In *Conference on Robot Learning* (2021), pp. 1100–1109.

[298] Rahman, Quazi Marufur, Corke, Peter, and Dayoub, Feras. Run-time monitoring of machine learning for robotic perception: A survey of emerging trends. *IEEE Access 9* (2021), 20067–20075.

[299] Ranganathan, Ananth, Ilstrup, David, and Wu, Tao. Light-weight localization for vehicles using road markings. In *International Conference on Intelligent Robots and Systems (IROS)* (2013).

[300] Ranganathan, Ananth, Matsumoto, Shohei, and Ilstrup, David. Towards illumination invariance for visual localization. In *International Conference on Robotics and Automation (ICRA)* (2013).

[301] Rao, C. Radhakrishna. Information and accuracy attainable in estimation of statistical parameters. *Bulletin of the Calcutta Mathematical Society* (1945).

[302] Rao, Yuan, Ni, Jiangqun, and Zhao, Huimin. Deep learning local descriptor for image splicing detection and localization. *IEEE Access 8* (2020), 25611–25625.

[303] Ravindran, Balaraman, and Barto, Andrew G. Model minimization in hierarchical reinforcement learning. In *International Symposium on Abstraction, Reformulation, and Approximation* (2002), pp. 196–211.

[304] Regan, Kevin, and Boutilier, Craig. Robust policy computation in reward-uncertain mdps using nondominated policies. In *AAAI Conference on Artificial Intelligence* (2010), pp. 1127–1133.

[305] Ren, Wei, and Sorensen, Nathan. Distributed coordination architecture for multi-robot formation control. *Robotics and Autonomous Systems 56* (2008), 324–333.

[306] Ren, Xiaofeng. Multi-scale improves boundary detection in natural images. In *European Conference on Computer Vision (ECCV)* (2008).

[307] Rice, John R. The algorithm selection problem. In *Advances in Computers*, vol. 15. 1976, pp. 65–118.

[308] Romero-Ramirez, Francisco J, Muñoz-Salinas, Rafael, Marín-Jiménez, Manuel J, Carmona-Poyato, Angel, and Medina-Carnicer, Rafael. ReSLAM: Reusable SLAM with heterogeneous cameras. *Neurocomputing* (2023), 126940.

[309] Rosenbluth, Marshall N, and Rosenbluth, Arianna W. Monte Carlo calculation of the average extension of molecular chains. *Journal of Chemical Physics 23* (1955), 356–359.

[310] Rosenfeld, Avi, Kaminka, Gal A, Kraus, Sarit, and Shehory, Onn. A study of mechanisms for improving robotic group performance. *Artificial Intelligence 172* (2008), 633–655.

[311] Rublee, Ethan, Rabaud, Vincent, Konolige, Kurt, and Bradski, Gary. ORB: An efficient alternative to SIFT or SURF. In *International Conference on Computer Vision (ICCV)* (2011), pp. 2564–2571.

[312] Rubner, Yossi, Tomasi, Carlo, and Guibas, Leonidas J. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision 40* (2000), 99–121.

[313] Ruiken, Dirk, Liu, Tiffany Q, Takahashi, Takeshi, and Grupen, Roderic A. Reconfigurable tasks in belief-space planning. In *International Conference on Humanoid Robots* (2016), pp. 1257–1263.

[314] Rumelhart, David E, Hinton, Geoffrey E, Williams, Ronald J, et al. Learning internal representations by error propagation, 1985.

[315] Russakovsky, Olga, Deng, Jia, Su, Hao, Krause, Jonathan, Satheesh, Sanjeev, Ma, Sean, Huang, Zhiheng, Karpathy, Andrej, Khosla, Aditya, Bernstein, Michael, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision 115* (2015), 211–252.

[316] Saarinen, Jari, Andreasson, Henrik, and Lilienthal, Achim J. Independent Markov chain occupancy grid maps for representation of dynamic environment. In *International Conference on Intelligent Robots and Systems (IROS)* (2012).

[317] Sarlin, Paul-Edouard, Cadena, Cesar, Siegwart, Roland, and Dymczyk, Marcin. From coarse to fine: Robust hierarchical localization at large scale. In *Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).

[318] Scheer, Jonas, Fritz, Mario, and Grau, Oliver. Learning to select long-track features for structure-from-motion and visual SLAM. In *German Conference on Pattern Recognition* (2016).

[319] Schindler, Andreas. Vehicle self-localization with high-precision digital maps. In *Intelligent Vehicles Symposium (IV)* (2013).

[320] Schirmer, Pascal A, and Mporas, Iosif. Double fourier integral analysis based convolutional neural network regression for high-frequency energy disaggregation. *IEEE Transactions on Emerging Topics in Computational Intelligence 6* (2021), 439–449.

[321] Schreiber, Markus, Knöppel, Carsten, and Franke, Uwe. Laneloc: Lane marking based localization using highly accurate maps. In *Intelligent Vehicles Symposium (IV)* (2013), pp. 449–454.

[322] Schuster, Martin J, Schmid, Korbinian, Brand, Christoph, and Beetz, Michael. Distributed stereo vision-based 6D localization and mapping for multi-robot teams. *Journal of Field Robotics 36* (2019), 305–332.

[323] Segal, Aleksandr, Haehnel, Dirk, and Thrun, Sebastian. Generalized-ICP. In *Robotics: Science and Systems (RSS)* (2009), vol. 2, p. 435.

[324] Shi, Wanying, and Guo, Jian. Application of Markov decision processes (MDPs) in petroleum industry. *Journal of Engineering Technology 2* (2014).

[325] Shi, Yahao, Cao, Xinyu, Lu, Feixiang, and Zhou, Bin. P^3-net: Part mobility parsing from point cloud sequences via learning explicit point correspondence. In *AAAI Conference on Artificial Intelligence* (2022), vol. 36, pp. 2244–2252.

[326] Shin, Dong-Won, and Ho, Yo-Sung. Local patch descriptor using deep convolutional generative adversarial network for loop closure detection in SLAM. In *Asia-Pacific Signal and Information Processing Association Annual Summit and Conference* (2017), pp. 546–549.

[327] Shin, Yujin, and Kim, Euiho. Hybrid path planning using positioning risk and artificial potential fields. *Aerospace Science and Technology 112* (2021).

[328] Sibley, Gabe, Matthies, Larry, and Sukhatme, Gaurav. Sliding window filter with application to planetary landing. *Journal of Field Robotics 27* (2010), 587–608.

[329] Sim, Robert, and Roy, Nicholas. Global a-optimal robot exploration in SLAM. In *International Conference on Robotics and Automation (ICRA)* (2005), pp. 661–666.

[330] Simo-Serra, Edgar, Trulls, Eduard, Ferraz, Luis, Kokkinos, Iasonas, Fua, Pascal, and Moreno-Noguer, Francesc. Discriminative learning of deep convolutional feature point descriptors. In *International Conference on Computer Vision (ICCV)* (2015).

[331] Simonyan, Karen, and Zisserman, Andrew. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)* (2015).

[332] Sivaraman, Sayanan, and Trivedi, Mohan Manubhai. Integrated lane and vehicle detection, localization, and tracking: A synergistic approach. *IEEE Transactions on Intelligent Transportation Systems 14* (2013), 906–917.

[333] Smith, Trey, and Simmons, Reid. Focused real-time dynamic programming for MDPs: Squeezing more out of a heuristic. In *AAAI Conference on Artificial Intelligence* (2006), pp. 1227–1232.

[334] Smith-Miles, Kate A. Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys 41* (2009), 1–25.

[335] Sohn, Hee Jin, and Kim, Byung Kook. VecSLAM: An efficient vector-based SLAM algorithm for indoor environments. *Journal of Intelligent and Robotic Systems 56* (2009), 301–318.

[336] Srinivasan, Krishnaswamy. State estimation by orthogonal expansion of probability distributions. *Transactions on Automatic Control 15* (1970), 3–10.

[337] Stachniss, Cyrill, Grisetti, Giorgio, and Burgard, Wolfram. Information gain-based exploration using Rao-Blackwellized particle filters. In *Robotics: Science and Systems (RSS)* (2005), pp. 65–72.

[338] Stentz, Anthony. The D* algorithm for real-time planning of optimal traverses. Tech. rep., Carnegie-Mellon University Robotics Institute, 1994.

[339] Strasdat, Hauke, Stachniss, Cyrill, and Burgard, Wolfram. Which landmark is useful? Learning selection policies for navigation in unknown environments. In *International Conference on Robotics and Automation (ICRA)* (2009).

[340] Sunderhauf, Niko, Lange, Sven, and Protzel, Peter. Using the unscented Kalman filter in mono-SLAM with inverse depth parametrization for autonomous airship control. In *International Workshop on Safety, Security and Rescue Robotics* (2007).

[341] Sünderhauf, Niko, Lange, Sven, and Protzel, Peter. Incremental sensor fusion in factor graphs with unknown delays. *Advanced Space Technologies in Robotics and Automation* (2013).

[342] Sünderhauf, Niko, and Protzel, Peter. Switchable constraints for robust pose graph SLAM. In *International Conference on Intelligent Robots and Systems (IROS)* (2012), pp. 1879–1884.

[343] Sünderhauf, Niko, and Protzel, Peter. Towards a robust back-end for pose graph slam. In *International Conference on Robotics and Automation (ICRA)* (2012), pp. 1254–1261.

[344] Sünderhauf, Niko, Shirazi, Sareh, Dayoub, Feras, Upcroft, Ben, and Milford, Michael. On the performance of convnet features for place recognition. In *International Conference on Intelligent Robots and Systems (IROS)* (2015).

[345] Sünderhauf, Niko, Shirazi, Sareh, Jacobson, Adam, Dayoub, Feras, Pepperell, Edward, Upcroft, Ben, and Milford, Michael. Place recognition with convnet landmarks: Viewpoint-robust, condition-robust, training-free. In *Robotics: Science and Systems* (2015).

[346] Svegliato, Justin, Nashed, Samer, and Zilberstein, Shlomo. An integrated approach to moral autonomous systems. In *European Conference on Artificial Intelligence* (2020), pp. 2941–2942.

[347] Svegliato, Justin, Nashed, Samer B, and Zilberstein, Shlomo. Ethically compliant planning in moral autonomous systems. In *International Joint Conference on Artificial Intelligence Workshop on AI Safety* (2020).

[348] Svegliato, Justin, Nashed, Samer B, and Zilberstein, Shlomo. Ethically compliant sequential decision making. In *AAAI Conference on Artificial Intelligence* (2021), pp. 11657–11665.

[349] Szegedy, Christian, Vanhoucke, Vincent, Ioffe, Sergey, Shlens, Jon, and Wojna, Zbigniew. Rethinking the inception architecture for computer vision. In *Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 2818–2826.

[350] Tang, Jiexiong, Kim, Hanme, Guizilini, Vitor, Pillai, Sudeep, and Ambrus, Rares. Neural outlier rejection for self-supervised keypoint learning. *arXiv preprint arXiv:1912.10615* (2019).

[351] Tao, Zui, Bonnifait, Ph, Fremont, Vincent, and Ibanez-Guzman, Javier. Mapping and localization using gps, lane markings and proprioceptive sensors. In *International Conference on Intelligent Robots and Systems (IROS)* (2013).

[352] Tapus, Adriana, Tomatis, Nicola, and Siegwart, Roland. Topological global localization and mapping with fingerprints and uncertainty. *Experimental Robotics IX* (2006), 99–111.

[353] Tasche, Philip, and Herber, Paula. A coverage-driven systematic test approach for simultaneous localization and mapping. In *Conference on Software Testing, Verification and Validation* (2023), pp. 25–36.

[354] Taylor, Zachary, and Nieto, Juan. Motion-based calibration of multimodal sensor extrinsics and timing offset estimation. *IEEE Transactions on Robotics 32* (2016), 1215–1229.

[355] Teng, Ma, Ye, Li, Yuxin, Zhao, Yanqing, Jiang, Zheng, Cong, Qiang, Zhang, and Shuo, Xu. An AUV localization and path planning algorithm for terrain-aided navigation. *ISA Transactions 103* (2020), 215–227.

[356] Thrun, Sebastian. Learning occupancy grid maps with forward sensor models. *Autonomous Robots 15* (2003), 111–127.

[357] Thrun, Sebastian, Burgard, Wolfram, and Fox, Dieter. A probabilistic approach to concurrent mapping and localization for mobile robots. *Autonomous Robots 5* (1998), 253–271.

[358] Thrun, Sebastian, and Montemerlo, Michael. The graph SLAM algorithm with applications to large-scale mapping of urban structures. *International Journal of Robotics Research 25* (2006), 403–429.

[359] Tian, Yulun, Chang, Yun, Arias, Fernando Herrera, Nieto-Granda, Carlos, How, Jonathan P, and Carlone, Luca. Kimera-Multi: Robust, distributed, dense metric-semantic SLAM for multi-robot systems. *IEEE Transactions on Robotics 38* (2022).

[360] Tian, Yurun, Fan, Bin, and Wu, Fuchao. L2-Net: Deep learning of discriminative patch descriptor in Euclidean space. In *Conference on Computer Vision and Pattern Recognition (CVPR)* (2017).

[361] Tipaldi, Gian Diego, and Arras, Kai O. FLIRT-interest regions for 2D range data. In *International Conference on Robotics and Automation (ICRA)* (2010), pp. 3616–3622.

[362] Tipaldi, Gian Diego, Braun, Manuel, and Arras, Kai O. FLIRT: Interest regions for 2D range data with applications to robot navigation. In *International Symposium on Experimental Robotics* (2014), pp. 695–710.

[363] Tipaldi, Gian Diego, Meyer-Delius, Daniel, and Burgard, Wolfram. Lifelong localization in changing environments. *International Journal of Robotics Research* (2013).

[364] Tokdar, Surya T, and Kass, Robert E. Importance sampling: A review. *Wiley Interdisciplinary Reviews: Computational Statistics 2* (2010), 54–60.

[365] Topp, Elin A, and Christensen, Henrik I. Topological modelling for human augmented mapping. In *International Conference on Intelligent Robots and Systems (IROS)* (2006), pp. 2257–2263.

[366] Topp, Elin A, and Christensen, Henrik I. Detecting region transitions for human-augmented mapping. *IEEE Transactions on Robotics 26* (2010), 715–720.

[367] Topp, Elin A, Huettenrauch, Helge, Christensen, Henrik I, and Eklundh, Kerstin Severinson. Bringing together human and robotic environment representations-a pilot study. In *International Conference on Intelligent Robots and Systems (IROS)* (2006), pp. 4946–4952.

[368] Trevor, Alexander JB, Rogers, John G, and Christensen, Henrik I. Omnimapper: A modular multimodal mapping framework. In *International Conference on Robotics and Automation (ICRA)* (2014), pp. 1983–1990.

[369] Trybała, Paweł. LiDAR-based simultaneous localization and mapping in an underground mine in Złoty Stok, Poland. In *IOP Conference Series. Earth and Environmental Science* (2021), vol. 942.

[370] Ullah, Inam, Su, Xin, Zhang, Xuewu, and Choi, Dongmin. Simultaneous localization and mapping based on Kalman filter and extended Kalman filter. *Wireless Communications and Mobile Computing 2020* (2020), 1–12.

[371] Ulrich, Iwan, and Nourbakhsh, Illah. Appearance-based place recognition for topological localization. In *International Conference on Robotics and Automation (ICRA)* (2000).

[372] Vanschoren, Joaquin. Meta-learning: A survey. *arXiv preprint arXiv:1810.03548* (2018).

[373] Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N, Kaiser, Łukasz, and Polosukhin, Illia. Attention is all you need. In *Advances in Neural Information Processing Systems* (2017).

[374] Vedder, Kyle, Schneeweiss, Edward, Rabiee, Sadegh, Nashed, Samer, Lane, Spencer, Holtz, Jarrett, Biswas, Joydeep, and Balaban, David. UMass MinuteBots 2017 team description paper, 2017.

[375] Vidal-Calleja, Teresa A, Berger, Cyrille, and Lacroix, Simon. Event-driven loop closure in multi-robot mapping. In *International Conference on Intelligent Robots and Systems (IROS)* (2009), pp. 1535–1540.

[376] Vilalta, Ricardo, and Drissi, Youssef. A perspective view and survey of meta-learning. *Artificial Intelligence Review 18* (2002), 77–95.

[377] Visser, Arnoud, and Slamet, Bayu A. Including communication success in the estimation of information gain for multi-robot exploration. In *International Symposium on Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks* (2008), pp. 680–687.

[378] Waibel, Alexander, Hanazawa, Toshiyuki, Hinton, Geoffrey, Shikano, Kiyohiro, and Lang, Kevin J. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing 31* (1989).

[379] Walcott-Bryant, A., Kaess, M., Johannsson, H., and Leonard, J. J. Dynamic pose graph SLAM: Long-term mapping in low dynamic environments. In *International Conference on Intelligent Robots and Systems (IROS)* (2012).

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

[381] Wang, Chao-Lei, Wang, Tian-Miao, Liang, Jian-Hong, Zhang, Yi-Cheng, and Zhou, Yi. Bearing-only visual SLAM for small unmanned aerial vehicles in GPS-denied environments. *International Journal of Automation and Computing 10* (2013), 387–396.

[382] Weigl, Martyna, Siemiaatkowska, B, Sikorski, Krzysztof A, and Borkowski, Andrzej. Grid-based mapping for autonomous mobile robot. *Robotics and Autonomous Systems 11* (1993), 13–21.

[383] West, Michael E, and Syrmos, Vassilis L. Navigation of an autonomous underwater vehicle (AUV) using robust SLAM. In *IEEE International Symposium on Intelligent Control* (2006).

[384] Westman, Eric, Hinduja, Akshay, and Kaess, Michael. Feature-based SLAM for imaging sonar with under-constrained landmarks. In *International Conference on Robotics and Automation (ICRA)* (2018).

[385] Wirtz, Stefan, and Paulus, Dietrich. Evaluation of established line segment distance functions. *Pattern Recognition and Image Analysis 26* (2016).

[386] Wolf, Denis F, and Sukhatme, Gaurav S. Mobile robot simultaneous localization and mapping in dynamic environments. *Autonomous Robots* (2005).

[387] Wolf, Denis F, and Sukhatme, Gaurav S. Semantic mapping using mobile robots. *IEEE Transactions on Robotics 24* (2008), 245–258.

[388] Wray, Kyle, Zilberstein, Shlomo, and Mouaddib, Abdel-Illah. Multi-objective MDPs with conditional lexicographic reward preferences. In *AAAI Conference on Artificial Intelligence* (2015), vol. 29.

[389] Wray, Kyle Hollins, Witwicki, Stefan J, and Zilberstein, Shlomo. Online decision-making for scalable autonomous systems. In *International Joint Conference on Artificial Intelligence* (2017).

[390] Wu, Jiangqi, Wen, Linjie, Green, Peter L, Li, Jinglai, and Maskell, Simon. Ensemble Kalman filter based sequential Monte Carlo sampler for sequential Bayesian inference. *Statistics and Computing 32* (2022), 20.

[391] Wu, Junjun, Zhang, Hong, and Guan, Yisheng. An efficient visual loop closure detection method in a map of 20 million key locations. In *International Conference on Robotics and Automation (ICRA)* (2014), pp. 861–866.

[392] Xiaofeng, Ren, and Bo, Liefeng. Discriminatively trained sparse code gradients for contour detection. In *Advances in Neural Information Processing Systems* (2012).

[393] Xing, Chunwei, Sun, Xinyu, Cramariuc, Andrei, Gull, Samuel, Chung, Jen Jen, Cadena, César, Siegwart, Roland, and Tschopp, Florian. Descriptellation: Deep learned constellation descriptors for SLAM. *arXiv preprint arXiv:2203.00567* (2022).

[394] Xu, Lin, Hutter, Frank, Hoos, Holger H, and Leyton-Brown, Kevin. SATzilla: Portfolio-based algorithm selection for SAT. *Journal of Artificial Intelligence Research 32* (2008), 565–606.

[395] Yamauchi, Brian. A frontier-based approach for autonomous exploration. In *International Symposium on Computational Intelligence in Robotics and Automation* (1997), pp. 146–151.

[396] Yang, Anqi Joyce, Cui, Can, Bârsan, Ioan Andrei, Urtasun, Raquel, and Wang, Shenlong. Asynchronous multi-view SLAM. In *International Conference on Robotics and Automation (ICRA)* (2021), pp. 5669–5676.

[397] Yang, Shichao, Song, Yu, Kaess, Michael, and Scherer, Sebastian. Pop-up SLAM: Semantic monocular plane SLAM for low-texture environments. In *International Conference on Intelligent Robots and Systems (IROS)* (2016).

[398] Yang, Ziang, Zhang, Haobo, Zhang, Hongliang, Di, Boya, Dong, Miaomiao, Yang, Lu, and Song, Lingyang. MetaSLAM: Wireless simultaneous localization and mapping using reconfigurable intelligent surfaces. *IEEE Transactions on Wireless Communications 22* (2022), 2606–2620.

[399] Yi, Kwang Moo, Trulls, Eduard, Lepetit, Vincent, and Fua, Pascal. LIFT: Learned invariant feature transform. In *European Conference on Computer Vision (ECCV)* (2016).

[400] Yin, Deyu, Zhang, Qian, Liu, Jingbin, Liang, Xinlian, Wang, Yunsheng, Maanpää, Jyri, Ma, Hao, Hyyppä, Juha, and Chen, Ruizhi. CAE-LO: Lidar odometry leveraging fully unsupervised convolutional auto-encoder for interest point detection and feature description. *arXiv preprint arXiv:2001.01354* (2020).

[401] Yoon, Sangwoong, Kang, Woo Young, Jeon, Sungwook, Lee, SeongEun, Han, Changjin, Park, Jonghun, and Kim, Eun-Sol. Image-to-image retrieval by learning similarity between scene graphs. In *AAAI Conference on Artificial Intelligence* (2021), pp. 10718–10726.

[402] Yoon, Sung Wook, Fern, Alan, and Givan, Robert. FF-Replan: A baseline for probabilistic planning. In *International Conference on Automated Planning and Scheduling (ICAPS)* (2007), pp. 352–359.

[403] Younes, Georges, Asmar, Daniel, Shammas, Elie, and Zelek, John. Keyframe-based monocular SLAM: Design, survey, and future directions. *Robotics and Autonomous Systems 98* (2017), 67–88.

[404] Yu, Huizhen, and Bertsekas, Dimitri P. Basis function adaptation methods for cost approximation in MDP. In *IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning* (2009), pp. 74–81.

[405] Yuan, Miaolong, Yau, Wei-Yun, and Li, Zhengguo. Lost robot self-recovery via exploration using hybrid topological-metric maps. In *TENCON IEEE Region 10 Conference* (2018), pp. 0188–0193.

[406] Zanichelli, F. Topological maps and robust localization for autonomous navigation. In *International Joint Conference on Artificial Intelligence Workshop on Adaptive Spatial Representations of Dynamic Environments* (1999).

[407] Zbontar, Jure, LeCun, Yann, et al. Stereo matching by training a convolutional neural network to compare image patches. *Journal of Machine Learning Research 17* (2016), 2287–2318.

[408] Zender, Hendrik, Jensfelt, Patric, Mozos, O Martinez, Kruijff, Geert-Jan M, and Burgard, Wolfram. An integrated robotic system for spatial understanding and situated interaction in indoor environments. In *AAAI Conference on Artificial Intelligence* (2007), pp. 1584–1589.

[409] Zender, Hendrik, Mozos, O Martínez, Jensfelt, Patric, Kruijff, G-JM, and Burgard, Wolfram. Conceptual spatial representations for indoor mobile robots. *Robotics and Autonomous Systems 56* (2008), 493–502.

[410] Zeng, Andy, Song, Shuran, Nießner, Matthias, Fisher, Matthew, Xiao, Jianxiong, and Funkhouser, Thomas. 3DMatch: Learning local geometric descriptors from RGB-D reconstructions. In *Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 1802–1811.

[411] Zhang, Ji, Kaess, Michael, and Singh, Sanjiv. On degeneracy of optimization-based state estimation problems. In *International Conference on Robotics and Automation (ICRA)* (2016).

[412] Zhang, Ji, and Singh, Sanjiv. Visual-lidar odometry and mapping: Low-drift, robust, and fast. In *International Conference on Robotics and Automation (ICRA)* (2015), pp. 2174–2181.

[413] Zhang, Li, and Ghosh, Bijoy K. Line segment based map building and localization using 2D laser rangefinder. In *International Conference on Robotics and Automation (ICRA)* (2000), pp. 2538–2543.

[414] Zhang, Wei, Tanida, Jun, Itoh, Kazuyoshi, and Ichioka, Yoshiki. Shift-invariant pattern recognition neural network and its optical architecture. In *Conference of the Japan Society of Applied Physics* (1988), vol. 564.

[415] Zheng, Kaiyu, and Tellex, Stefanie. pomdp_py: A framework to build and solve POMDP problems. In *ICAPS 2020 Workshop on Planning and Robotics (PlanRob)* (2020).

[416] Zhou, Kun, Hou, Qiming, Wang, Rui, and Guo, Baining. Real-time KD-tree construction on graphics hardware. *ACM Transactions on Graphics 27* (2008), 1–11.

[417] Zhu, Jihua, Zheng, Nanning, Yuan, Zejian, Zhang, Qiang, Zhang, Xuetao, and He, Yongjian. A SLAM algorithm based on the central difference Kalman filter. In *Intelligent Vehicles Symposium (IV)* (2009), pp. 123–128.