Scalable Full Posterior Inference for Uncertainty-Aware Robot Perception

by

Qiangqiang Huang

B.E., Tsinghua University (2014)
M.S., Tsinghua University (2017)

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

Massachusetts Institute of Technology

September 2023

Authored by:	Qiangqiang Huang Department of Mechanical Engineering August 31, 2023
Certified by:	John J. Leonard Samuel C. Collins Professor of Mechanical and Ocean Engineering Thesis Supervisor
Accepted by:	Nicolas G. Hadjiconstantinou Chair, Department Committee on Graduate Theses

Scalable Full Posterior Inference for Uncertainty-Aware Robot Perception

by

Qiangqiang Huang

Submitted to the Department of Mechanical Engineering
on August 31, 2023, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

Robot perception is crucial for both fully autonomous systems, like self-driving cars, and human-centric devices such as mixed reality glasses. While advances have been made in perception problems like simultaneous localization and mapping (SLAM) and visual localization, the quest for self-diagnosable, robust systems capable of operating in large, complex environments continues.

This thesis aims to improve self-diagnosis and robustness in robot perception by promoting continuous uncertainty reasoning in localization and mapping, particularly under limited and ambiguous world observations. We investigate scalable and expressive approximations for posterior distributions in SLAM, overcoming the limited expressivity of Gaussian approximations for representing commonly encountered non-Gaussian posteriors. We harness the sparsity in factor graphs for scalability and utilize diverse density approximations to enhance expressivity. In advancing SLAM algorithms, we have achieved three contributions that provide unprecedented accuracy in describing posterior distributions, especially in highly non-Gaussian situations: 1) real-time inference of marginal posteriors by blending Gaussian approximation and particle filters, 2) incremental inference of joint posterior through learning normalizing flows on the Bayes tree, and 3) reference solutions to full posterior inference via nested sampling. Additionally, we develop a streaming platform that connects mobile devices and servers through web applications to conduct live demos of object-based SLAM, featuring the sharing of mapping results among online peers and continuous visualization of localization and mapping uncertainty.

We also introduce a novel application of full posterior inference for uncertainty-aware robot perception, focusing on evaluating camera pose localizability to pinpoint visual localization challenges in 3D scenes. By employing this framework, we optimize fiducial marker placements in 3D environments, boosting localization rates by 20%.

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

Acknowledgments

I have been fortunate to meet many wonderful people during my PhD journey. First and foremost, I owe immense gratitude to my advisor, John Leonard. He introduced me to the field of robotics, continuously offering insightful guidance and unwavering support. The intellectual, emotional, and financial freedom that John afforded me has been a cherished, once-in-a-lifetime experience. Our numerous meetings not only inspired my enthusiasm for exploration but also reminded me to be aligned with my research goals. Whenever I faced setbacks, John was the first to stand beside me, offering encouragement, wisdom, and optimism.

I would also like to thank my thesis committee members, Jonathan How and Themistoklis Sapsis. Collaborating with Jon on two papers was immensely rewarding. I was consistently amazed by the attention to detail that Jon exhibited. His in-depth feedback was invaluable in refining these papers and deepening my understanding of the field. The stochastic systems class taught by Themis provided me with a thorough exploration of probability theory, stochastic processes, and dynamical systems. Parts of this thesis have been significantly shaped by the insights I gained from these subjects.

I would like to extend my gratitude to my collaborators: Can Pu, Kasra Khosoussi, David Rosen, Dehann Fourie, Alan Papalia, Ziqi Lu, Jiahui Fu, Kevin Doherty, Joseph DeGol, Victor Fragoso, and Sudipta Sinha. This thesis would not have been possible without our collaboration. What I learned from all of you goes beyond doing research. I hope our paths cross again in the future.

I would also like to thank my labmates in the Marine Robotics Group, listed in the order of our first meetings: Yihao, Kevin, Dehann, Pedro, Kurran, Tonio, Brendan, Jiahui, Alan, and Ziqi. Our spontaneous discussions have considerably broadened my research horizons. The quick feedback from all of you has been a guiding light, ensuring I stay on course in my exploration and learning journey. I am grateful to Nira who keeps administrative operations running efficiently in the Marine Robotics Group.

My heartfelt gratitude also goes to my friends and family. The friendships I have forged during my PhD journey will be treasures for a lifetime. I have not seen my parents in person for five years, and our plans to travel together in the States have been postponed for various reasons. I hope to fulfill this plan very soon. As the elder brother, my debt to my sister Rong goes beyond mere words. I am grateful for the considerable time and effort she has committed to taking care of our family. I deeply appreciate the constant help and support from my cousin Gang's family. Their home has felt like a peaceful harbor for me in the States. I am incredibly lucky to have met my partner Ru in Boston. We have supported each other in both good times and challenging ones for many years. I am looking forward to the adventures that await us for the rest of our lives.

My PhD research was generously supported by ONR grant N00014-18-1-2832, ONR Neuroautonomy MURI grant N00014-19-1-2571, and the MIT Portugal Program.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

1.1.1 Uncertainty-aware robot perception

Robot perception refers to the process that a robot interprets its own state (e.g., location) and the environment around it using sensor data and prior information (e.g., pre-built maps, learned models). Fig. 1-1 illustrates this process. Not only does robot perception support planning and control in autonomous systems such as self-driving cars and mobile manipulators, but also it connects with human-centric devices such as mixed reality glasses to amplify humans' interaction with each other and the world. Thus robot perception serves a critical role no matter in a fully autonomous system or systems with a human in the loop [16]. While we have witnessed great research advances for solving perception problems such as simultaneous localization and mapping (SLAM) [18, 110, 19, 130], visual localization [115, 116, 140], object detection and pose estimation [127], and point cloud registration [136, 86], we still pursue self-diagnosable and robust perception systems that *safely* work in large-scale, complex environments.

In this thesis, the way we improve the self-diagnosis (or introspection in [110]) and robustness of robot perception is to achieve *uncertainty-aware robot perception* via inferring uncertainty in perception results (e.g., location). We present a moti-

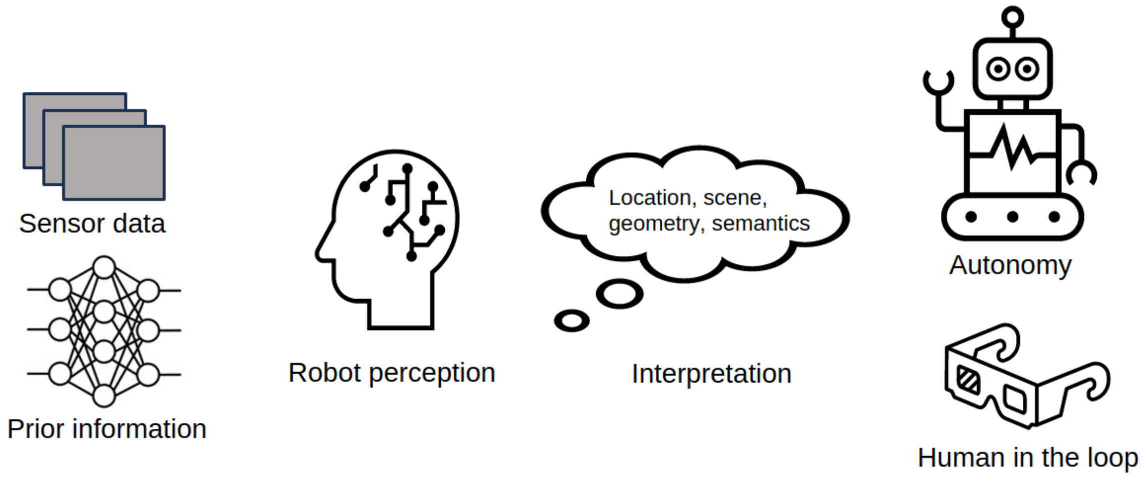


Figure 1-1: Illustration explaining robot perception. Robot perception interprets states of robots and environments using sensor data and prior information. It is pivotal in both fully autonomous systems (e.g., mobile manipulators) and human-in-the-loop systems (e.g., mixed reality glasses).

vating example to explain the advantages of evaluating uncertainty. The example is localization services provided by a map app (Fig. 1-2a) and a ride app (Fig. 1-2b). As seen in the figure, although these apps localize me to wrong spots, they manage to show large circles around estimated locations to indicate a significant level of localization uncertainty. Furthermore, alerted by uncertainty, I would carefully plan my next steps, for example, setting a good place to meet my driver in the case of the ride app, and look around to localize myself, which are similar to planning and active perception in autonomous systems for reducing uncertainty [110]. In summary, our investigation on uncertainty-aware robot perception is motivated by three advantages: 1) understanding uncertainty in perception results, 2) alerting us to potentially inaccurate results and guiding us where/how to improve the results, and 3) supporting autonomous systems for safe navigation and active perception.

While the importance and advantage of uncertainty awareness are general to various perception problems, methods for achieving uncertainty-aware robot perception depend on specific context. For example, data-driven methods are widely adopted to learn probability distributions of object pose or orientation from images [94]. As seen in Fig. 1-3a, when a robot sees a mug with an occluded handle, the robot's per-

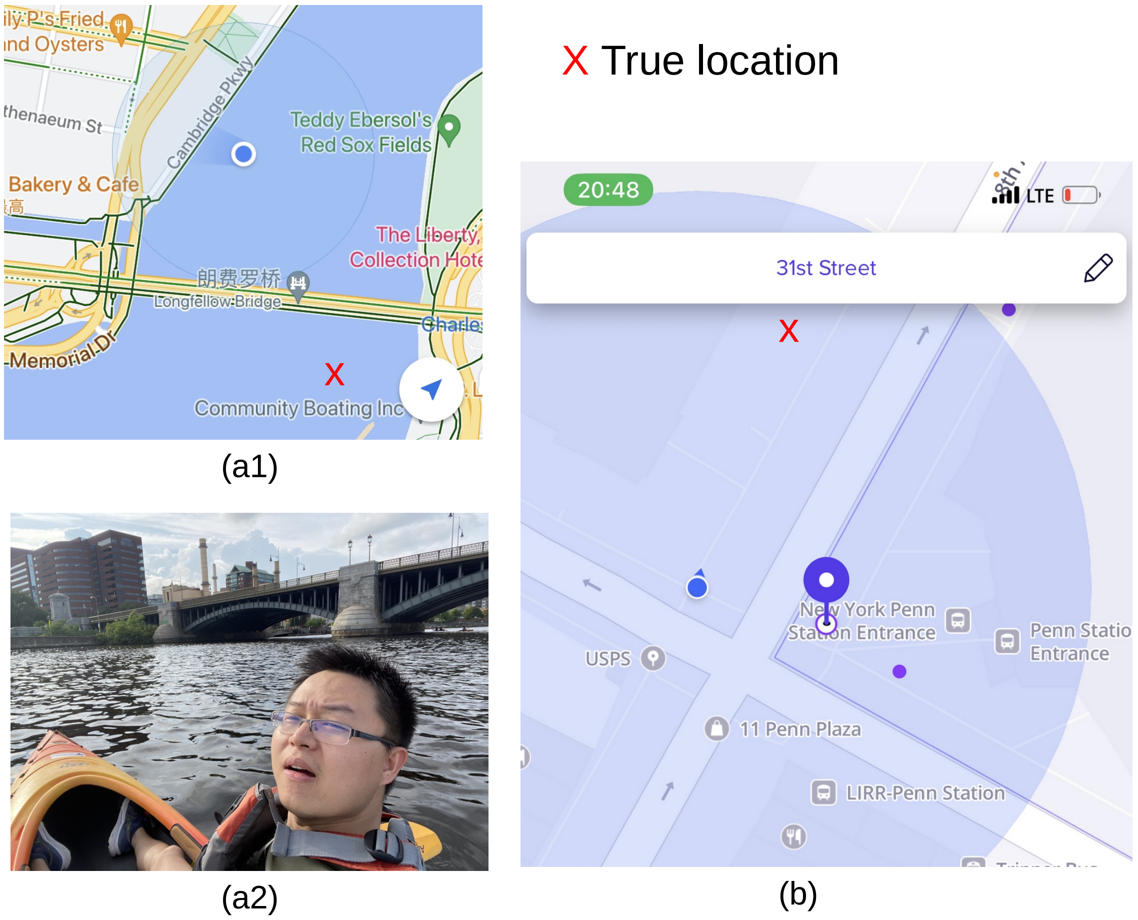


Figure 1-2: Example of apps for localization: (a1) and (a2) were taken on a cloudy day on the south of the Longfellow Bridge on the Charles river, and (b) was taken in the midtown of New York City. The red crosses indicate my true locations. The big circles in (a1) and (b), provided by these apps, alerted me potentially inaccurate localization services at these moments.

ception about the orientation of the handle should be very uncertain and described by a multimodal likelihood function rather than a unimodal likelihood centering at a single value. However, this thesis does not focus on how to perceive uncertainty from a single measurement like Fig. 1-3a. Our focus is to understand uncertainty in estimation problems for perception: developing full posterior inference algorithms that fuse probabilistic models about various measurements, as seen in Fig. 1-3b. Specifically, given likelihoods of measurements (e.g., relative pose, distance, direction), we seek parametric or sample-based approximations of posterior distributions of robot poses or landmark locations. In the thesis, we consider estimation problems in SLAM

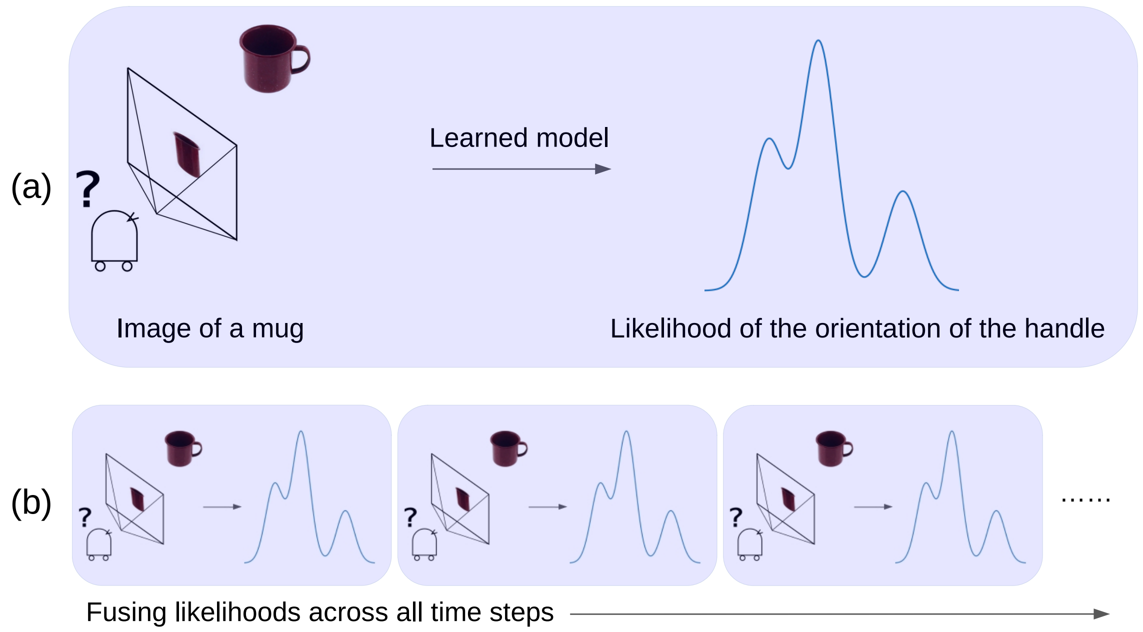

(a)
Learned model
Image of a mug
Likelihood of the orientation of the handle
(b)
Fusing likelihoods across all time steps

Figure 1-3: Example of tasks for uncertainty-aware robot perception: (a) single-frame measurement likelihood and (b) full posterior inference that fuses likelihoods from multiple time steps. This thesis focuses on algorithms for (b). We thank Ziqi Lu for providing the figure of the robot and mug [85].

and visual localization, which are arguably among the most important robot perception problems. In the following subsections, we discuss challenges encountered in our investigation on these two problems, respectively.

1.1.2 SLAM

Simultaneous localization and mapping (SLAM) is a foundational capability for mobile robots, enabling such basic functions as planning, navigation, and control. As such, the development of *robust*, *accurate*, and *computationally efficient* SLAM algorithms has been a major focus of research in robotics over the previous three decades [5, 18, 110].

We state the estimation problem for SLAM as inferring the *full posterior distribution* of latent variables (i.e., robot and landmark poses) provided noisy relative measurements between those variables. Note that full posterior inference is different from the point estimation typically seen in the SLAM literature (e.g., the maximum

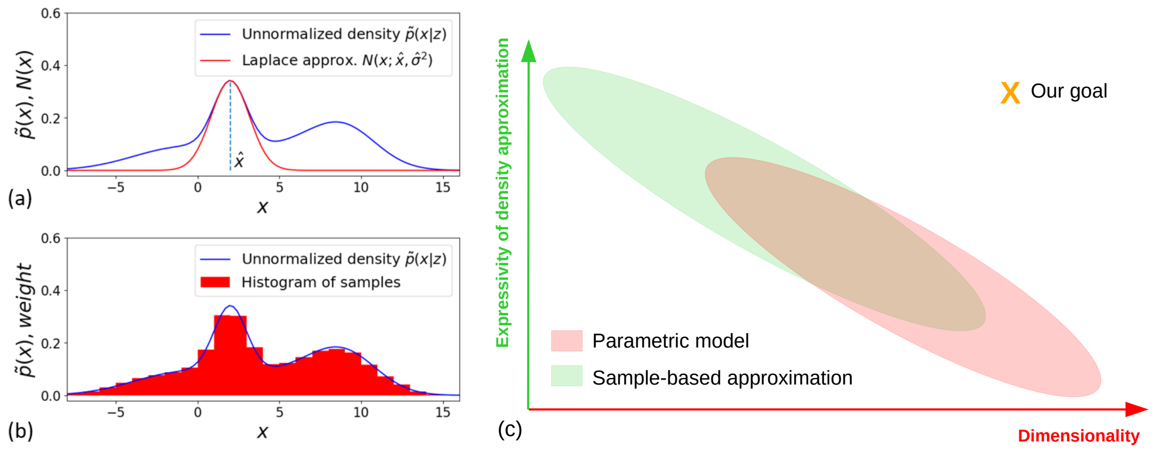


Figure 1-4: Approximations of an unnormalized posterior density, $\tilde{p}(x|z)$: (a) the Laplace (or Gaussian) approximation centering on the maximum a posteriori (MAP) estimate, $\hat{x}$, (b) sample-based approximation, and (c) sketch for roughly positioning scopes of different representations of the posterior density under similar computation budgets (interested readers can find a similar plot in [138, Fig. 2.2]) with a focus of showing the scalability-reliability trade-off across specific probabilistic inference algorithms).

a posteriori estimation in [18, Fig. 2]). We pursue full posterior inference since estimation of the distribution is required in many applications including probabilistic data association, collision avoidance, and active perception.

Standard methods for full posterior inference in SLAM research either compute a Gaussian approximation, centered on the point estimate, to the posterior (Fig. 1-4a) [28], or represent the posterior using samples (e.g., Fig. 1-4b) in the framework of Rao-Blackwellized particle filtering (RBPF) [91]. Constructing the Gaussian approximation enjoys great advantages in scalability for tackling high-dimensional posteriors, owing to efficient nonlinear optimization solvers for SLAM [71, 77]. However, the Gaussian approximation is inherently incapable of describing highly non-Gaussian/multi-modal posteriors, which often appear in realworld SLAM problems due to non-linear measurement models and non-Gaussian factors [111, 100]. Real-world examples that incur highly non-Gaussian posteriors include systems with range measurements [12], scale ambiguity in images [56], pose transformations on the special Euclidean group [83], multi-modal data association [34], the bimodal slip/grip behavior of odometry measurements [131], multipath effects of sonar and radar [126], the

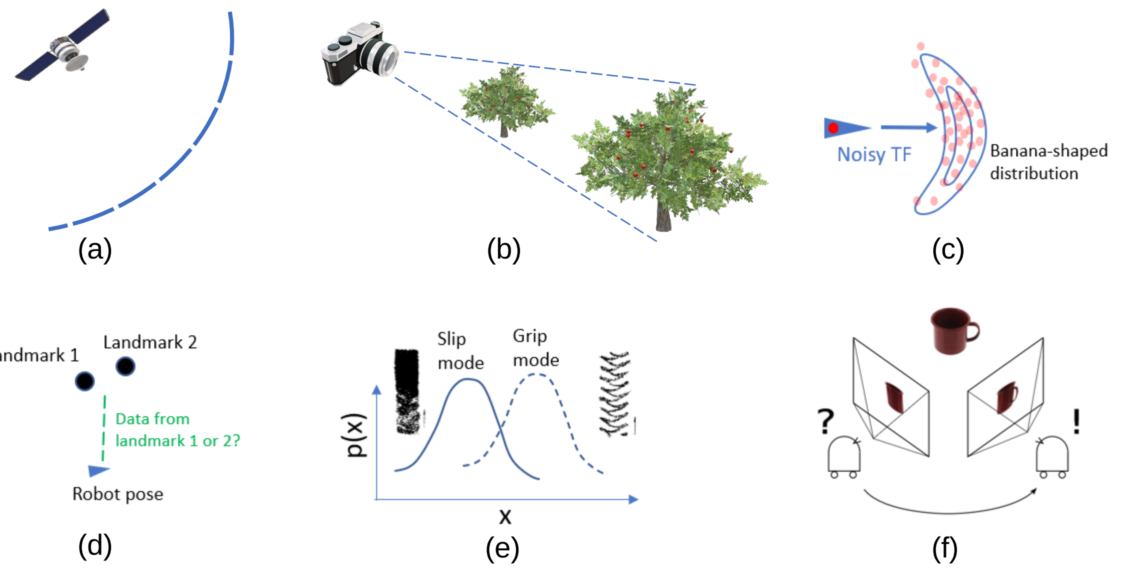


Figure 1-5: Examples of measurements that incur non-Gaussian posteriors of poses or locations: (a) range sensing which implies 2D ring-shaped or 3D sphere-shaped distributions of the receiver location, (b) RGB images where scaled objects along a ray may look the same, (c) noisy pose transformations (TF) that cause the high-curvature distribution, (d) multi-modal data association that connects multiple landmarks to a robot pose, (e) the uncertain traction state (i.e., slip and grip) between a wheel and the road surface that leads to multiple modes in the measurement about the displacement x, and (f) object pose ambiguity in images due to occlusion and symmetry. Image credit for (f): Ziqi Lu [85].

shadow matching technique for GNSS positioning in urban canyons [135, 53], and object pose ambiguity in images due to occlusion and symmetry [85, 94, 32, 46]. Some of these examples are illustrated in Fig. 1-5. While the RBPF can capture non-Gaussian features of the posterior via samples, it suffers from scalability issues, incurred by particle degeneracy and depletion [4], when dealing with high-dimensional posteriors. Fig. 1-4c positions the sample-based approximation and parametric models according to their strengths on pursuing greater expressivity and higher dimensionality. In summary, both non-Gaussian distributions and high dimensionality pose difficulties on scalable full posterior inference for SLAM while standard techniques usually focus on addressing one of the difficulties.

This thesis explores the sparsity structure in SLAM factor graphs and various density representations (e.g., samples, parametric models, and their hybrid) to tackle

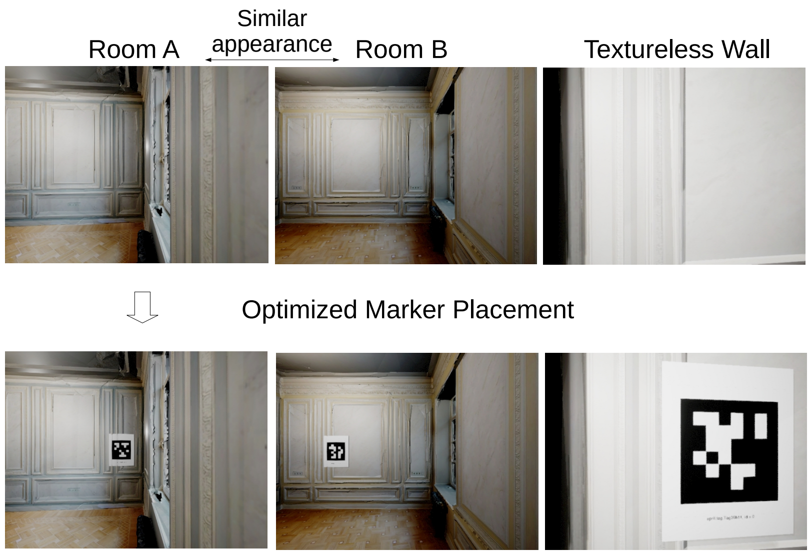


Figure 1-6: Three challenging examples for visual localization within the same scene. The images on the left and middle show two almost identical rooms in the scene, whereas the image on the right depicts a very weakly textured surface. Marker placements[1] in this scene guided by our optimized marker placement approach led to improved visual localization on these examples.

high dimensionality and non-Gaussian distributions, respectively, achieving scalable full posterior inference for SLAM.

1.1.3 Visual localization

Visual localization is a foundational technique for applications including AR/VR, autonomous driving, and robotic navigation and manipulation. A typical problem in visual localization is to estimate the camera pose of a query image, provided a pre-built map. While the problem has long been investigated in many fields [140], visual localization still suffers due to challenging scenes such as textureless walls and repetitive structures (e.g., Rooms A and B in Fig. 1-6). A quantitative analysis about camera localizability over the entire scene will help to identify difficult places for improved visual localization but still remains as an open problem. This thesis will define and compute camera localizability via applying full posterior inference to evaluate camera pose uncertainty that occurs in visual localization.

[1]Fiducial markers in the examples are AprilTags [99] but our algorithm is general and can be used with any existing family of fiducial markers.

In addition, one common solution to challenges in visual localization is to place fiducial markers as additional texture and identifiers in the scene [92, 27]; however, placing fiducial markers in larger environments is a time consuming process and the resulting performance improvement depends on marker positions. Traditionally, these marker locations are selected by humans who are familiar with visual localization techniques. This thesis will also explore the problem of automatic marker placement within a scene. Specifically, given a predetermined set of markers and a scene model, we aim to compute optimized marker positions within the scene that can improve accuracy in visual localization. The automatic selection of marker positions will be based on the camera localizability framework we mentioned above.

1.2 Contributions

This thesis develops scalable full posterior inference algorithms for SLAM that achieve superior accuracy in comparison to state-of-the-art SLAM algorithms in describing the full posteriors encountered in highly non-Gaussian SLAM settings. In addition, we identify a new application of full posterior inference for uncertainty-aware robot perception: evaluating uncertainty in the estimation of camera poses for visual localization. Contributions are stated in detail as follows:

1. We develop GAPSLAM, an algorithm that achieves real-time operation for inferring highly non-Gaussian marginal posteriors in SLAM. Specifically, we propose two techniques: 1) an adaptive modeling strategy whereby only marginals with great uncertainty are sampled by particle filters, while others are represented by the Gaussian approximation, and 2) an uncertainty-aware re-initialization technique that leverages particle filters to reset linearization points in the nonlinear optimization solver for computing the Gaussian approximation.

2. We develop NF-iSAM, an algorithm that achieves incremental non-Gaussian inference of the *joint* posterior in SLAM, warranting the scalability for handling high-dimensional non-Gaussian posteriors. Specifically, our technical contri-

butions are threefold: 1) introducing normalizing flows [101] to factor graph inference for robot perception, 2) generalizing the Bayes tree [71], which has beed used to implement incremental updates of the Gaussian approximation, to perform full (non-Gaussian) posterior inference for the joint posterior distribution, and 3) augmenting normalizing flows from low-dimensional inference to high-dimensional cyclic factor graphs.

3. We develop NSFG, an algorithm that provides high-quality samples for the offline validation of other inference methods such as GAPSLAM and NF-iSAM. NSFG adapts nested sampling methods to directly draw samples from the posterior, serving as reference solutions for full posterior inference at the expense of computation resources. Our technical contributions are twofold: 1) introducing nested sampling methods for solving robotics problems, and 2) exploiting the sparsity structure of SLAM factor graphs for improved sampling efficiency in nested sampling by providing more informative and sampling-efficient priors to nested sampling methods.

4. We propose a novel framework that models localizability of camera poses in a scene and computes localizability scores. This framework enables the development of OMP, a new algorithm that optimizes marker placement for visual localization based on scene features and fiducial markers. We demonstrate that the optimized marker placement by our approach can improve the localization rate by up to 20 percent on four different scenes.

Note that all the code and datasets for testing the algorithms above are open source online:

- GAPSLAM: `https://github.com/doublestrong/gapslam`
- NF-iSAM: `https://github.com/MarineRoboticsGroup/NF-iSAM`
- NSFG: `https://github.com/MarineRoboticsGroup/nsfg`
- OMP: `https://github.com/doublestrong/OMP`

1.3 Overview and publications

1.3.1 Thesis outline

The remainder of this thesis is organized as follows. Relevant publications we contributed to are also listed and * indicates equal contributors. Chapter 2 provides a review of general probabilistic inference methods and relevant ones to SLAM and visual localization.

Chapter 3 introduces the NSFG algorithm.

- Qiangqiang Huang, Alan Papalia, John J. Leonard. **N**ested **S**ampling for Non-Gaussian Inference in SLAM **F**actor **G**raphs. *IEEE Robotics and Automation Letters (RA-L) & IEEE/RSJ International Conference on Intelligent Robots and System (IROS)*, 2022.

Chapter 4 introduces the GAPSLAM algorithm.

- Qiangqiang Huang, John J. Leonard. GAPSLAM: Blending **G**aussian **A**pproximation and **P**article Filters for Real-Time Non-Gaussian **SLAM**. *IEEE/RSJ International Conference on Intelligent Robots and System (IROS)*, 2023.

Chapter 5 introduces the NF-iSAM algorithm.

- Qiangqiang Huang, Can Pu, Kasra Khosoussi, David M. Rosen, Dehann Fourie, Jonathan P. How, John J. Leonard. Incremental Non-Gaussian Inference for SLAM Using Normalizing Flows. *IEEE Transactions on Robotics (T-RO)*, 2023.

- Qiangqiang Huang*, Can Pu*, Dehann Fourie, Kasra Khosoussi, Jonathan P. How, John J. Leonard . NF-iSAM: **I**ncremental **S**moothing and **M**apping via **N**ormalizing **F**lows. *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

Chapter 6 introduces the OMP algorithm.

- Qiangqiang Huang, Joseph DeGol, Victor Fragoso, Sudipta N. Sinha, John J. Leonard. **O**ptimizing Fiducial **M**arker **P**lacement for Improved Visual Localization. *IEEE Robotics and Automation Letters (RA-L) & IEEE/RSJ International Conference on Intelligent Robots and System (IROS)*, 2023.

1.3.2 Additional work not in the thesis

In addition, we have contributed to two related papers from our lab (i.e., the Marine Robotics Group at MIT), in the object-based SLAM area, which inspired some experiments in Chapter 4 for demonstrating the GAPSLAM algorithm.

- Ziqi Lu*, Qiangqiang Huang*, Kevin Doherty, John J. Leonard. Consensus-Informed Optimization Over Mixtures for Ambiguity-Aware Object SLAM. *IEEE/RSJ International Conference on Intelligent Robots and System (IROS)*, 2021.

- Jiahui Fu, Qiangqiang Huang, Kevin Doherty, Yue Wang, John J. Leonard. A Multi-Hypothesis Approach to Pose Ambiguity in Object-Based SLAM. *IEEE/RSJ International Conference on Intelligent Robots and System (IROS)*, 2021.

Chapter 2

Preliminaries and literature review

2.1 Preliminaries

2.1.1 Notation

General notation: Deterministic values are denoted by lowercase letters while random variables are indicated by uppercase letters. If $\mathcal{V}$ is a set of indices, then $x_{\mathcal{V}}$ denotes a vector or collection of variables associated with those indices. For example, $x_{\{i,i+1,\dots,j\}} = (x_i, x_{i+1}, \dots, x_j)$. A vector of variables with all n indices is bolded, e.g., $\mathbf{x} = x_{\{1,2,\dots,n\}}$. Particularly, $x_{<d} = (x_1, x_2, \dots, x_{d-1})$ and $x_{>d} = (x_{d+1}, x_{d+2}, \dots, x_n)$. We use $p(X)$ to denote the probability density function $p_X(\cdot)$ of random variable X. We denote the function $p_X(x)$ at the deterministic value x by $p(x)$ or $p(X = x)$. We use $m(\cdot)$ to indicate a non-negative potential function. We use $g(\cdot; \mathbf{w})$ to indicate a function $g(\cdot)$ that is determined by parameters $\mathbf{w}$. We denote the sampling of X from a distribution $p(X)$ using the notation $x \sim p(x)$. The resulting n i.i.d. samples are denoted by $\{x^{(k)}\}_{k=1}^{n}$. The conditional independence relation $X \perp\!\!\!\perp Y | Z$ reads X and Y are conditionally independent given Z.

Graphical model notation: We define a factor graph $\mathcal{G}(\mathbf{f}, \mathbf{\Theta}, \mathcal{E})$ by nodes of random variables $\mathbf{\Theta}$ and factors $\mathbf{f}$ and edges $\mathcal{E}$ between variables and factors. Variables that are adjacent to factor f_i are denoted by $\Theta_{f_i} = \{\Theta_i \in \mathbf{\Theta} | (\Theta_i, f_i) \in \mathcal{E}\}$. We use C to denote a node or clique on the Bayes tree. We also use C to denote the collection of

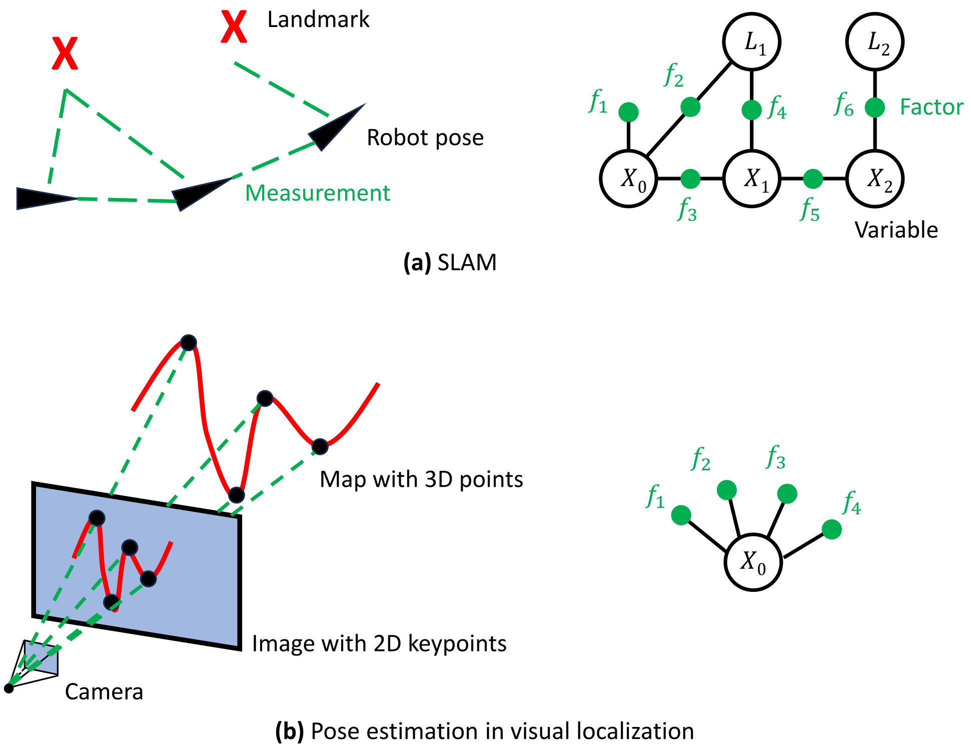


Figure 2-1: Examples of perception problems (left) and factor graphs (right) represent posterior distributions encountered in the problems: (a) a SLAM example where we estimate robot poses X_s and landmark locations L_s given measurements such as relative poses and distances, and (b) a visual localization example where we estimate the camera pose X_0 given matches between 2D keypoints and 3D points. A factor refers to a probabilistic model of a measurement or prior.

variables in the clique so $C \subseteq \boldsymbol{\Theta}$. The collection of child cliques of C is denoted by $\mathcal{C}_C$. The parent clique of C is Π_C. The intersection of clique C and its parent Π_C is called separator $S_C = C \cap \Pi_C$ while the remaining variables in C are called frontal variables $F_C = C \setminus \Pi_C$.

2.1.2 Posterior distributions and factor graphs

Chapter 1 introduces that this thesis focuses on reasoning uncertainty in estimation problems for robot perception. Before stating any estimation problem, we present the probabilistic description of the latent variable that we aim to estimate. The latent variable $\boldsymbol{\Theta} := (\Theta_1, \Theta_2, \ldots, \Theta_n)$ is a high-dimensional random variable with n components such as robot poses and landmark locations. All measurements are denoted by $\mathbf{z}$. The posterior distribution of the latent variable is

$$p(\mathbf{\Theta}|\mathbf{z}) = \frac{p(\mathbf{z}|\mathbf{\Theta})p(\mathbf{\Theta})}{p(\mathbf{z})} \propto p(\mathbf{z}|\mathbf{\Theta})p(\mathbf{\Theta}) = \prod_{i=1}^{m} f_i(\Theta_{f_i}), \tag{2.1}$$

where m is the number of factors. A factor $f_i(\Theta_{f_i})$ represents either a measurement likelihood or a prior. A prior factor has density $f_i(\Theta_{f_i}) = p(\Theta_{f_i})$, where Θ_{f_i} are variables involved in factor f_i. A likelihood factor represents density $f_i(\Theta_{f_i}) = p(z_i|\Theta_{f_i})$ where z_i is the measurement in $\mathbf{z}$ that is associated with factor f_i. The posterior distribution is often expressed with factor graphs [31], which are bipartite graphical models consisting of variable and factor nodes. Fig. 2-1 shows factor graphs of a SLAM example and a visual localization example. Factor graphs highlight the sparsity structure or conditional independence relations [9] which will be essential assets for us to develop scalable algorithms.

We introduce basic methods to find a point estimate of the latent variable and to approximate the posterior distribution in the following two subsections, respectively.

2.1.3 Maximum a posteriori (MAP) estimation via optimization

In perception problems such as SLAM or visual localization, the primary task is to find optimal configurations of the latent variable (e.g., robot poses and landmark locations) that best explain measurements and priors. The MAP estimation is widely exploited to search such a configuration by maximizing the posterior, as expressed in

$$\hat{\boldsymbol{\theta}} = \operatorname*{argmax}_{\boldsymbol{\theta} \in \mathcal{M}} p(\mathbf{\Theta}|\mathbf{z}), \tag{2.2}$$

$$= \operatorname*{argmax}_{\boldsymbol{\theta} \in \mathcal{M}} \prod_i f_i(\Theta_{f_i}) \tag{2.3}$$

where $\hat{\boldsymbol{\theta}}$ is the point estimate of the latent variable and $\mathcal{M}$ denotes the domain that the latent variable must belong to. For example, if the latent variable $\mathbf{\Theta}$ refers to a 6 Degree-of-Freedom (DoF) pose and a 3D location, the domain would be $\mathrm{SE}(3) \times \mathbb{R}^3$.

In practice, measurements are often assumed being perturbed by zero-mean, normally distributed noise for which factors are formulated as

$$f_i(\Theta_{f_i}) \propto \exp\left(-\frac{1}{2}\|h_i(\Theta_{f_i}) - z_i\|^2_{\Sigma_i}\right). \tag{2.4}$$

Thus the MAP estimation problem is equivalent to a nonlinear least-squares problem, for which many existing optimization solvers such as ceres [2], GTSAM [29], and g2o [77] can efficiently find a solution. The solution is usually an approximate MAP point due to linearization errors and local optima of the posterior. Note that the Gaussian noise is not necessary for creating the nonlinear least-squares problem since it has been proven that the MAP estimation can be converted to a nonlinear least-squares minimization under mild conditions [111, Theorem 1].

2.1.4 Full posterior inference via Gaussian approximation

Since Chapters 4 and 6 still exploit the Gaussian approximation for developing new algorithms, we briefly review how to construct a Gaussian to *locally* approximate the posterior via the Laplace approximation [9, Ch. 4.4]. The Gaussian centers on the MAP estimate $\hat{\boldsymbol{\theta}}$ with a covariance

$$\hat{\Sigma} \approx \{\text{Hess}[-\ln p(\mathbf{\Theta}|\mathbf{z})]|_{\mathbf{\Theta}=\hat{\boldsymbol{\theta}}}\}^{-1}, \tag{2.5}$$

which is the inverse of an estimated Hessian of the negative logarithm of the posterior. In the nonlinear least-squares form of the MAP estimation problem, the Hessian can be approximated as the square of the Jacobian A at the MAP estimate, i.e., $A^\intercal A$ [70, Sec. 2]. Thus we define the Gaussian approximation

$$g(\mathbf{\Theta}) = \mathcal{N}(\mathbf{\Theta}|\hat{\boldsymbol{\theta}}, \hat{\Sigma}), \tag{2.6}$$

where $\mathcal{N}(\cdot)$ denotes a normal distribution. Note that the substraction operation in the expression of the Gaussian depends on the domain or manifold of $\mathbf{\Theta}$.

2.2 Literature review

We briefly review related work and point out connections with proposed approaches in this thesis.

2.2.1 Sampling methods for probabilistic inference

The related work in this section connects with the NSFG algorithm in Chapter 3 which also directly draws samples from the posterior. These samples are recognized as more computationally expensive, but more accurate, approximation of the full posterior so they serve as reference solutions for evaluating other full posterior inference algorithms in this thesis. A brief review of state-of-the-art sampling methods is as follows.

Hamiltonian Monte Carlo (HMC) is an Markov Chain Monte Carlo (MCMC) variant which guides sampling exploration with the gradient of the density function [8]. The No-U-Turn Sampler (NUTS) [59] extends HMC with automatic parameter tuning, improving usability and performance. However, multi-modal distributions with distant modes still pose general difficulties as state space exploration uses only local density information. Specialized MCMC algorithms designed for multi-modal distributions were recently developed [128, 105].

Sequential Monte Carlo (SMC) is a sampling algorithm which combines importance sampling, re-sampling, tempering, and MCMC. Particle filters in general are just special instances of SMC [36]. The use of tempering and MCMC alleviates sample impoverishment in standard particle filters (e.g., sampling importance resampling algorithms in [4, Alg. 4]). SMC usually requires a proposal distribution that possesses good coverage of the typical set of the target density. It is difficult to design such a proposal distribution for a general high dimensional target density.

Nested sampling was proposed by Skilling [120] to compute the evidence or marginal likelihood for Bayesian inference with a by-product of posterior samples. It is mostly developed and used in the field of astronomy. Nested sampling has two attractive features: (1) well-defined stopping criteria related to the convergence of estimated evidence and (2) global exploration of the state space. Nested sampling (NS) methods keep exploring and drawing new samples from the state space until the estimate of evidence converges. NS has demonstrated great success with complex posterior distributions that possess multiple modes [17]. The practical benefits of NS methods include: (i) no predetermined number of samples and (ii) little to no tuning of proposal distributions is required to get accurate results. The estimated evidence by nested sampling is also helpful for model selection problems (e.g., multiple factor graphs under different data associations). There are several open-source nested sampling packages [40, 55, 123]. In particular, [123] developed dynesty, an open-source dynamic nested sampling package which our implementation of NSFG in Chapter 3 is built upon.

The SLAM community has already explored sampling methods such as particle filtering [51, 13, 91] and Markov Chain Monte Carlo (MCMC) [119]. Many particle filtering algorithms for SLAM (e.g., FastSLAM2.0 [91]) track landmarks by extended Kalman filters which cannot represent general non-Gaussian/multi-modal distributions. Recent work leveraged factor graphs and probabilistic modeling techniques such as non-parametric belief propagation [45, 42] and normalizing flows [64]. NSFG differs from these works in that it does not assume any parametric density models and builds upon a stochastic inference technique (i.e. nested sampling) which is generally more accurate for multi-modal distributions [123]. NSFG aims to improve inference fidelity at the cost of computational complexity. This tradeoff suggests that NSFG could be used in accuracy evaluation of approximate distributions found by SLAM techniques.

2.2.2 MAP estimation and sparsity structure in SLAM

The literature review in this section is related to the NF-iSAM algorithm in Chapter 5 because NF-iSAM exploits the same sparsity structure of factor graphs as state-of-the-art SLAM algorithms for improved computation efficiency, with an extension to (non-Gaussian) full posterior inference.

The state-of-the-art optimization-based solutions to SLAM, such as iSAM2 [71], are MAP-based *point* estimators that approximate the posterior distribution by a single, parametric Gaussian model. iSAM2 presents the Bayes tree [71], a graphical model that provides a probabilistic interpretation for sparse linear algebra in square root smoothing and mapping ($\sqrt{SAM}$) [30] and incremental smoothing and mapping (iSAM) [72]. The Bayes tree can be generalized to non-Gaussian settings since it is a result purely based on conditional independence structures in graphical models; however, in iSAM2, the estimated model of the joint posterior is still constructed by linear-Gaussian conditionals [31, Sec. 3.4]. Recent works of parametric SLAM solutions focus on robust MAP estimation in the presence of outliers or multi-modal factors. [110] extensively reviews robust estimation techniques including switchable constraints [126], robust cost functions [1, 111], and mixture models [100, 104]. However, these techniques do not aim at capturing the shape of the full posterior.

Alternatively, nonparametric models yield more expressive representations of the full posterior. These methods use sampling techniques, such as particle filters, MCMC, or nested sampling [90, 132, 63]. The most well-known nonparametric SLAM algorithm is FastSLAM [90] which is based on Rao-Blackwellized particle filters. FastSLAM leverages the relation that map features are conditionally independent once robot poses are given. However, due to sample impoverishment in particle filters, smoothed estimates degenerate as the loss of diversity in particles' paths [4]. In order to further exploit conditional independence relations, a more recent method, multimodal-iSAM (mm-iSAM) [43], leverages the Bayes tree [71] to solve SLAM problems with a variety of non-Gaussian error sources. mm-iSAM uses nested Gibbs sampling, derived from nonparametric belief propagation [125], to approximate the *marginal* belief of each node on the Bayes tree. As a direct extension of iSAM2, MH-iSAM2 [60] explicitly solves point estimates of multiple modes for SLAM problems involving multiple hypotheses (e.g., uncertain data association and ambiguous loop closures), but it cannot directly tackle more general factors such as range measurements. In contrast, the NF-iSAM algorithm in Chapter 5 models and draws samples from the joint posterior and is able to deal with both multi-hypothesis factors (e.g., multi-modal data association) and nonlinear measurement factors (e.g., range measurements).

NF-iSAM extends iSAM2 to non-Gaussian settings, because NF-iSAM replaces the Gaussian posterior in iSAM2 by non-Gaussian posteriors which are represented by normalizing flows, a learning-based technique for probabilistic modeling. Full posterior inference has also drawn researchers' interest in the machine learning community. Many tools such as kernel embedding [50, 74] and Gaussian copula [79, 87] have been leveraged to model non-Gaussian densities. A recent class of algorithms aims to draw samples from a non-Gaussian target distribution by estimating a transformation that maps samples from a simple reference distribution onto the target. These algorithms are known as transport maps [38, 102], or normalizing flows [108]. Although they have shown good performance in modeling very complex densities, research on high-dimensional probabilistic graphical models is limited. Chapter 5 will show that NF-iSAM augments normalizing flows from low-dimensional inference to high-dimensional cyclic factor graphs.

2.2.3 Full posterior inference for SLAM

This section connects to the GAPSLAM algorithm in Chapter 4 because GAPSLAM blends both Gaussian approximation and particle filters, which are arguably standard techniques for full posterior inference for SLAM. We briefly review some work on full posterior inference for SLAM with a focus on non-Gaussian representations of the posterior. Note that the non-Gaussian representation can be parametric models (e.g., sum of Gaussians) or non-parametric (i.e., samples).

The framework of RBPF has been leveraged in a big class of SLAM algorithms (e.g., FastSLAM) [91, 12, 11], where the posterior of robot poses is represented by a set of particles, and each particle is attached with parametric models [91, 11] or samples [12] of the conditional of the map. Although our work exploits the same conditional independence relation as FastSLAM, i.e., landmarks are conditionally independent given robot poses, our work swaps representations of the robot path and map in RBPF, resulting in a parametric (Gaussian) model to describe the belief about robot poses and a set of particle filters to independently draw samples from the posterior of each landmark. Thus the issue of losing diversity in robot path particles no longer exists, warranting the scalability of our work. We will compare to [11] in our experiments section. mm-iSAM [45] and NF-iSAM [61] are recent non-Gaussian inference algorithms that leverage the Bayes tree algorithm [71] to exploit more conditional independence relations. As solvers for general factor graphs, they can tackle both nonlinear measurement models and multi-modal data association, while our work is only focused on nonlinear measurement models. Another class of approaches purely relies on Monte Carlo techniques to directly draw samples from the joint posterior [132, 119, 63]. These methods, in general, do not suit real-time online applications due to the exorbitant computational cost but can serve as reference solutions to full posterior inference.

The GAPSLAM algorithm in Chapter 4 is inspired by some works [25, 11, 12] dedicated to bearing-only (ponit-object-based) or range-only SLAM. Existing approaches for these two problems can be categorized as batch optimizations [96], delayed initializations of landmarks [82, 25], and undelayed initializations of landmarks [78, 121, 11, 12]. Our work falls into the undelayed category. Our experiments show that we can solve both of these problems in a unified framework and demonstrate the evolution of landmark posteriors since time step zero. Furthermore, we clarify that, while our object-based SLAM system requires similar input (RGB images and object detections) as the systems in [97, 137, 98, 112], we do not estimate occupied areas of objects and focus soloty on inferring how distributions of object locations evolve. The ellipsoids in the object-based SLAM section are confidence intervals of object locations rather than models often seen in those works for indicating the 3D occupancy of objects in a scene.

2.2.4 Visual localization and fiducial markers

Chapter 6 investigates how to optimize the positions of fiducial markers for improved visual localization. We briefly review some recent work related to mapping and localization with fiducial markers and marker/landmark placement optimization. Examples of fiducial markers include tag families with explicit IDs (e.g., ArUco markers [49], AprilTag [99], ChromaTag [26]) and emerging learning-based marker designs [139].

Fiducial markers are widely recognized as an effective approach for improving localization and mapping accuracy. DeGol et al. [27] demonstrate that marker IDs are useful in image matching and resectioning for structure from motion (SfM), leading to improvements in reconstruction results. The UcoSLAM system [92] integrates marker detection with a bag-of-words approach and presents more robust tracking and relocalization than SLAM techniques with no marker detection [93, 47]. However, marker placements in these SfM or SLAM systems are manually determined and not planned by algorithms.

Existing work about marker deployment focuses on robotic localization without considering scene features [21, 134, 69]. Beinhofer et al. [7] explore optimal placement of artificial landmarks such that a robot equipped with range and/or bearing sensors repeatedly follows predetermined trajectories in planar environments with improved accuracy. Meyer-Delius et al. [89] introduce a measure that defines the uniqueness of robot poses in the context of Monte Carlo localization using laser scanners and then propose a greedy algorithm to incrementally select landmark locations for maximizing the measure. While we find the greedy algorithm is similar to ours, it is not straightforward to apply the measure to visual localization using images and scene features. Lei et al. [81] investigate landmark deployment for poses on SE(3) and demonstrate placing fiducial markers in a cubic environment; however, features in the scene are not involved in optimizing the marker placement.

On the note of uncertainty quantification in visual localization, [56, Ch. 5] reviews methods for computing the covariance of a transformation (e.g., 6 DoF poses) under settings of least-squares and Monte Carlo simulation. Zhang et al. [140] apply and develop those methods to characterize uncertainty of reference poses that are used for benchmarking visual localization approaches. The uncertainty is then exploited to define a new evaluation metric that computes the percentage of queries localized within per image error thresholds rather than fixed thresholds. The OMP algorithm in Chapter 6 leverages a similar method as the first order approximation in [140, Sec. 3.4] to compute the covariance of a reference camera pose. However, our focus is to utilize the covariance to define localizability scores over the entire environment and optimize fiducial marker placement for increased localizability scores.

Chapter 3

Reference solutions to full posterior inference via nested sampling

The content in this chapter is mainly based on the following paper:

- Qiangqiang Huang, Alan Papalia, John J. Leonard. **N**ested **S**ampling for Non-Gaussian Inference in SLAM **F**actor **G**raphs. *IEEE Robotics and Automation Letters (RA-L) & IEEE/RSJ International Conference on Intelligent Robots and System (IROS)*, 2022.

3.1 Introduction

As introduced in Chapter 1.1.2, due to nonlinearities in real-world sensing models or multi-modal factors (e.g., multi-modal data association), the SLAM posterior is in general non-Gaussian. Exact inference over those non-Gaussian posteriors is generally intractable so practitioners resort to either deterministic approximations (e.g., Gaussian approximation) or sampling methods [9]. Sampling methods, which directly draw samples from the posterior distribution, are generally recognized as more computationally expensive, but more expressive, inference solutions [9]. Our work in this chapter falls into this class of algorithms. The resulting samples of these algorithms enable qualitative analysis of the posterior as well as estimation of statistical quantities of interest. Thus, stochastic approximation algorithms provide value in (1) offline accuracy evaluation of other algorithms and (2) high-fidelity posterior estimation for problems which do not require real-time results onboard a robotic system.

Nested sampling is a recent stochastic approximation technique that is powerful for sampling multi-modal distributions [120]. This work combines nested sampling with informative priors obtained from factor graphs. We term our algorithm nested sampling for factor graphs (NSFG). To the best of our knowledge, there are no existing general purpose algorithms for reference solutions even on posteriors of small SLAM problems. This need for accuracy evaluation motivated the development of NSFG, which pursues the *bona fide* shape of the posterior and thus aids accuracy evaluation of other SLAM inference algorithms.

3.2 Problem statement

We define the SLAM inference problem that NSFG solves. We use the latent variable $\boldsymbol{\Theta}$ to denote all robot poses (X) and landmark locations (L). Let $\mathbf{z}$ be all measurements. *We aim to sample the posterior distribution of* $\boldsymbol{\Theta}$ *given all measurements*, i.e.

$$p(\boldsymbol{\Theta}|\mathbf{z}) \propto p(\mathbf{z}|\boldsymbol{\Theta})p(\boldsymbol{\Theta}) = \prod_{i=1}^{m} f_i(\Theta_{f_i}) \tag{3.1}$$

where m is the number of factors in the factor graph representation of the posterior.

3.3 Approach

We discuss nested sampling at a high level, and elaborate on how NSFG uses the conditional independence structure of factor graphs to improve the abilities of nested sampling.

3.3.1 Nested sampling

Nested sampling [120] was proposed to compute the evidence $p(\mathbf{z})$, with a by-product of posterior samples. The evidence is defined by an integral of the likelihood function $L(\mathbf{z}|\mathbf{\Theta})$ and the prior distribution $\pi(\mathbf{\Theta})$ over the latent variable $\mathbf{\Theta}$, as seen in

$$p(\mathbf{z}) = \int_{\mathbf{\Theta}} L(\mathbf{\Theta})\pi(\mathbf{\Theta})d\mathbf{\Theta}, \tag{3.2}$$

where $L(\mathbf{\Theta})$ stands for the likelihood $L(\mathbf{z}|\mathbf{\Theta})$. In nested sampling, samples of $\mathbf{\Theta}$ are drawn from the prior distribution. The prior distribution must cover all variables and is ideally computationally efficient to sample. The likelihood function contains the remaining factors of the posterior distribution.

Conceptually, nested sampling breaks up the sample space of the prior into *nested* areas. Each nested area is enclosed by an iso-likelihood contour. We define the probability of a nested area as the prior volume $V \in [0, 1]$, while the likelihood on the contour is denoted by $L(V)$; thus, the small prior volume dV denotes the probability of a small iso-likelihood shell in the sample space. With the notion of prior volumes, nested sampling transforms the integration (3.2) to a one-dimensional integral of the likelihood over iso-likelihood small prior volumes, as follows [120, 17]:

$$p(\mathbf{z}) = \int_0^1 L(V)dV. \tag{3.3}$$

In practice, each of the samples from the prior represents a small prior volume dV, and the sum of all small prior volumes is 1. Thus, the integration (3.3) can be implemented numerically by the likelihood-weighted sum of those prior volumes to estimate the evidence. To increase the precision of the numerical integration, nested sampling chooses to draw more samples from the prior volume where the likelihood is higher. This can be visualized using the V-L plot in Fig. 3-1b. Each bin under the V-L curve corresponds to a sample. The small prior volume and likelihood of a sample, respectively, determine the width and height of the bin. Those bins are ordered from left to right following a descending order of sample likelihoods (or bin heights). Thus, more samples naturally lead to finer bins that approach the theoretical V-L curve better.

Instead of drawing all samples of $\mathbf{\Theta}$ only once from the prior distribution, nested sampling draws samples across iterations. With each iteration the feasible space of new samples gradually shrinks to high-likelihood areas in the sample space, leading to an efficient and accurate estimate of the evidence. This gradual focusing is often referred to as likelihood-restricted prior sampling (LRPS) [17], and is the most important step in nested sampling. We stress that LRPS is different from proposals in particle filters. While particle filters attempt to draw samples of a variable once from a proposal, the LRPS draws new samples across iterations until the estimated evidence converges; additionally, the weight of each sample depends on not only the likelihood but also the small prior volume. As the contribution of our work is not in performing nested sampling, but in using conditional independence structures from factor graphs to more efficiently prepare a problem for nested sampling, we refer interested readers to [120, 123, 17] for further details.

3.3.2 Nested sampling for factor graphs

Proposed approach

While nested sampling is a powerful approach to sampling complex distributions, naive application of nested sampling to SLAM does not take advantage of the conditional independence structure of SLAM. We focus on how to use this structure to enable nested sampling to be effectively applied to SLAM. Specifically, we exploit the sparsity structure of SLAM factor graphs to construct a prior $\pi(\mathbf{\Theta})$ and likelihood function $L(\mathbf{\Theta})$, which enhances nested sampling.

Factors in SLAM factor graphs usually consist of a few unary factors as priors, a number of binary factors modeling measurement likelihoods, and a few factors connected to a robot pose and multiple landmarks for modeling multi-model data association. As described in Section 3.3.1, the prior model for nested sampling, $\pi(\mathbf{\Theta})$, must be a tractable distribution from which samples of all latent variables can be effi-

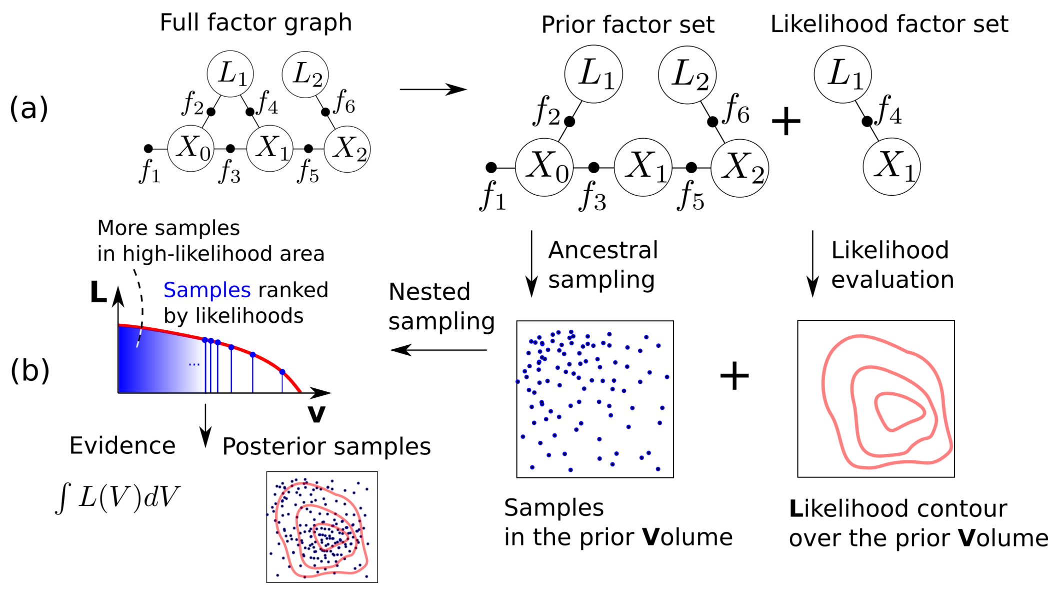


Figure 3-1: Our approach to performing nested sampling over factor graphs. (a) The factor graph is first decomposed into two parts: a prior factor set and a likelihood factor set. (b) From the acyclic graph in the prior factor set we apply ancestral sampling to generate samples of the prior distribution. The functions for sampling and likelihood evaluation are supplied to a nested sampler. Nested sampling returns estimated evidence and samples which approximate the posterior distribution.

ciently drawn. Thus, $\pi(\mathbf{\Theta})$ must incorporate factors more than the nominal priors to cover all variables and, accordingly, these factors will be excluded from the likelihood model, $L(\mathbf{\Theta})$. We will introduce our strategy for selecting factors that compose $\pi(\mathbf{\Theta})$ and $L(\mathbf{\Theta})$.

Our strategy constructs a $\pi(\mathbf{\Theta})$ that enables ancestral sampling [9] for all variables, as ancestral sampling admits very efficient distributional sampling. NSFG effectively builds $\pi(\mathbf{\Theta})$ from a spanning tree, for trees naturally afford ancestral sampling.

NSFG assumes that any variable in the SLAM factor graph is connected to at least a binary factor (i.e., each variable is created along with a bivariate factor), implying that binary factors already form a connected graph of all variables. NSFG designates a node or variable connected to a prior factor as the root of the spanning tree. Starting from samples drawn from the prior factor for the root variable, one can use binary factors along the spanning tree to generate samples of descendant variables up to the leaves of the tree. As seen in Fig. 3-1a, we designate the factors involved in this ancestral sampling procedure as **PF** (prior factor set) and incorporate them in the prior model $\pi(\mathbf{\Theta})$ while the remaining factors are referred to as **LF** (likelihood factor set) and make up the likelihood model, $L(\mathbf{\Theta})$. The resulting factorization of the posterior distribution in (3.1) is

$$p(\mathbf{\Theta}|\mathbf{z}) \propto \underbrace{\prod_{f_j \in \mathbf{LF}} f_j(\Theta_{f_j})}_{L(\mathbf{\Theta})} \underbrace{\prod_{f_i \in \mathbf{PF}} f_i(\Theta_{f_i})}_{\pi(\mathbf{\Theta})} . \tag{3.4}$$

For example, for a typical SLAM factor graph with a single robot and K unknown landmarks, the **PF** set can be composed of the prior factor at the starting pose of the robot, odometry factors, and K binary factors that are connected to different landmarks. Notions such as the **PF** set have commonly been used in SLAM to initialize variables in MAP solvers and construct proposal distributions in particle filters. This work introduces such approaches to nested sampling, with a focus on sampling from joint posteriors in SLAM problems.

This partition of **PF** and **LF** sets improves nested sampling in two aspects: (i) the prior model resembles the posterior distribution better than simple proposals such as uniform distributions of all variables and (ii) the likelihood model involves fewer factors, which reduces the cost of likelihood evaluation in nested sampling. SLAM factor graphs are usually sparse, which implies that the cardinality of **PF** set can be comparable to or even much greater than the cardinality of **LF** set. Therefore, exploiting the sparsity structure of SLAM factor graphs can effectively improve both the computational performance and solution quality of nested sampling.

Algorithms

We implemented Algorithms 1 and 2 for obtaining the **PF** and **LF** sets and drawing posterior samples via NSFG. The factors in the **PF** set are constructed from a spanning tree, as seen in Algorithm 1, and are thus ordered for performing ancestral sampling. Note that selection of the prior factor for designating the root of the tree in line 3 of Algorithm 1 depends on user-defined heuristics (e.g., choosing a more informative prior). Using the **PF** and **LF** sets, Algorithm 2 defines the prior model

```
Algorithm 1: Obtain PF and LF sets (spanning tree)
  Input: Univariate factor set $\mathcal{P}$, binary factor set $\mathcal{B}$, and all other factors $\mathcal{L}$
  Output: Prior factor queue PF and likelihood factor queue LF
1 Initialize empty FIFO queues of PF and LF
2 Construct a spanning tree $\mathcal{T}$ from the graph formed by binary factors $\mathcal{B}$
3 Push a prior factor $f$ from $\mathcal{P}$ to PF and designate its variable as the root of
    $\mathcal{T}$
4 Traverse $\mathcal{T}$ from the root to leaves and push binary factors to PF once they
    have been visited
5 Push all factors that are not in PF to LF
6 return PF and LF
```

$\pi(\mathbf{\Theta})$ and likelihood model $L(\mathbf{\Theta})$ that enable nested sampling. The likelihood model is simply evaluating the sum of the log-likelihoods of the factors in the **LF** set (line 1 in Algorithm 2). The use of the **PF** set for realizing $\pi(\mathbf{\Theta})$ is less straightforward since it involves density transformation (line 10 in Algorithm 2).

Nested sampling methods usually require transformation functions that map from the uniform distribution over unit hypercube to the prior distribution $\pi(\mathbf{\Theta})$, rather than explicit expressions of prior distributions [40, 55, 123, 17]. We refer to these transformations as hypercube transforms. Hypercube transforms are necessary because the unit hypercube is first sampled to enable global exploration of the state space. The transform applied to these samples casts the global exploration into the domain of the variable $\mathbf{\Theta}$, improving global coverage of the state space. Hypercube transforms can be implemented by quantile functions of noise models, which are available in typical SLAM problems (e.g., Gaussian noise models). With these hypercube transforms and observation models, we obtain samples for our first variable in the **PF** set and then apply ancestral sampling to propagate the samples along the tree encoded in the **PF** set (line 8 in Algorithm 2).

Algorithm 2: NSFG

```
Input: Prior factor queue PF and likelihood factor queue LF
Output: Samples of the joint posterior distribution
1  Function LLK(latent variable $\boldsymbol{\Theta}$) :
2      $l \leftarrow 0$                                   // Initialize log likelihood
3      for $f$ in LF do
4          $l = l + f.\text{loglikelihood}(\boldsymbol{\Theta})$
5      return $l$
6  Function PriorTrans(hypercube sample $\boldsymbol{u}$):
7      Initialize a dictionary $\mathcal{P}$ for containing prior samples
8      for $f$ in PF do // Ancestral sampling in each iteration
9          $v \leftarrow$ Variable in $f$ but not in $\mathcal{P}$
10         $\mathcal{P}[v] \leftarrow f.\text{hypercube\_transform}(\boldsymbol{u}, \mathcal{P}, v)$
11     return $\mathcal{P}$
12 $\mathcal{S} \leftarrow NestedSampling(PriorTrans, LLK)$
13 return $\mathcal{S}$                                   // Return posterior samples
```

3.4 Implementation

3.4.1 Observation and noise models

We introduce observation and noise models that will be used in our experiments. A noisy pose observation is defined by $\tilde{T} = T \exp(\boldsymbol{\xi}^{\wedge})$ where pose $T \in \mathrm{SE}(d)$ is a latent variable, $\wedge$ turns $\boldsymbol{\xi}$ into a member of the Lie algebra $\mathfrak{se}(d)$, and $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \Sigma)$ is the perturbation vector subject to a Gaussian distribution. A noisy range measurement is modeled by $\tilde{r} = \|\mathbf{t}_i - \mathbf{t}_j\|_2 + \mathcal{N}(0, \sigma^2) \in \mathbb{R}$ where $\mathbf{t}_i$ is the translation component of variable X_i or L_i. Beyond binary factors with known data associations, we use sum-mixture factors to model ambiguous data association as follows

$$f_i(\Theta_{f_i}) = p(z_i|\Theta_{f_i}) = \sum_{j=1}^{|\mathcal{D}|} p(z_i|\Theta_{f_i}, d_j)p(d_j) \tag{3.5}$$

where each component is a binary factor with certain data association $d_j \in \mathcal{D}$. In our experiments we assume the data association prior $p(d_j)$ is a uniform distribution given no prior knowledge.

3.4.2 Other solvers for comparison

We use an open source package called dynesty, developed in [123], for performing nested sampling in line 12 of Algorithm 2. A vanilla sampler based on nested sampling was also implemented to justify the advantage of exploiting factor graph structure. In the vanilla sampler, all SLAM factors are incorporated into the likelihood model for nested sampling while predetermined uniform distributions of all variables are supplied as the prior distribution. This vanilla sampler is denoted as NS(UnifPr) in Section 3.5 for comparison.

Two other state-of-the-art stochastic inference methods, NUTS and SMC, a state-of-the-art Gaussian SLAM solver, GTSAM [28], and a non-Gaussian SLAM solver, NF-iSAM [64], are tested in our experiments as well. We supply our SLAM factors to the NUTS and SMC implementations in PyMC3 [113], GTSAM, and NF-iSAM to solve our SLAM problems. We used the default built-in initialization functions in PyMC3 for NUTS and provided a predetermined uniform distribution that covers the space of interest to SMC. The C++ library of GTSAM was used while all other techniques were implemented in Python. All computation was run on an AMD Ryzen ThreadRipper 3970X processor with 32 cores.

3.5 Results

NSFG is evaluated across four simulated and one real-world dataset to observe different aspects of its performance and capability. The datasets are: pose-graph SLAM (Section 3.5.1), range-only SLAM (Section 3.5.2), sensor network localization (Section 3.5.3), ambiguous data association (Section 3.5.4), and the Plaza1 dataset (Sec. 3.5.5). We emphasize that NSFG pursues high-fidelity samples of the posterior at the cost of computational complexity. While examples in this work are small-scale, they possess abundant non-Gaussian features in posteriors, warranting their potential as canonical examples for comparing algorithms in this work and validating efficient non-Gaussian inference techniques in the future.

Qualitative evaluation is performed by plotting the samples drawn by each inference algorithm. Points of different colors indicate different variables. We choose to compare the empirical mean, rather than the MAP point, of the samples with the ground truth to compute the RMSE. This choice was made for two reasons: (i) the MAP point among samples can be very random in posteriors with equally weighted modes, and (ii) the MAP point does not reflect secondary modes when gauging distributional errors.

3.5.1 Pose graphs

To test the simplest scenario in which all methods would be expected to work, we first evaluate 2D pose graphs. In Fig. 3-2 we show the results on one such problem. Samples of the GTSAM solution are drawn from the Laplace approximation provided by GTSAM. Estimated distributions by different methods were qualitatively similar with the exception of some spurious modes found in the NUTS solution.

Fig. 3-3 indicates that estimates made by the different samplers indeed converge as more samples are drawn. As a result of spurious modes shown in Fig. 3-2, the estimates by NUTS converge to different values from those by the other solvers. It is worth noting that estimated standard deviations of the x coordinate of $X4$ by NSFG, SMC, NS(UnifPr), and NFiSAM converge to roughly the same value, but visibly differ from the GTSAM estimate. This difference is reasonable since the standard deviation estimated by GTSAM is a local Gaussian approximation at the MAP point of this non-Gaussian posterior.

For quantitative evaluation, Fig. 3-4a shows the root mean squared error (RMSE) and runtime of solutions for 10 randomly generated pose graphs with 10 poses. The runtime and accuracy of GTSAM are not plotted in Fig. 3-4a as GTSAM outperformed the RMSE and runtime of other approaches by several orders of magnitude on these pose graphs. Since the posteriors in these pose graphs are expected to be unimodal, a large RMSE in the figure is intended to indicate samples from spurious modes. Compared with the vanilla sampler relying on uniform prior distributions, NS(UnifPr), both accuracy and runtime performances are improved in NSFG. Notably, NSFG appears faster by a factor of 3-4. These results suggest the benefits of

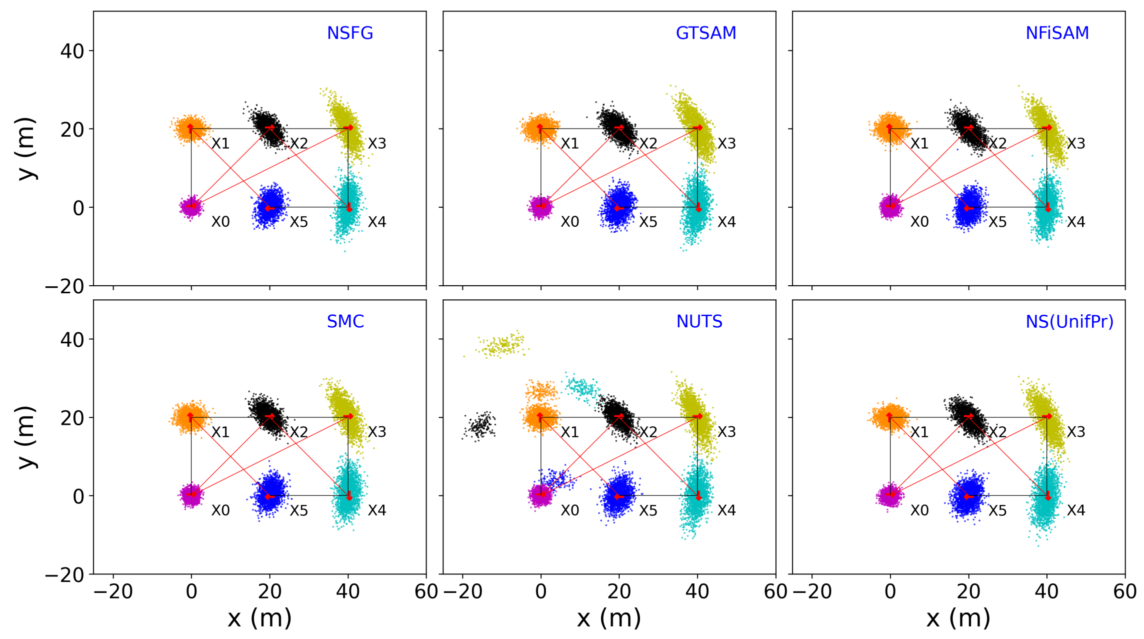


Figure 3-2: Samples that represent estimated posterior distributions for a pose-graph with 6 poses (X0-X5) and 4 loop-closures. The robot starts in the lower left and travels counter-clockwise. The groundtruth poses are marked by arrows. The black lines indicate odometry measurements between successive poses. The red lines indicate loop closures. Samples are marked by colored dots, with different colors indicating different pose variables.

supplying an informative prior model to nested sampling. The narrow error bands for NSFG suggest that NSFG is more robust than other samplers.

3.5.2 Range-only SLAM

We evaluate range-only SLAM problems to explore performance on a simple, well-understood problem with non-Gaussian, multi-modal posterior distributions. We show a single result from the range-only SLAM problems in Fig. 3-5. In the first three time steps (X0, X1, X2) the robot moves along a line, as such the posterior distribution of the landmark position consists of two distinct modes mirrored across the line driven by the robot. At the final time step (X3) the robot breaks away from the line, disambiguating the landmark position and causing the posterior distribution to converge to a single mode around the true landmark position. Qualitative evaluation of the solutions shows that NSFG best matches the expected posterior distribution for

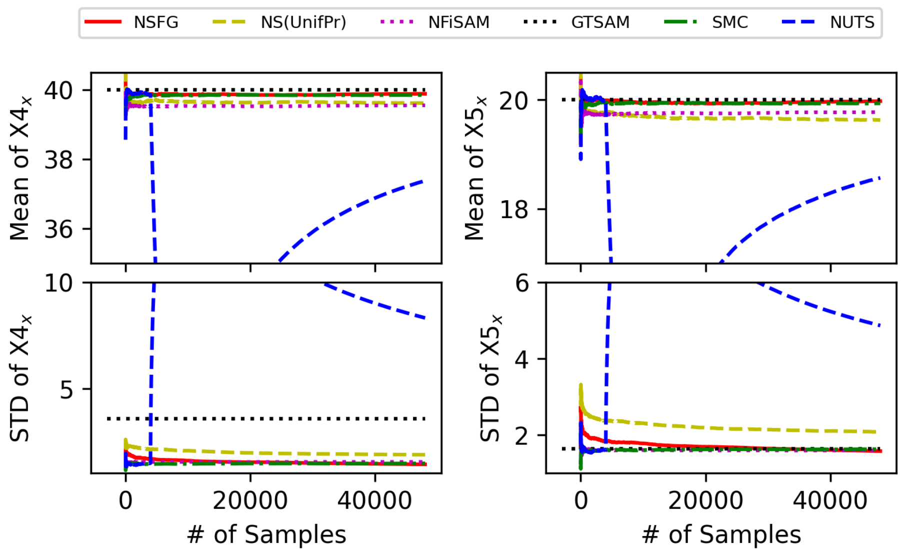


Figure 3-3: Empirical mean and standard deviation over samples of x coordinates of selected variables ($X4$ and $X5$) in the pose graph example in Fig. 3-2. Note that samples generated in the burn-in and tuning stages of NUTS are discarded.

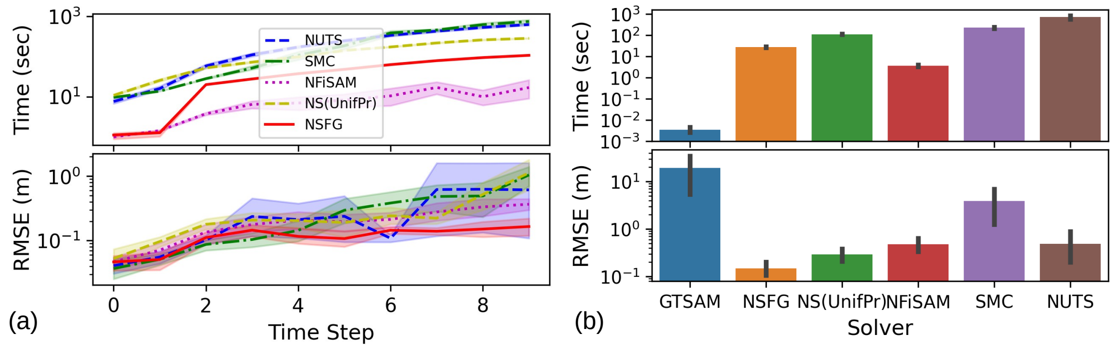


Figure 3-4: Performance of different approaches in two experiments: (a) 10 randomly generated pose graphs with 30 dimensions each (10 poses), and (b) 10 randomly generated range-only SLAM problems with 14 dimensions (4 poses and 1 landmark position). Shaded areas in plot (a) and error bars in plot (b) indicate the 95% confidence interval.

all time steps. This strongly suggests that NSFG can best approximate the posterior distribution of range-only SLAM problems.

In Fig. 3-4b we quantitatively compare all solvers on 10 other randomly generated range-only experiments. The statistics presented were computed at the final time step of each experiment after the distribution becomes unimodal (similar to time step 3 in Fig. 3-5). NSFG enjoys the lowest RMSE across different approaches. Note that in

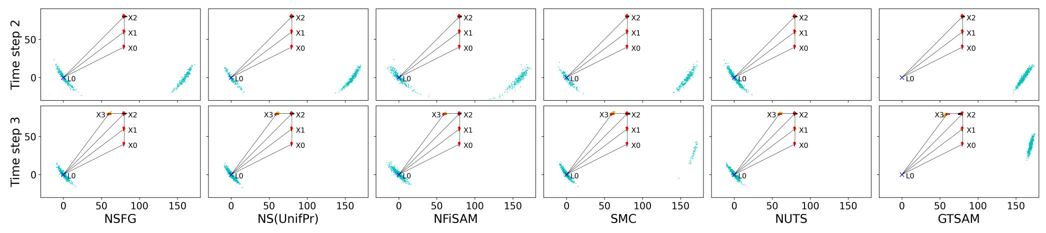


Figure 3-5: Solutions by different inference methods for a range-only SLAM problem. The red arrows indicate the groundtruth robot pose and the blue points indicate samples of the estimated landmark state. The red and black lines denote range and odometry measurements, respectively. For the GTSAM solution, the initial value of the landmark $L0$ is randomly picked on a circle that centers around the pose $X0$.

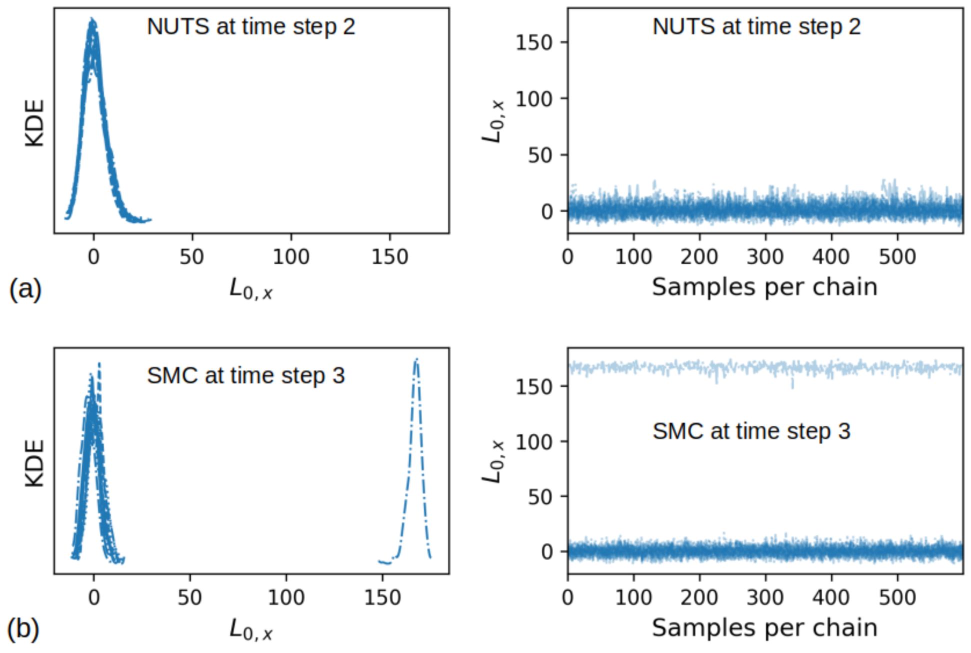


Figure 3-6: Traces of the x-coordinate values of the landmark explored by the MCMC chains in Fig. 3-5 and kernel density estimation performed on the final distributions obtained from these MCMC chains. (a) NUTS solution for the baseline range-only problem at time step 2 and (b) SMC solution for the baseline range-only problem at time step 3.

both pose-graph and range-only SLAM experiments, NSFG presents advantages over the other sampling techniques for accuracy and runtime (faster by over an order of magnitude in Fig. 3-4).

To explain the errors in SMC and NUTS, in Fig. 3-6 we display diagnostics explaining the issues observed in Fig. 3-5. As observed in the two plots on the right of Fig. 3-6, the MCMC chains are effectively stuck in local optima of the posterior. These local optima prevent the MCMC chains from mixing between the two modes of the distribution, leading to an incorrect estimate of weights over different modes. In the case of the NUTS solution at time step 2 (Fig. 3-5), all of the MCMC chains are stuck around a single mode, and are thus prevented from exploring and sampling the equally probable mirrored solution. In the case of the SMC solution at time step 3 (Fig. 3-5), since there is a local optimum around the spurious landmark position $L_{0,x} = 160$ which some MCMC chains cannot escape, samples from these chains incorrectly stress and overestimate the mode around the spurious landmark position.

3.5.3 Sensor network localization

To evaluate NSFG on a well-known non-Gaussian inference problem, we tested against the sensor network localization problem of [128]. The scenario and solution by NSFG are shown in Fig. 3-7. In brief, the inference goal is to estimate the posterior distributions of four unknown sensor locations, $\mathbf{t}_1, \dots, \mathbf{t}_4$, on the x-y plane provided two sensors with known locations, $\mathbf{t}_5$ and $\mathbf{t}_6$, a few range measurements, $\{y_{ij}\}$, and a likelihood model. The measurement likelihood between sensors i and j is modeled as

$$f_{ij}(\mathbf{t}_i, \mathbf{t}_j | y_{ij}, w_{ij}) =$$
$$\begin{cases} \exp(-\frac{\|\mathbf{t}_i - \mathbf{t}_j\|_2^2}{2\times 0.3^2}) \exp(-\frac{(y_{ij} - \|\mathbf{t}_i - \mathbf{t}_j\|_2)^2}{2\times 0.02^2}), & \text{if } w_{ij} = 1 \\ 1 - \exp(-\frac{\|\mathbf{t}_i - \mathbf{t}_j\|_2^2}{2\times 0.3^2}), & \text{otherwise.} \end{cases}$$

where w_{ij} equals 1 if there is a distance measurement between sensors i and j otherwise it is zero. Thus the posterior for this example is

$$p(\mathbf{t}_1, \dots, \mathbf{t}_4 | \mathbf{y}, \mathbf{w}) \propto \prod_{\substack{i=2,\dots,6 \\ j=1,\dots,4 \\ i>j}} f_{ij}(\mathbf{t}_i, \mathbf{t}_j | y_{ij}, w_{ij}).$$

The scatter plots, histograms, and kernel density estimation in Fig. 3-7a and b highly resemble those in [128], further justifying the NSFG's ability to represent highly

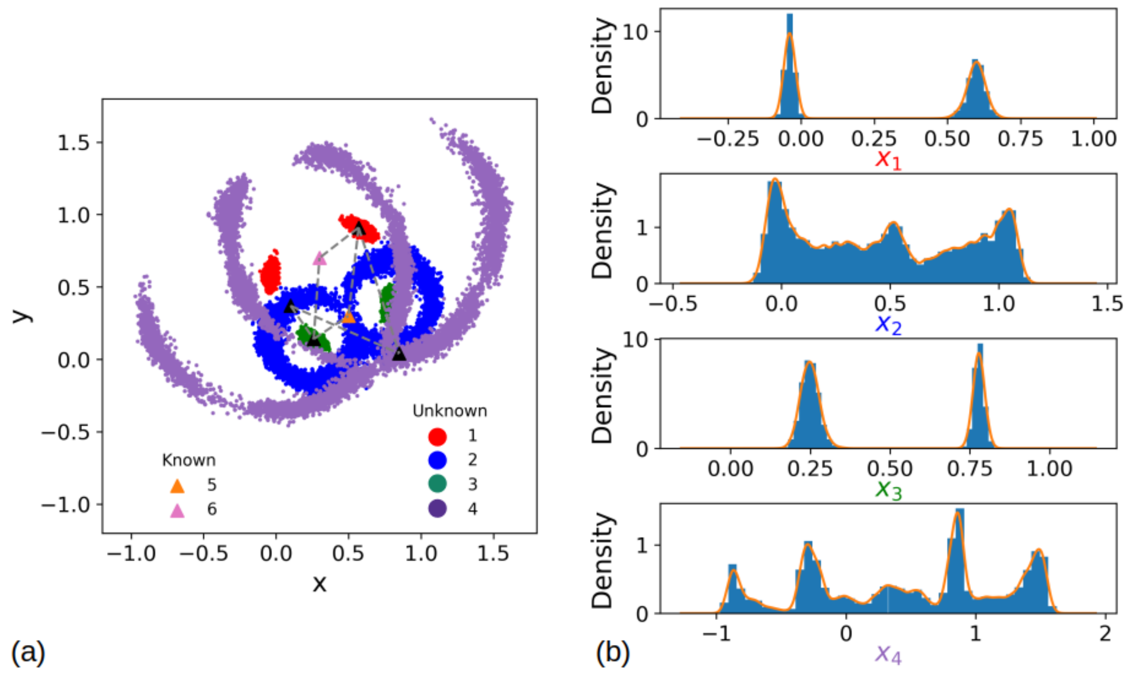


Figure 3-7: The sensor network localization example in [128]: samples and corresponding histograms and kernel density estimation drawn by NSFG. The four black triangular markers designate the groundtruth locations of four sensors whose positions are initially unknown. The two colored triangles designate the locations of two sensors at known locations. The dashed lines indicate range measurements from the various sensors to one another.

non-Gaussian posteriors.

3.5.4 Range-only SLAM with ambiguous data association

In Figs. 3-8 and 3-9, we evaluate the performance of NSFG on a range-only SLAM problem with data association ambiguity. At time step 0, range measurements to two beacons are acquired with the identify information of beacons; from pose $X1$ to $X4$, however, the landmark association is ambiguous (i.e. the robot is unsure of which landmark the distance measurement goes to). This class of problems was chosen as the posterior distribution is highly complex and demonstrates that NSFG can solve mixed continuous-discrete inference problems. In ground truth, $L1$ is observed by $X1$ and $X2$ while $X3$ and $X4$ spot $L2$.

As seen in Fig. 3-8, NSFG displays the posterior distribution that arises from this

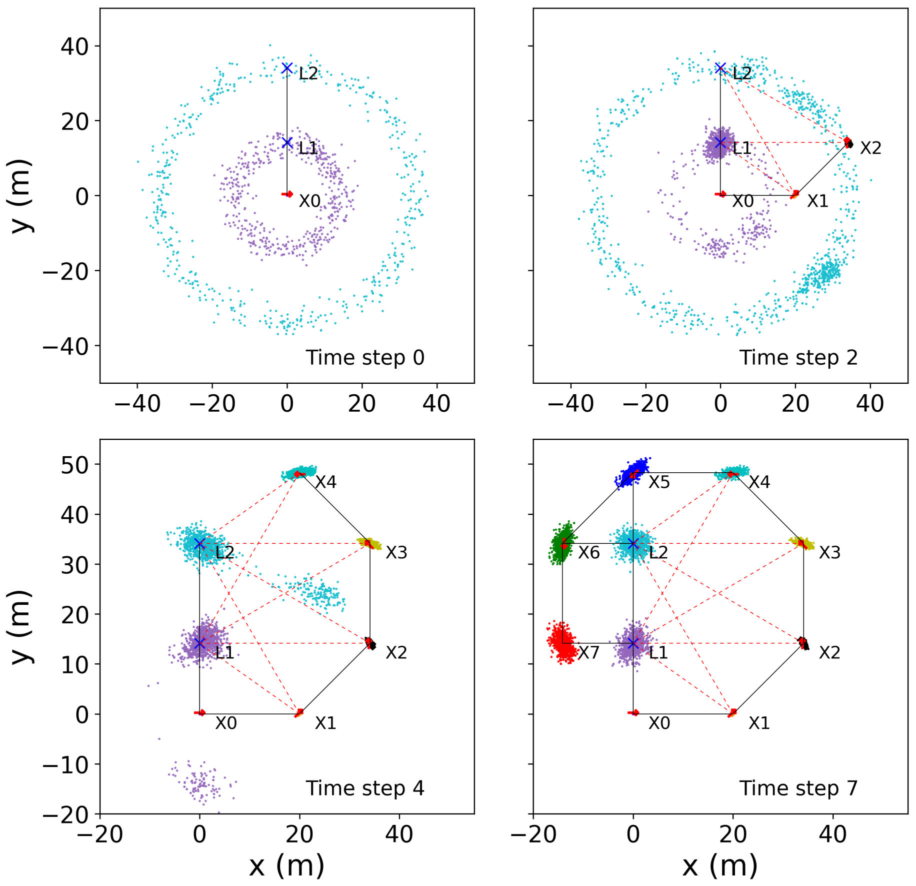


Figure 3-8: Scatter plots for posteriors of a range-only SLAM problem with data association ambiguity using NSFG. The dashed red lines from the same robot pose (X) to multiple landmarks (L) denote a range measurement that can be associated with all these landmarks. Black lines denote measurements with known association.

situation and is capable of disambiguating the landmark locations by time step 7.

Alternatively, the posterior with data-association ambiguity can be represented with the following mixture model:

$$p(\mathbf{\Theta}|\mathbf{z}) = \sum_{i=1}^{|\mathcal{D}|} w_i p(\mathbf{\Theta}|z, d_i), \tag{3.6}$$

$$w_i = \frac{p(z|d_i)p(d_i)}{\sum_{i=1}^{|\mathcal{D}|} p(\mathbf{z}|d_i)p(d_i)}, \tag{3.7}$$

where $\mathcal{D}$ denotes the set all data association combinations. Each mixture component stems from one of the data association hypotheses, i.e., $p(\mathbf{\Theta}|z, d_i)$.

Fixing the data association for a given combination, d_i, results in a new posterior,

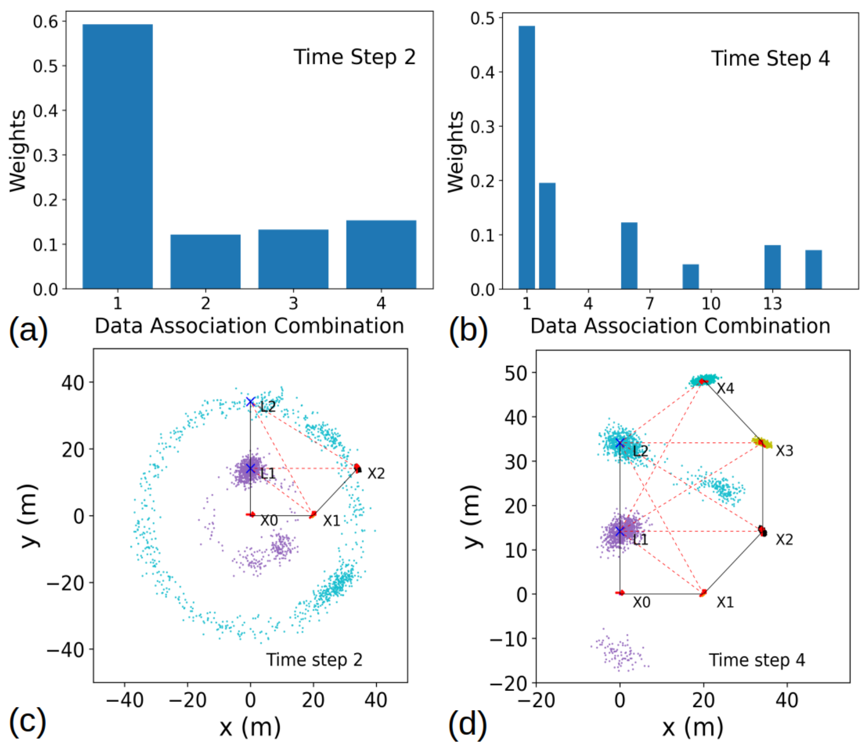


Figure 3-9: Weights and posteriors under different data associations. (a) and (b) are estimated weights of data association hypotheses at time step 2 and 4 respectively. (c) and (d) are posterior samples formed by combining samples from individual components in Eq. (3.6).

$p(\boldsymbol{\Theta}|z, d_i)$. For the new posterior, NSFG can draw samples and estimate the evidence, $p(\mathbf{z}|d_i)$. As there is no prior on data associations, $p(d_i)$ is assumed to be $\frac{1}{|\mathcal{D}|}$. Thus, the weights of components in (3.6), w_i, can be computed if we apply NSFG to solve factor graphs resulted from all combinations of data association (Fig. 3-9, top). A new ensemble of samples representing the joint posterior can be formed by performing re-sampling over the samples and weights for different data associations, as seen in the bottom of Fig. 3-9. These scatter plots resemble their counterparts in Fig. 3-8 well. Effectively, this demonstrates that NSFG is self-consistent and can be used in multiple ways to reliably approximate complex posterior distributions.

3.5.5 Real-world dataset

We apply NSFG to solve early time steps in the Plaza1 dataset [33]. The dataset provides a sequence of timestamped odometry and range measurements collected by a mobile robot in a planar environment. The ranges were measured between the robot and four landmarks using ultra-wideband sensors for which a noise model can be found in [61]. We use measurements in the early stage of the sequence to create a range-only SLAM factor graph. The factor graph involves 29 robot poses and 4 landmark locations, thus leading to a 95-dimensional posterior distribution at the final time step. The posterior distribution contains one prior factor, twenty-eight odometry factors, and thirty-three range factors. Additionally, we generate three more synthetic problems from this dataset which associate 20%, 40%, and 60% of range measurements with all landmarks. Such a range measurement with ambiguous data association (ADA) is modeled as a sum-mixture (3.5). These mixture factors are expected to incur a higher function evaluation cost than binary factors (a mixture factor entails evaluating 4 binary factors, given 4 landmarks here). We use these synthetic problems to investigate the impact of complex likelihood factor sets on the performance of NSFG.

Fig. 3-10a shows samples drawn by NSFG at time steps 21 and 24. Given data association (0% ADA), we estimate that the posterior at time step 21 is dominated by two modes, which differ in the location of landmark $L3$; at time step 24, the posterior becomes unimodal. In contrast, the belief of landmark $L3$ becomes much more uncertain in the 60% ADA case. The RMSE in Fig. 3-10b supports this claim of distributional uncertainty. The transition from multimodal to unimodal posteriors sharply decreases the RMSE in all cases. However, such sharp reduction occurs to the 60% ADA case later than other cases. In the end of the sequence, the RMSE of 60% ADA converges to a low value, indicating the recovery of the groundtruth mode. As seen in Fig. 3-10b, more ADA factors incur greater computational cost, but these factors do not alter the scaling behavior of runtime with respect to dimensions. Note that later time steps entail more poses and greater dimensionality. In addition, we

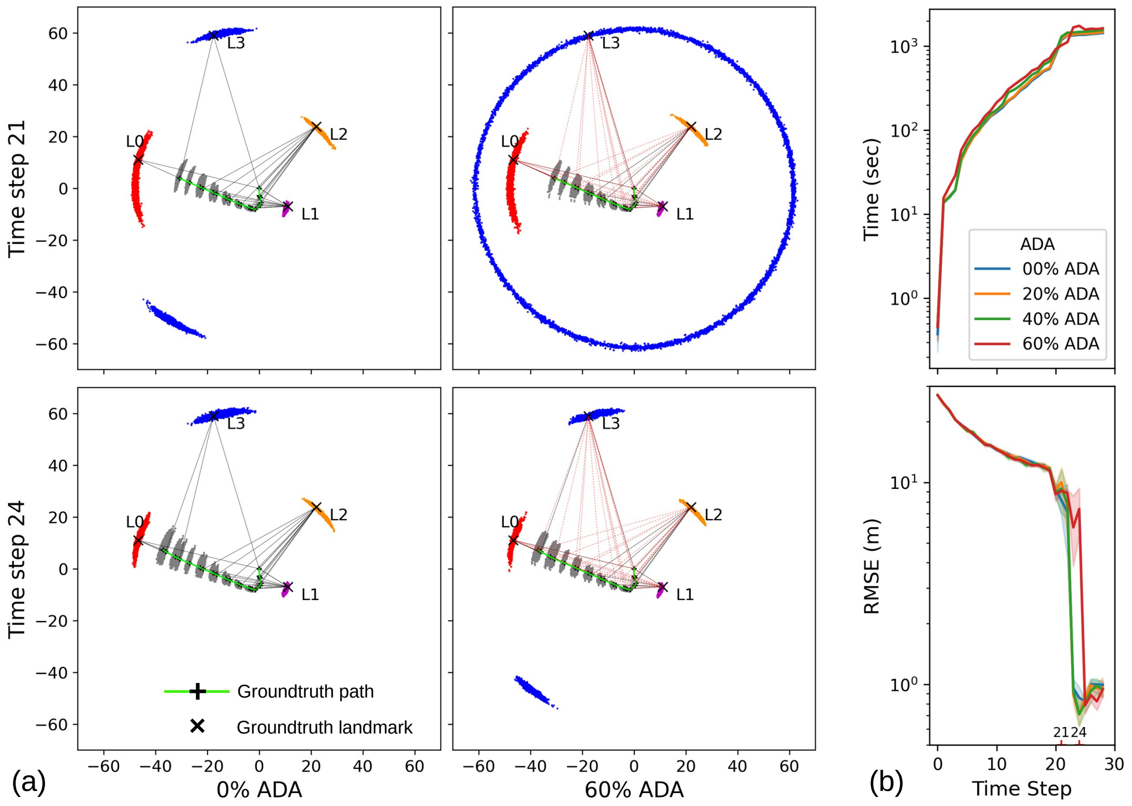


Figure 3-10: Plaza1 dataset with different fractions of ADA: (a) posterior samples and (b) runtime and error. In plot (a), black lines indicate range measurements with known data association. Red lines from a robot pose indicate ambiguous range measurements. In (b), the shaded area indicates the 95% confidence interval estimated by six runs with different random seeds.

solve each of the factor graphs six times using different random seeds. The narrow error bands in Fig. 3-10b indicate that random seeds have little effect on NSFG.

We note that NSFG confirms that the joint posterior of landmarks and the robot path becomes peaked at a single point within the early time steps solved here. For the subsequent time steps in the dataset, one can confidently apply the MAP estimation and (unimodal) Gaussian approximation to represent the joint posterior.

3.6 Summary

We introduced nested sampling methods to directly draw samples from the posterior distributions encountered in non-Gaussian SLAM problems, pursuing the *bona fide* shape of the posterior. Leveraging the sparsity structure of SLAM factor graphs, our proposed approach, NSFG, provides nested sampling with informative prior distributions which can be efficiently sampled, leading to computational benefits for nested sampling methods.

We have demonstrated the advantage of NSFG over other sampling techniques and state-of-the-art Gaussian/non-Gaussian SLAM algorithms. NSFG presents superior robustness in inferring posteriors than all other approaches and operates over an order of magnitude faster than other sampling techniques. Additionally, we showed that the estimate of evidence, as a unique benefit of nested sampling, can be used to compute the posterior belief of ambiguous data associations, indicating the potential of NSFG for solving mixed continuous-discrete inference problems. Lastly, the performance of NSFG was demonstrated to be consistent under various conditions such as dimensionality, fractions of ambiguous data associations, and random seeding.

We believe that NSFG can be a promising tool for providing reference solutions for the posteriors in non-Gaussian SLAM problems. These solutions can aid accuracy evaluation of approximate inference algorithms and promote deeper understanding of uncertainty propagation on cyclic non-Gaussian SLAM factor graphs. Future work includes developing goodness-of-fit criteria of posterior samples for SLAM.

Chapter 4

Real-time inference of marginal posteriors in SLAM via blending Gaussian approximation and particle filters

The content in this chapter is mainly based on the following paper:

- Qiangqiang Huang, John J. Leonard. GAPSLAM: Blending **G**aussian **A**pproximation and **P**article Filters for Real-Time Non-Gaussian **SLAM**. *IEEE/RSJ International Conference on Intelligent Robots and System (IROS)*, 2023.

4.1 Introduction

The NSFG algorithm in Chapter 3 provides high-quality samples as a reference solution for full posterior inference, but is computationally intractable for real-time applications. This chapter develops a scalable, real-time algorithm to infer marginal posteriors in SLAM. A highlight of this chapter is that, in Sec. 4.5, we describe a streaming platform that we developed to conduct live demo of object-based SLAM.

As mentioned in Chapter 1.1.2, standard real-time full posterior inference techniques, such as Gaussian approximation and particle filters, either lack expressiveness for representing non-Gaussian posteriors or suffer from performance degeneracy when estimating high-dimensional posteriors. By blending advantages of the **G**aussian **A**pproximation and **P**article filters on scalability and non-Gaussian estimation, respectively, we present a novel algorithm, **GAP**SLAM, to infer marginal posteriors encountered in SLAM. The contributions of the chapter include:

1. An adaptive modeling strategy whereby only marginals with great uncertainty are sampled by particle filters, while others are represented by the Gaussian approximation.
2. An uncertainty-aware re-initialization technique that leverages particle filters to reset linearization points in the nonlinear optimization solver.
3. Range-only and object-based bearing-only SLAM experiments that demonstrate the scalability, generalizability, and accuracy of GAPSLAM, as well as its ability to precisely describe the evolution of non-Gaussian posteriors in real-time.

4.2 Problem statement

We focus on landmark-based SLAM. Let $\mathbf{X}_t := (X_0, X_1, \dots, X_t)$ be robot pose variables up to time step t where $X_i \in \mathrm{SE}(d)$. Landmark variables are denoted by $\mathbf{L}_n := (L_1, L_2, \dots, L_n)$ where each element denotes the location or pose of a landmark (e.g., ultra-wideband tags, 3D points of visual features, and daily objects). For SLAM problems, we consider two types of measurements: i) odometry and ii) landmark measurements. Let $\mathbf{o}_t := (o_1, o_2, \dots, o_t)$ be odometry measurements of which each is modeled by the likelihood function $p(o_i|X_{i-1}, X_i)$. Without loss of generality, for the ease of notation in the method formulation, we assume the robot makes at most a landmark measurement at a time. Let $\mathbf{z}_t := (z_0, z_1, \dots, z_t)$ be all landmark measurements in which each is modeled by the likelihood function $p(z_i|X_i, L_{d_i})$, where $d_i \in \{1, 2, \dots, n\}$ is the landmark index associated with measurement z_i. Let $\mathbf{d}_t := (d_0, d_1, \dots, d_t)$ be data associations for all landmark measurements thus the posterior of robot and landmark variables at time step t can be formulated by

$$p(\mathbf{X}_t, \mathbf{L}_n | \mathbf{z}_t, \mathbf{o}_t, \mathbf{d}_t) \tag{4.1}$$

$$\overset{\text{Bayes}}{\propto} \prod_{i=0}^{t} p(z_i | X_i, L_{d_i}) \prod_{i=1}^{t} p(o_i | X_{i-1}, X_i) p(\mathbf{X}, \mathbf{L}), \tag{4.2}$$

where $p(\mathbf{X}, \mathbf{L})$ denotes the prior. *Our goal is to infer marginal posteriors.*

4.3 Approach

Our method represents marginals either by samples or parametric models, depending on the uncertainty of the marginal. This hybrid density modeling aims to achieve a balance between computational efficiency and expressiveness. Our method is inspired by a typical observation about the posterior: oftentimes, only a few of the landmarks have very uncertain marginal posteriors, while marginals of most variables can be well described by Gaussian distributions. For example, a noisy range-only *or* bearing-only measurement between a robot pose and a landmark can simulate either spherical or conic landmark distributions in 3D, while accumulated measurements may well constrain the landmark to a concentrated and (almost) unimodal distribution.

Specifically, we use particle filters to estimate very uncertain landmarks for the expressiveness of sample-based density representations. Meanwhile, we maintain a Gaussian approximation of the posterior for computation efficiency. Fig. 4-1 illustrates our method. Combining these two techniques leads to a win-win situation over time: i) the Gaussian approximation provides particle filters with a parametric model of the smoothed robot path, reducing the sampling complexity to landmarks, and ii) particle filters afford the Gaussian approximation statistically probable values of landmarks, which can be used to explicitly reset linearization points in nonlinear optimization solvers, thereby benefiting the pursuit of global optima. We describe the details of our method in the following sections.

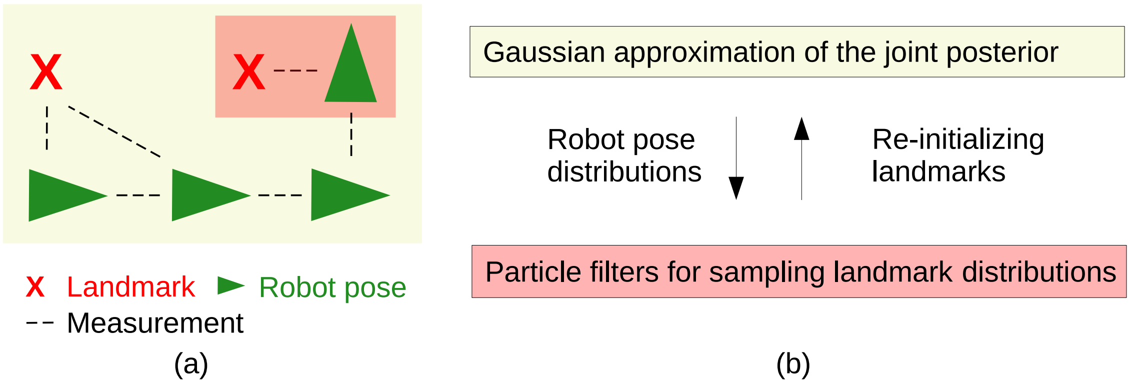

X Landmark
Robot pose
-- Measurement
(a)
Gaussian approximation of the joint posterior
Robot pose distributions
Re-initializing landmarks
Particle filters for sampling landmark distributions
(b)

Figure 4-1: Illustration of the GAPSLAM algorithm: (a) a SLAM example, where the robot moves along poses in green and makes measurements to landmarks in red, and (b) our method, which blends Gaussain approximation in yellow and particle filters in pink. The Gaussian approximation, centered on the maximum a posteriori (MAP) estimate, provides robot pose distributions on which the particle filters are conditioned to draw samples that represent landmark distributions. If a sample attains a higher probability than the MAP estimate when evaluating the posterior, landmarks in the Gaussian solver will be re-initialized by that sample.

4.3.1 Gaussian approximation

We denote the Gaussian approximation (GA) of the posterior at time step t as $g(\mathbf{X}_t, \mathbf{L}_n|\mathbf{z}_t, \mathbf{o}_t, \mathbf{d}_t)$. We categorize all landmarks into a set of non-Gaussian landmarks $L_{\mathcal{N}_t}$ and a set of Gaussian landmarks $L_{\mathcal{G}_t} = \mathbf{L}_n \setminus L_{\mathcal{N}_t}$, where $\mathcal{N}_t \cup \mathcal{G}_t = \{1, 2, \dots, n\}$ and $\mathcal{N}_t \cap \mathcal{G}_t = \emptyset$ (see Sec. 4.3.3 for the approach to identify Gaussian landmarks). We use the Gaussian approximation to represent the posterior of robot poses and the Gaussian landmarks. That is,

$$
\begin{aligned}
&p(\mathbf{X}_t, \mathbf{L}_n|\mathbf{z}_t, \mathbf{o}_t, \mathbf{d}_t) && (4.3)\\
&\overset{\text{GA}}{\approx} p(L_{\mathcal{N}_t}|\mathbf{X}_t, L_{\mathcal{G}_t}, \mathbf{z}_t, \mathbf{o}_t, \mathbf{d}_t) g(\mathbf{X}_t, L_{\mathcal{G}_t}|\mathbf{z}_t, \mathbf{o}_t, \mathbf{d}_t), && (4.4)\\
&\overset{\text{CIR}}{=} p(L_{\mathcal{N}_t}|\mathbf{X}_t, \mathbf{z}_t, \mathbf{o}_t, \mathbf{d}_t) g(\mathbf{X}_t, L_{\mathcal{G}_t}|\mathbf{z}_t, \mathbf{o}_t, \mathbf{d}_t), && (4.5)\\
&\overset{\text{CIR}}{=} \prod_{i \in \mathcal{N}_t} p(L_i|X_{\mathcal{V}_i}, z_{\mathcal{V}_i}, d_{\mathcal{V}_i}) g(\mathbf{X}_t, L_{\mathcal{G}_t}|\mathbf{z}_t, \mathbf{o}_t, \mathbf{d}_t) && (4.6)
\end{aligned}
$$

where $X_{\mathcal{V}_i}$ denotes the set of robot poses that observed landmark L_i (i.e., $\mathcal{V}_i = \{j \in [0, t] | d_j = i\}$ denotes the time steps observing landmark L_i). The reduction from (4.4) to (4.5) exploits the conditional independence relation (CIR) that landmarks are conditionally independent given a robot path. Note that while we use the CIR in a similar way to typical Rao-Blackwellized particle filters (RBPFs) for SLAM (e.g., FastSLAM 2.0 [91]), our probabilistic modeling and inference methods differ from these RBPFs. The Gaussian approximation can be computed via the Laplace approximation [10]: 1) finding the MAP estimate as the mean and 2) estimating the Hessian of the negative logarithm of the posterior at the MAP estimate as the covariance. In practice, many nonlinear local optimization solvers, such as GTSAM, are capable of providing efficient out-of-the-box solutions to the Gaussian approximation, although they are subject to linearization errors and local optima in the MAP estimate. In this paper, we refer to them as Gaussian solvers.

4.3.2 Marginal posterior of non-Gaussian landmarks

We will use particles to represent the marginal posterior of any non-Gaussian landmark $L_i \in L_{\mathcal{N}_t}$. Alg. 3 summarizes how we draw the particles. Integrating out all variables except L_i in (4.6), we can formulate the marginal posterior by

$$\begin{aligned}
&p(L_i|\mathbf{z}_t, \mathbf{o}_t, \mathbf{d}_t) && (4.7)\\
&\overset{\text{GA}}{\approx} \int_{X_{\mathcal{V}_i}} p(L_i|X_{\mathcal{V}_i}, z_{\mathcal{V}_i}, d_{\mathcal{V}_i}) g(X_{\mathcal{V}_i}|\mathbf{z}_t, \mathbf{o}_t, \mathbf{d}_t) && (4.8)\\
&\overset{\text{MC}}{\approx} \frac{1}{K}\sum_{k=1}^{K} p(L_i|X_{\mathcal{V}_i} = x_{\mathcal{V}_i}^{(k)}, z_{\mathcal{V}_i}, d_{\mathcal{V}_i}), && (4.9)
\end{aligned}$$

where $\{x_{\mathcal{V}_i}^{(k)}\}_{k=1}^{K}$ are K i.i.d. samples of part of the robot path drawn from the Gaussian marginal $g(X_{\mathcal{V}_i}|\mathbf{z}_t, \mathbf{o}_t, \mathbf{d}_t)$ (line 1 in Alg. 3). Monte Carlo (MC) integration is applied in (4.9). The conditional of L_i in (4.9) is a result of eliminating L_i from factors adjacent to L_i (line 2 in Alg. 3), thus

$$p(L_i|X_{\mathcal{V}_i}, z_{\mathcal{V}_i}, d_{\mathcal{V}_i}) = \frac{\psi(L_i, X_{\mathcal{V}_i})}{\int_{L_i} \psi(L_i, X_{\mathcal{V}_i})}, \tag{4.10}$$

where

$$\psi(L_i, X_{\mathcal{V}_i}) = \prod_{j \in \mathcal{V}_i} p(z_j | X_j, L_i) p(L_i). \tag{4.11}$$

We use importance sampling to draw samples representing $p(L_i | X_{\mathcal{V}_i} = x_{\mathcal{V}_i}^{(k)}, z_{\mathcal{V}_i}, d_{\mathcal{V}_i})$. We design the proposal distribution of landmark L_i as the sum-mixture of binary factors, as seen in

$$q(L_i; x_{\mathcal{V}_i}^{(k)}) = \frac{1}{|\mathcal{V}_i|} \sum_{j \in \mathcal{V}_i} p(L_i | X_j = x_j^{(k)}, z_j). \tag{4.12}$$

This proposal[1] was chosen for two reasons: i) covering more area in the space of the landmark variable and ii) it is efficient to draw landmark samples $\{l_i^{(m,k)}\}_{m=1}^{M}$ from the proposal $q(L_i)$ by sampling a multinomial distribution and binary factors independently. With proposal samples $\{l_i^{(m,k)}\}_{m=1}^{M}$ in hand, the normalized weight of each sample can be computed by

$$w^{(m,k)} \propto \frac{p(l_i^{(m,k)} | x_{\mathcal{V}_i}^{(k)}, z_{\mathcal{V}_i}, d_{\mathcal{V}_i})}{q(l_i^{(m,k)}; x_{\mathcal{V}_i}^{(k)})} \propto \frac{\psi(l_i^{(m,k)}, x_{\mathcal{V}_i}^{(k)})}{q(l_i^{(m,k)}; x_{\mathcal{V}_i}^{(k)})}, \tag{4.13}$$

where $\sum_{m=1}^{M} w^{(m,k)} = 1$ (line 8 in Alg. 3). Finally we apply the re-sampling and regularization steps in [4, Alg. 6] to generate equally weighted samples of L_i (line 9 in Alg. 3).

Thus the marginal posterior in (4.9) can be approximated by samples, as seen in

$$p(L_i | \mathbf{z}_t, \mathbf{o}_t, \mathbf{d}_t) \overset{\text{MC}}{\approx} \tilde{p}(L_i | \mathbf{z}_t, \mathbf{o}_t, \mathbf{d}_t) \tag{4.14}$$

$$= \frac{1}{K} \sum_{k=1}^{K} \sum_{m=1}^{M} \delta(L_i - l_i^{(m,k)}). \tag{4.15}$$

To summarize, for each of the K robot path samples, we draw M landmark samples. The delta functions (or other kernels) of all KM landmark samples can be used to

[1]Although we sacrificed sample diversity, we adopted a lighter-weight proposal in our experiments to ensure computational efficiency. Specifically, we limited the number of components in the sum-mixture to 5 and chose these components randomly.

Algorithm 3: LandmarkSampler

```
   Input: Landmark $l$, Gaussian solver $g$
1  Draw robot path samples $\mathcal{X}$ from the Gaussian approx.
2  $\mathcal{F} = \{f(l, \mathbf{x})\} \leftarrow$ All factors adjacent to $l$          // robot vars $\mathbf{x}$
3  $\psi(l, \mathbf{x}) \leftarrow$ Product of all factors in $\mathcal{F}$          // potential func. (4.11)
4  $q(l, \mathbf{x}) \leftarrow$ Sum of elements in $\mathcal{F}$          // proposal density (4.12)
5  Initialize a set $\mathcal{S}$ for storing landmark samples
6  for sample $\hat{\mathbf{x}}$ in $\mathcal{X}$ do
7      Draw landmark samples $\mathcal{L}$ from density $q(l, \mathbf{x} = \hat{\mathbf{x}})$
8      Weights $\mathcal{W}$ of landmark samples by computing $\psi/q$
9      $\mathcal{L} \leftarrow$ Resampling using $\mathcal{L}$ and $\mathcal{W}$
10     Add $\mathcal{L}$ to $\mathcal{S}$
11 return $\mathcal{S}$          // samples of $l$
```

Algorithm 4: GAPSLAM

```
   Input: New factor $f$ of landmark $l$, Gaussian solver $g$, dictionary $\mathcal{D}$ of
          non-Gaussian landmark samples
1  if l not in the Gaussian solver then          // new landmark
2      Add $l$ to non-Gaussian landmarks $\mathcal{D}$ as a new key
3  Add $f$ to the Gaussian solver
4  if $l$ in non-Gaussian landmarks $\mathcal{D}$ then
5      LandmarkReinitializer($l$, $g$, $\mathcal{D}[l]$)          // re-init. in the solver
6  Update the MAP estimate in the Gaussian solver
7  if $l$ in non-Gaussian landmarks $\mathcal{D}$ then
8      $\mathcal{D}[l]$ = LandmarkSampler($l$, $g$)          // update samples
9      Compute empirical covariance matrix $C$ of $\mathcal{D}[l]$
10     if the largest eigenvalue of $C$ is small then
11         Delete $l$ from $\mathcal{D}$          // no longer non-Gaussian landmarks
12 return $g$ and $\mathcal{D}$          // updated solver and dictionary
```

Algorithm 5: LandmarkReinitializer

```
   Input: Landmark $l$, Gaussian solver $g$, samples $\mathcal{S}$ of landmark $l$
1  $\psi(l, \mathbf{x}) \leftarrow$ Product of all factors adjacent to $l$          // robot vars $\mathbf{x}$
2  $\hat{l}, \hat{\mathbf{x}} \leftarrow$ Current estimate in the Gaussian solver $g$
3  $l^* \leftarrow$ Value in $\{\mathcal{S}, \hat{l}\}$ maximizes function $\psi(l, \mathbf{x} = \hat{\mathbf{x}})$
4  Reset the value of $l$ in the Gaussian solver to $l^*$
5  return $g$          // updated solver
```

approximate the marginal posterior of the landmark.

4.3.3 Updates with new odometry and landmark measurements

The previous sections describe how the Gaussian approximation helps compute the particle-based belief of landmarks. Now we discuss how these particles provide the Gaussian approximation with linearization points, thus leading to the uncertainty-aware re-initialization in the Gaussian approximation. We will begin by analyzing the incremental update with one time step.

From time step t to time step $t+1$, the robot receives a new odometry reading o_{t+1} and a new landmark measurement z_{t+1}. It is easy to create a new robot pose X_{t+1} and absorb the odometry o_{t+1} to form the intermediate Gaussian $g(\mathbf{X}_{t+1}, \mathbf{L}_n | \mathbf{z}_t, \mathbf{o}_{t+1}, \mathbf{d}_t)$. However, fusing the measurement z_{t+1} is a more involved process. Alg. 4 summarizes steps for tackling the landmark measurement. Assuming that the measurement z_{t+1} comes from a landmark with index d_{t+1}[2], several cases are considered:

1. If the observed landmark $L_{d_{t+1}}$ is not new or a non-Gaussian landmark (lines 3 and 6 in Alg. 4), we simply add the likelihood model of z_{t+1} (i.e., factor in the algorithm) to the Gaussian solver to update the Gaussian approximation without making any further changes.

2. If the observed landmark $L_{d_{t+1}}$ is new (line 2 in Alg. 4), i.e. $d_{t+1} = n + 1$, we update the landmark set to $\mathbf{L}_{n+1} = \mathbf{L}_n \cup \{L_{n+1}\}$ and indices of non-Gaussian landmarks to $\mathcal{N}_{t+1} = \mathcal{N}_t \cup \{n + 1\}$. We then add the likelihood model of measurement z_{t+1} to the Gaussian solver. Finally, we can simulate equally weighted samples of the landmark using this single measurement and randomly select a point from the samples as the initial guess of the landmark.

3. If the observed landmark $L_{d_{t+1}}$ is not new and is in non-Gaussian landmarks $L_{\mathcal{N}_t}$, this is the challenging case we will focus on.

In the third case, our goal is to use the new measurement z_{t+1} to determine whether we should explicitly re-initialize the non-Gaussian landmark in the Gaussian

[2]Data association and creation of new landmarks are left to the specific application in Sec. 4.4.2.

solver (line 5 in Alg. 4). Furthermore, if our belief of the landmark converges to a less uncertain situation after fusing measurement z_{t+1}, we remove the landmark from the non-Gaussian set $L_{\mathcal{N}_t}$ (line 11 in Alg. 4).

Algorithm 5 summarizes how we re-initialize a non-Gaussian landmark L_i. As described in Sec. 4.3.2 and Alg. 3, we already have samples of the landmark at hand, which can be passed to Alg. 5 as input. The union of the samples and the current estimate of the landmark forms a set of candidate points for re-initialization. The point in the set that maximizes the posterior will be used to re-initialize the landmark (lines 3 and 4 in Alg. 5). During the evaluation of the posterior, all variables other than the landmark are fixed by the current mean of the Gaussian approximation. This reduces the evaluation to only the product of factors that are adjacent to the landmark, simplifying the computation. The re-initialization can be formulated as

$$l_i^* = \underset{l_i \in \{\mathcal{S}, \hat{l}_i\}}{\operatorname{argmax}} \, \psi(L_i = l_i, X_{\mathcal{V}_i} = \hat{x}_{\mathcal{V}_i}), \tag{4.16}$$

where the potential $\psi(\cdot)$ is the same as (4.11), $\mathcal{S} = \{l_i^{(j)}\}_{j=1}^{M}$ denotes current samples of the landmark, and $\hat{l}_i$ and $\hat{x}_{\mathcal{V}_i}$ denote current estimates of the landmark and robot poses in the Gaussian solver.

After performing the re-initialization, we compute the Gaussian approximation for time step $t+1$ (line 6 in Alg. 4), $g(\mathbf{X}_{t+1}, \mathbf{L}_n | \mathbf{z}_{t+1}, \mathbf{o}_{t+1}, \mathbf{d}_{t+1})$, where the new measurement z_{t+1} has been incorporated. Given the new Gaussian approximation, we update samples of the marginal posterior of L_i (line 8 in Alg. 4), which are cached for potential re-initialization in the future. Following [12], we compute an empirical covariance matrix of these samples as well as its largest eigenvalue. When the largest eigenvalue falls below a threshold, it indicates that the belief of landmark L_i has become sufficiently certain, and L_i is removed from the non-Gaussian landmark set $L_{\mathcal{N}_{t+1}}$ (line 11 in Alg. 4). As a result, L_i will no longer be estimated by particle filters and will only be involved in updates in the Gaussian solver. Note that currently, we set the threshold of the eigenvalue with predetermined values. One can explore automatic approaches such as monitoring the convergence rate of the largest eigenvalue or performing normality tests on samples.

4.3.4 Summary of the approach

Algorithms 3-5 summarize our approach. Algorithm 4 is the main process where we update the Gaussian solver/approximation and a dictionary that maps non-Gaussian landmarks to their samples.

4.4 Experiments and results

We implemented Algorithms 3-5 in Python by blending our in-house code of particle filters and the Gaussian solver provided by the GTSAM library. In addition, we applied explicit regularization to the Gaussian solver by adding large-covariance priors on new non-Gaussian landmarks. These prior factors were not involved in the sampling or re-initialization and were removed once the corresponding landmarks were removed from non-Gaussian landmarks. We perform two sets of experiments: 1) we validate our solutions for distribution and point estimations using range-only SLAM experiments, and 2) we demonstrate the generalizability and scalability of our method using object-based bearing-only SLAM experiments. All experiments were conducted on a laptop with a 2.30GHz Intel Core i7-10875H CPU running Ubuntu 20.04.4 LTS.

4.4.1 Range-only SLAM

Datasets and methods for comparison

The performance of GAPSLAM is evaluated by comparing it with other methods on a range-only SLAM dataset, as range measurements can easily lead to highly non-Gaussian/multi-modal posteriors. We use the Plaza 1 dataset [33] which provides time-stamped range and odometry measurements collected by a mobile robot in a planar environment. Ranges between the robot and four landmarks were measured by an ultra-wideband ranging system so each range measurement was tagged with the identity of the landmark. Two methods were used for comparison: 1) NSFG [63], which directly draws samples from the joint posterior using nested sampling methods [120, 123] and is considered as reference solutions at the expense of computational burden, and 2) RBPF-SOG [11], which models landmark conditionals as sum-of-Gaussians (SOGs) in the RBPF framework, updates the SOGs using multi-hypothesis EKFs, and attains real-time operation. We present two sets of results: 1) marginal posteriors for demonstrating how the Gaussian approximation helps particle filters to draw landmark samples, and 2) point estimates for validating the use of particle filters in re-initializing the Gaussian approximation.

Results on marginal posteriors

We present the posteriors of early time steps, which are expected to be strongly non-Gaussian (e.g., multi-modal). Fig. 4-2 shows samples drawn by different methods to represent the posteriors. GAPSLAM shows good consistency with the reference samples by NSFG, while RBPF-SOG overestimates the uncertainty in the posteriors, as indicated by spurious modes of landmark L_3 samples (blue dots in the last column of Fig. 4-2). The excessive number of modes is an expected result of tracking multiple hypotheses in SOGs. Although we applied the strategy suggested in [11] to prune hypotheses with negligible weights, as shown by the decreasing blue line in the bottom of Fig. 4-3b, the averaged number of modes per landmark is still 11 at time step 21.

In order to explain how GAPSLAM draws the samples in Fig. 4-2 and when the re-initialization occurs, we can start by looking at the green confidence intervals, which represent the Gaussian approximation. Samples of robot positions and Gaussian landmarks are drawn from the Gaussian. At time step 21, L_0 and L_3 are non-Gaussian landmarks, and we use the method described in Sec. 4.3.2 and Algorithm 3 to draw their samples. Following (4.16), these samples are then used for re-initialization across time steps 21 and 24, which is reflected by the increased computation time for re-initialization in Fig. 4-3a. Because the set of Gaussian landmarks extends from (L_1, L_2) at time step 21 to all landmarks at time step 24, the number of non-Gaussian landmarks drops to zero, as shown in the bottom of Fig. 4-3b.

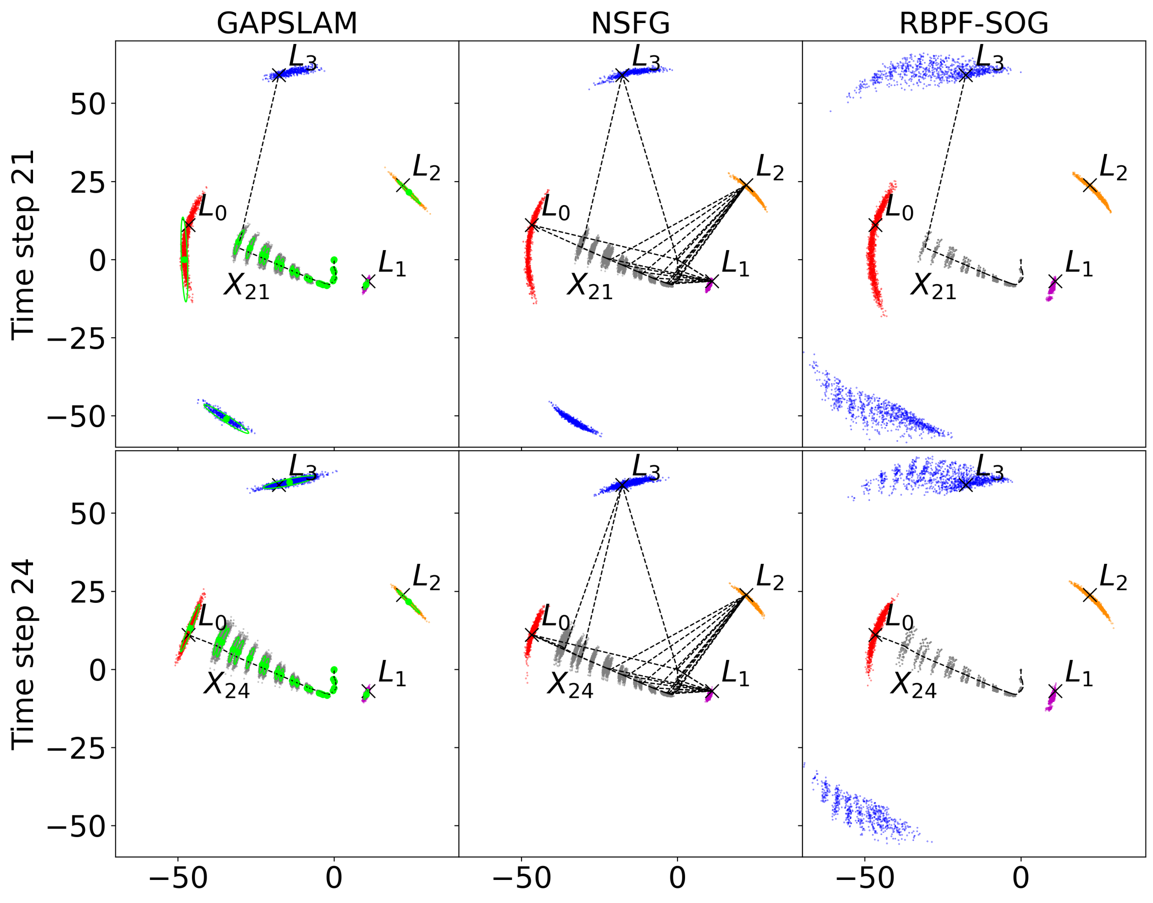


Figure 4-2: Samples from marginal posteriors in the range-only experiment. The robot moves from $(0, 0)$ to pose X_{21} or X_{24}, observing 4 landmarks L_{0-3}. Black lines and markers indicate the ground truth. Measurements are shown as dashed lines (accumulated measurements are shown in the middle column). Gray dots indicate robot position samples while colored dots denote landmark samples. Green ellipses in GAPSLAM represent confidence intervals (2σ) of the Gaussian approximation.

Fig. 4-3a illustrates the computation time of different methods for updating their probabilistic models. Note that the computation time for drawing dense samples ($K = 200, M = 100$ in (4.15)) of landmarks shown in Fig. 4-2 is separately shown in the top of Fig. 4-3b. There are two reasons for excluding it from Fig. 4-3a: 1) sampling landmark posteriors can be implemented as a parallel process, and 2) RBPF-SOG does not need landmark samples to update the SOGs, and GAPSLAM only needs sparse samples to assist the re-initialization to the Gaussian approximation. For the sparse samples (i.e., 'update samples' in Fig. 4-3a), we draw landmark samples given only the current mean of the robot path (i.e., $K = 1$ in (4.15)). While

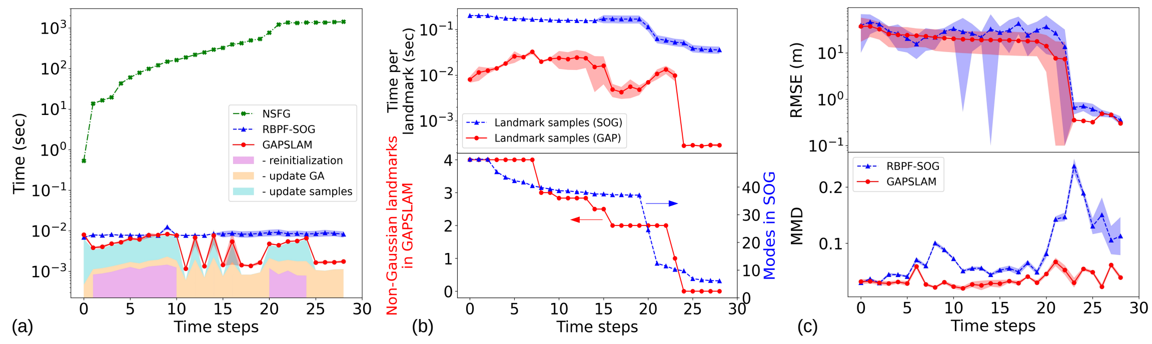


Figure 4-3: Performance comparison at early time steps in the Plaza1 dataset: (a) computational time for updating probabilistic models, (b) computational time for posterior samples of landmarks, the number of non-Gaussian landmarks in GAPSLAM, and the averaged number of modes per landmark in RBPF-SOG, and (c) the root mean square error (RMSE) of point estimates and the maximum mean discrepancy (MMD) [52] of landmark samples. The MMD is a distance between two distributions represented by samples. We compute the MMD between the reference samples and samples by GAPSLAM (or RBPF-SOG), so a lower MMD means samples that describe the posterior better. Error bands indicate the mean and standard deviation across 6 runs with different random seeds.

sampling landmark posteriors consumes slightly more computation time than updating models in GAPSLAM before time step 24, the sampling still supports an update frequency of at least 30 Hz. After time step 24, there are no non-Gaussian landmarks, so GAPSLAM is just the Gaussian solver with additional functions for sampling Gaussian marginals, which enjoys significantly faster speeds.

Fig. 4-3c presents the performance evaluation of point estimates and distribution estimation using the root mean square error (RMSE) and maximum mean discrepancy (MMD) [52], respectively. We compute the MMD between samples by GAPSLAM (or RBPF-SOG) and the reference samples by NSFG. The MMD is a distance between two distributions represented by samples, so a lower MMD here indicates a more accurate set of samples for representing posteriors. The results show that GAPSLAM significantly outperforms RBPF-SOG in terms of MMD, especially after time step 21 when the spurious modes in SOGs become more prominent, providing quantitative evidence of the superior accuracy of GAPSLAM in sampling posteriors. Additionally, it is worth noting that the RMSE of GAPSLAM only slightly outperforms that of RBPF-SOG at these early time steps. This indicates that MMD is a more suitable

Table 4.1: Point estimates for the full sequence of the Plaza1 dataset. The RMSE is presented by the mean $\pm$ standard deviation across 50 runs with different random seeds which lead to different random initial values in the Gaussian solver. The case 'GAP. w/o reinit.' serves as an ablation study and is configured by disabling the non-Gaussian landmarks set and operations for re-initialization and sampling, which is essentially just GTSAM with the explicit regularization treatment (large-covariance priors). The RMSE in this case is computed on 29 out of 50 runs (i.e., 58% success rate) that were not ceased by the GTSAM error, indeterminant linear system. The GTSAM error is expected because, in this problem, bad initial values may incur near-singular linear systems.

Metric	Odom.	GAPSLAM	RBPF-SOG	GAP. w/o reinit.
RMSE (cm)	639.6	$\mathbf{34.4 \pm 0.0}$	56.0 ± 5.4	34.5 ± 0.0
Success rate (%)	-	**100**	**100**	58

evaluation metric than RMSE for *full posterior inference*, especially in highly non-Gaussian settings.

Results on point estimates

Table 4.1 presents the errors of point estimates for the entire Plaza1 sequence. RBPF-SOG incurs less accurate and consistent results due to the particle depletion issue. GAPSLAM without re-initialization is essentially the Gaussian solver with additional priors on landmarks. However, only 58% of runs without re-initialization return solutions despite having RMSE values comparable to those of GAPSLAM. The superior performance of GAPSLAM supports the effectiveness of the sample-based, uncertainty-aware re-initialization.

4.4.2 Object-based bearing-only SLAM

Datasets and real-world experiments

We estimate 6DOF camera poses and a map of object locations, using visual odometry and object detections from RGB videos. We aim to demonstrate the scalability of GAPSLAM to three-dimensional (3D) environments and its ability to fuse other types of measurements such as bearing-only. We test GAPSLAM using RGB data from

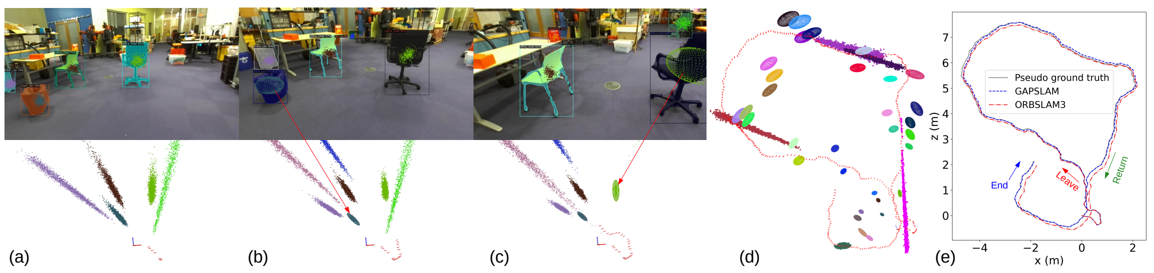


Figure 4-4: Estimates made by GAPSLAM for the realworld RGB sequence: (a,b,c) samples or confidence intervals (3σ) of object locations in 3D and their reprojection in images at key frames 10, 20, and 30, (d) final estimates of camera poses and samples and confidence intervals of object locations (we show samples of only 5 non-Gaussian landmarks to avoid clutter), and (e) a comparison of trajectory estimates with the pseudo ground truth. ORBSLAM3 did not close the big loop between 'leave' and 'return' due to the large change ($\sim$180 degrees) in viewing angles.

the RGB-D Scenes Dataset v2 [80] and a video collected by a monocular camera in our office area. For visual odometry, we use camera poses of key frames computed by ORB-SLAM3 [19], while object classes and masks in the key frames are detected using the Detic detector [142]. Camera poses are optimized using both visual odometry and measurements to objects.

Pre-processing object detections

We treat the center z_k of an object mask on an image as a projection factor that models the reprojection error $h(x_i, l_j; K) - z_i$, where $h(\cdot)$ denotes the reprojection of 3D object center l_j in the image with pose x_i and camera intrinsics K. This measurement z_k is either associated with an existing landmark or yields a new landmark, depending on the semantic and geometric information of the map. Thus wrong semantic labels may affect data association or spawn spurious landmarks. On the other hand, partial views of objects, which are often caused by occlusions or objects on image edges, lead to 2D mask centers that significantly deviate from the object center in 3D. To address these issues, we pre-process object detections as follows:

- Rejecting object detections that are potentially occluded by other detections on the image. Specifically, for any pair of detections (say i and j) whose bounding boxes intersect, one of the detections (say i) will be discarded if any of the following conditions is satisfied:

  - The object mask of detection i is smaller than a fraction (e.g., 10% in our experiments) of the mask of detection j. This condition is for rejecting very small masks.
  - The area of mask i within the bounding-box intersection is smaller than that of mask j and the area ratio between the intersection and the mask i is greater than a fraction (e.g., 10% in our experiments). This condition is for rejecting object detections that may be occluded by large bounding-box intersections.

- Increasing uncertainty in noise models of object mask centers that are close to image edges. Let us denote the relative position of the center of an object mask on an image as $(r_x, r_y) \in [-1, 1]^2$ so $r_x = 0$ and $r_y = 0$ indicate a mask center located at the image center and $|r_x|$ or $|r_y|$ that approaches 1 indicates a mask center that is close to image edges. Thus we define standard deviations in the noise model as

$$\sigma_x = \delta_x(1 - |r_x|) + \max(\delta_x, \delta_y)|r_x|, \tag{4.17}$$

$$\sigma_y = \delta_y(1 - |r_y|) + \max(\delta_x, \delta_y)|r_y| \tag{4.18}$$

  where δ_x and δ_y denote standard deviations of the mask in pixels.

- Rejecting any object detection which cannot be associated with landmarks in the same class via the maximum likelihood (ML) data association [70, 95] but passes the data association with landmarks that belong to different classes. This step intends discarding detections with wrong object classes. See the following section about the data association.

Data association and creation of new landmarks

We choose to implement the ML data association with some additional treatment for exploiting semantic information in object detections and posterior samples in

GAPSLAM. For a mask center z_k labeled by an object class and observed from pose x_i, we first compute its Mahalanobis distances to reprojections of any landmark l_j in the same class, as defined in

$$D_{kj}^2 = \|h(\hat{\mathbf{x}}) - z_k\|_C^2 \tag{4.19}$$

where $\mathbf{x} = (x_i, l_j)$ and

$$C = \left.\frac{\partial h}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}} \Sigma \left.\frac{\partial h}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}}^T + \Gamma. \tag{4.20}$$

Σ is the covariance between x_i and l_j in the Gaussian approximation of the posterior. Γ is the covariance of measurement noise model (e.g., (4.17) and (4.18)). The measurement is associated with l_j if D_{kj}^2 is the smallest among landmarks in the same class and passes the chi-square test, $D_{kj}^2 < \mathcal{X}_{d,\alpha}^2$, where d is the dimension of the measurement and α is the desired confidence level.

Note that the quality of the Gaussian approximation affects the ML data association so, if the ML data association does not accept any landmark, we then perform another round of data association using landmark samples. For any landmark l_j in the same class as detection z_k, we reproject samples of l_j onto the image and count the percent p_{kj} of samples that fall in the bounding box of the detection. The detection is associated with l_j if p_{kj} is the highest among landmarks in the same class and higher than a threshold (e.g., 10% in our experiments).

If the sample-based data association does not accept any landmark either, we create a new landmark in the map.

Results

Fig. 4-4 shows qualitative results on our realworld sequence. We set the Detic detector to recognize only a certain number of classes: cup, cereal box, trash can, skateboard, office chair, football, bottle, traffic cone, and toy car since they are quite common in our office and we found Detic worked with good recall and precision for these classes.

Figs. 4-4a, b, and c show how marginal posteriors of object locations evolve over key frames 10, 20, and 30, which occur before the 'leave' arrow on the path in Fig. 4-4e. The trash can in Fig. 4-4b and the chair in Fig. 4-4c have been identified as Gaussian landmarks by GAPSLAM so we draw confidence intervals (3σ) of their locations using the Gaussian approximation in lieu of samples.

There are 415 camera poses and 100 landmarks being created in the end. Fig. 4-4d visualizes all camera poses and landmarks marginals in 3D. Fig. 4-4e compares the pseudo ground truth and the estimated trajectories by GAPSLAM and ORBSLAM3. ORBSLAM3 fails to close the big loop between 'leave' and 'return' since the camera returns from an opposite viewing angle, and the bag-of-word approach in ORBSLAM3 does not recognize the return. The trajectory by GAPSLAM is aligned with the pseudo ground truth much better since object detections introduce many loop closures along the path. We found object detections by Detic performed consistently well under big changes of viewing angles. In GAPSLAM, the big loop between 'leave' and 'return' is successfully closed when the camera returns to and recognizes chairs and trash cans that appeared at early key frames (Figs. 4-4a, b, and c). The pseudo ground truth is the early portion of ORBSLAM3 results on a longer video, which circles our office three times along the same path, so the early portion has been corrected by loop closures introduced in the later portion.

Table 4.2 shows the RMSEs of estimated trajectories on the Scenes v2 sequences and our realworld sequence. GAPSLAM attains lowest errors in all the sequences. We attribute this accuracy to two reasons: 1) extra constraints introduced by object-based measurements, and 2) functions in the GAPSLAM algorithm. We also disable some functions in the object-based SLAM system to demonstrate their contributions to the estimation of the path. We find the pre-processing makes the biggest impact to the estimation while the re-initialization ranks the second and the sample-based data association ranks the third. Disabling all of them incurs significant estimation errors in some of the sequences (e.g., seqs. 9 and realwrold).

Fig. 4-5 shows the runtime of our system for the realworld sequence. The system implemented in Python supports an update frequency of 6 Hz for the slowest key

Table 4.2: Root mean square error (cm) of estimated paths under different settings. Each row below GAPSLAM serves as an ablation study that disables one (or three) of the functions in the object-based SLAM system.

Method	Sequences in RGB-D Scenes Dataset v2						Real world
	1	2	3	4	9	10	
ORBSLAM3	1.9	2.2	2.0	2.0	2.6	3.7	14.0
GAPSLAM	**0.9**	**1.0**	**0.7**	**1.2**	**0.7**	**2.6**	**5.2**
- no pre-processing	1.0	**1.0**	0.8	1.4	2.4	2.7	6.6
- no sample-based DA	**0.9**	**1.0**	**0.7**	**1.2**	0.8	**2.6**	5.8
- no reinitialization	**0.9**	**1.0**	0.8	1.3	**0.7**	**2.6**	6.6
- disable all above	1.0	2.2	0.9	1.3	4.2	2.7	9.5

Table 4.3: Runtime profiling of the object-based SLAM system and specifics for different datasets.

Items	Sequences in RGB-D Scenes Dataset v2						Real world
	1	2	3	4	9	10	
Time/frame (ms)	43.2	28.6	29.7	32.1	23.9	19.7	63.3
- pre-processing	4.2	2.7	2.2	3.0	1.5	1.1	9.0
- data association	6.2	4.6	7.6	7.2	7.0	4.8	9.8
- GAPSLAM	32.8	21.3	21.2	21.9	15.4	13.9	44.4
- reinit.	3.2	2.4	1.6	1.4	1.0	1.3	6.1
- GA	8.8	6.1	7.6	8.8	4.8	3.7	13.9
- samples	13.6	8.0	6.8	6.2	4.7	5.5	17.1
Obj. det./frame (-)	5.1	5.0	5.2	5.0	2.8	3.0	5.3
Key frames (-)	99	95	108	109	96	87	415
# of landmarks (-)	8	7	8	8	4	3	100

frames and, on average, affords an update frequency of 16 Hz. We are confident that the runtime can be greatly optimized by an improved implementation. Given that we only process key frames, GAPSLAM is able to support real-time operation. Table 4.3 shows the runtime profiling of our system and specifics of different datasets. The runtime on Scenes v2 sequences is expected to be faster since the sequences involve fewer poses and objects. The crux of averaged runtime is on steps for updating samples and the Gaussian approximation. However, Fig. 4-5 shows that updating samples only dominates the runtime in early key frames, when many non-Gaussian

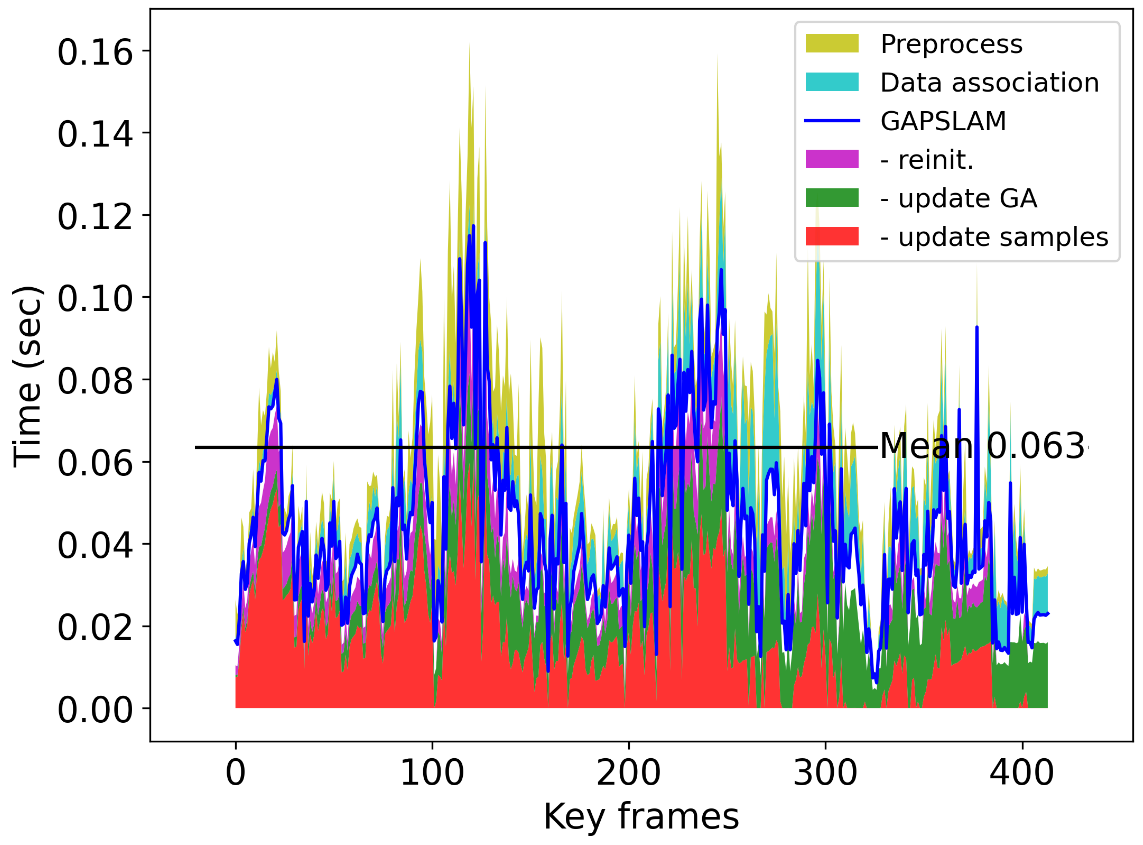


Figure 4-5: Runtime profiling of the object-based SLAM system (in Python).

landmarks had just been created. Revisiting objects at later key frames reduces the uncertainty in the posteriors of object locations, so more and more objects are moved to the set of Gaussian landmarks. Thus the computation burden gradually transfers to updating the Gaussian approximation. The scalability of updating the Gaussian approximation can be improved via many strategies such as fix-lag smoothing [31].

4.5 Live demo: a streaming platform for object-based SLAM

We developed a streaming platform for executing real-time object-based SLAM (see Fig. 4-6). The platform consists of a web-based client application on a mobile device and a centralized SLAM application server that can handle multiple connected clients simultaneously. The mobile client offers instant feedback to the user while uploading a camera feed to the SLAM server in real-time. The SLAM server is an implementation of the GAPSLAM algorithm [62] that offers undelayed initialization of objects via a hybrid representation of particles and Gaussian models for describing probable object locations (see Fig. 4-7). We highlight two key features of our technical demonstration: First, our ability to deploy over-the-air through the browser on multiple mobile and robot hardware platforms, enabling the sharing of mapping results among multiple online peers. Second, our system renders critical feedback to the user by visualizing the uncertainty of the camera pose and object location estimates and their evolution as object detection accrues. A sample video about the demo can be found at `https://youtu.be/4XnWBBHBjc8`.

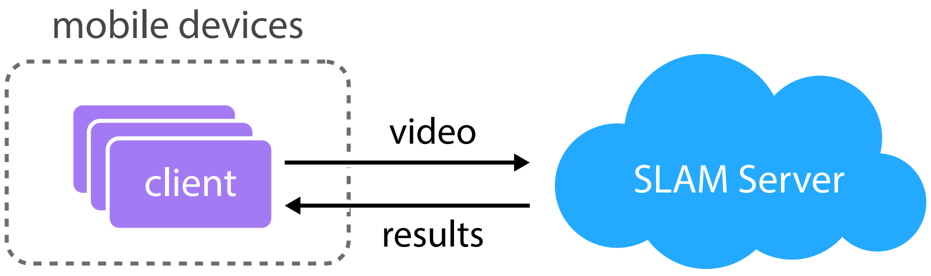


Figure 4-6: The streaming platform for real-time object-based SLAM.

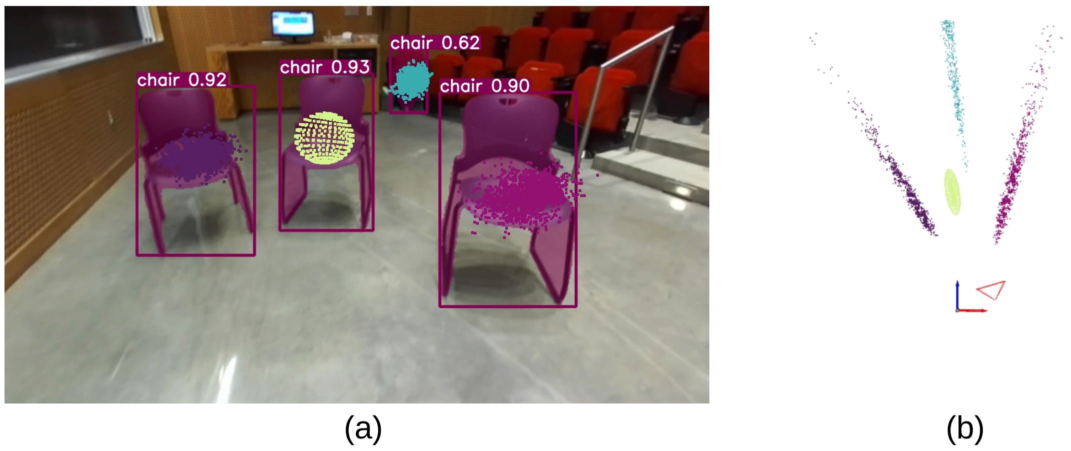


Figure 4-7: Object-based SLAM results: a) projection of object representations and b) bird view of the camera and object representations.

4.5.1 Detailed implementation

The mobile devices simply stream videos to the server so the implementation was mainly made on the SLAM server. The server is a workstation with an AMD Ryzen ThreadRipper 3970X CPU with 32 cores and 64 threads and NVIDIA RTX 3090 GPUs. On the workstation, we detect objects in videos using Yolo-v8 [68], compute visual odometry using ORB-SLAM3 [19], infer camera poses and object locations using the GAPSLAM algorithm, and stream visualization to a web application which can be viewed from any browser using the IP of the server. The web application was developed using the web framework Flask. We used Redis, an in-memory key–value database as the middleware to exchange data across processes.

4.6 Extension study of the GAPSLAM algorithm

4.6.1 Criteria for determining Gaussian landmarks

We used the largest eigenvalue of the empirical covariance of samples to determine the transition to the Gaussian approximation, as mentioned in Sec 4.3.3. Fig. 4-8a compares the largest eigenvalues derived from samples and Gaussians in the range-only SLAM experiment in Sec. 4.4.1. Fig. 4-9 visualizes the samples and Gaussians to explain the evolution of the largest eigenvalues. At time step 0, both sample-based belief and Gaussians derive large eigenvalues for all landmarks. Note that we added large-covariance prior factors on each landmark to enable the computation of the Gaussians at this time step. Later at time step 5, the Gaussians for $L1$ and $L2$ converge to local modes, ignoring other modes captured by samples in Fig. 4-9, so the largest eigenvalues in these Gaussians fall below eigenvalues derived from samples. At time step 10, the sample-based belief of $L1$ and $L2$ then converges to uni-modal distributions, leading to similar eigenvalues as the Gaussians for these two landmarks. One can find in Fig. 4-8a that a small value such as 3.0 for this problem should serve sufficiently well as the threshold of the eigenvalue derived from samples and determine the transition to solely using the Gaussian approximation. However, the drawback of using the eigenvalue is that the threshold must change with the scale of problems or landmarks.

We explore three other metrics whose values range from 0 to 1. The key idea behind these metrics is to check the similarity between the covariances derived from samples and Gaussians. We review a geometric and probabilistic interpretation of the covariance to understand better the motivation for using these metrics. Fig. 4-10 shows an ellipsoid derived from a covariance. The length and directions of the semi-axes of the ellipsoid are determined by the square roots of eigenvalues and eigenvectors of the covariance. The ellipsoid can also be interpreted as the 1-σ confidence interval of the zero-mean Gaussian distribution with the covariance. Therefore, the similarity between the covariances can be examined by comparing the ellipsoids constructed from eigenvalues and eigenvectors.

The first metric is the correlation matrix distance (CMD) [58]

$$\mathrm{CMD}(\Sigma_1, \Sigma_2) = 1 - \frac{\mathrm{tr}\,(\Sigma_1 \Sigma_2)}{\|\Sigma_1\|_f \|\Sigma_2\|_f}, \tag{4.21}$$

where $\|\cdot\|_f$ denotes the Frobenius norm. Note that, while CMD is coined for processing correlation matrices. Equation (4.21) also applies to covariances. The CMD becomes zero if the covariances are equal up to a scaling factor. Fig. 4-8b shows the CMD between covariances derived from samples and Gaussians of landmarks in the range-only SLAM experiment. The shaded areas in the figure show that the CMD drops below 0.1 when the sample-based belief converges to a uni-modal, certain situation, indicating that the CMD is also an effective metric for determining the transition.

Figs. 4-8c and d show the remaining two metrics we explored, although they are not effective for detecting the transition. Fig. 4-8c visualizes the evolution of the cosine similarity between vectors of eigenvalues. The cosine similarity (CS) was defined by

$$\mathrm{CS}(\Sigma_1, \Sigma_2) = \frac{\mathbf{v}_1^T \mathbf{v}_2}{\|\mathbf{v}_1\|_2 \|\mathbf{v}_2\|_2}, \tag{4.22}$$

where $\mathbf{v}_i$ is the vector of sorted eigenvalues of covariance Σ_i and Σ_i is a covariance

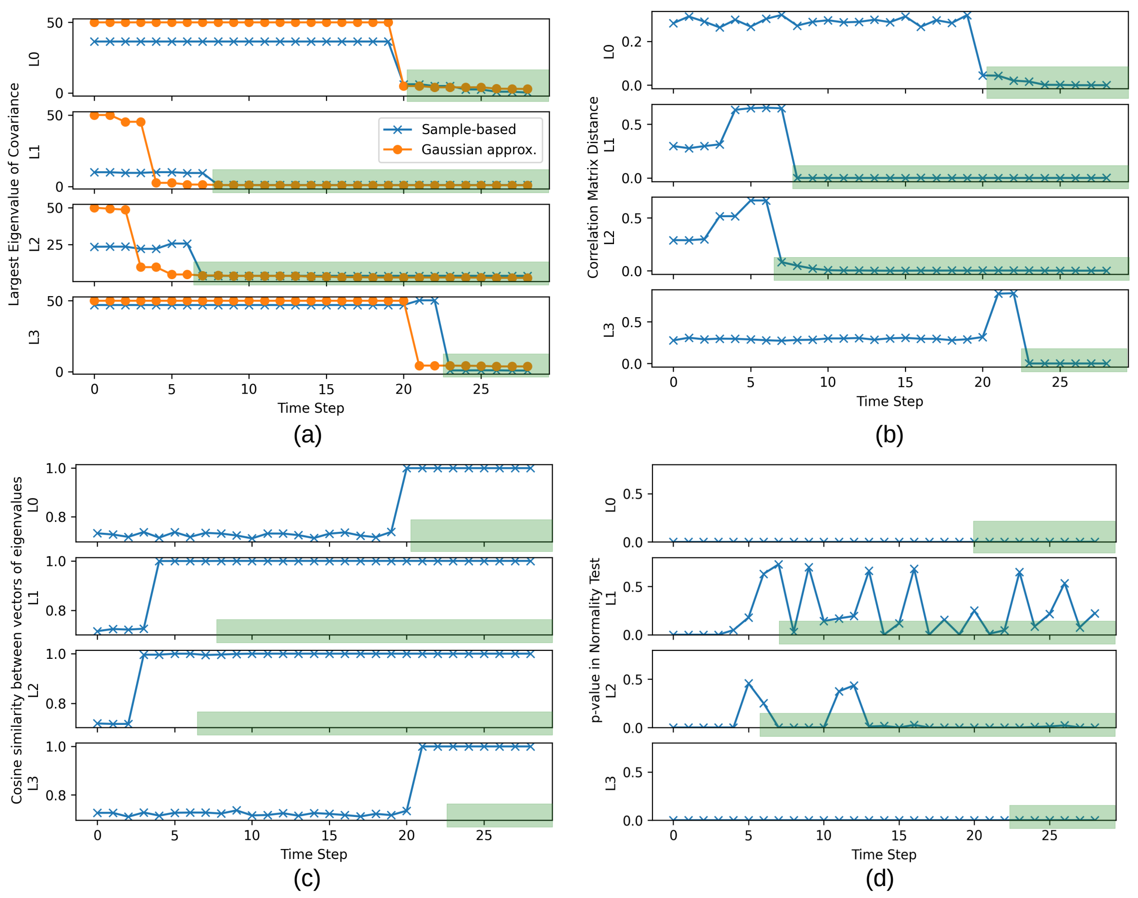


Figure 4-8: Criteria for the transition from samples to Gaussians: a) largest eigenvalues of covariances, b) correlation matrix distance (CMD) [58], c) cosine similarity between vectors of eigenvalues, and d) D'Agostino and Pearson's normality test [24]. The shaded areas indicate time steps where landmark samples look similar to Gaussians. The figure shows that the CMD, which ranges from 0 to 1, can serve as a good criterion using a problem- or scale-independent threshold (e.g., 0.1).

derived from samples or Gaussians. Fig. 4-8c shows that the sharp rise of cosine similarity for landmarks $L1$ or $L2$ does not occur at the time step when the distribution of samples is uni-modal and highly concentrated (i.e., shaded areas in Fig. 4-8). The inefficacy of the cosine similarity is in part due to ignoring eigenvectors of the covariance (e.g., directions of semi-axes of ellipsoid in Fig. 4-10). Last, we performed D'Agostino and Pearson's normality test on samples and show the resulting p-value in Fig. 4-8d. Higher p-value indicates a higher likelihood that the samples were drawn from a Gaussian distribution. Unfortunately, the p-value does not exhibit any useful pattern. The inefficacy of the normality test is expected because the samples are still

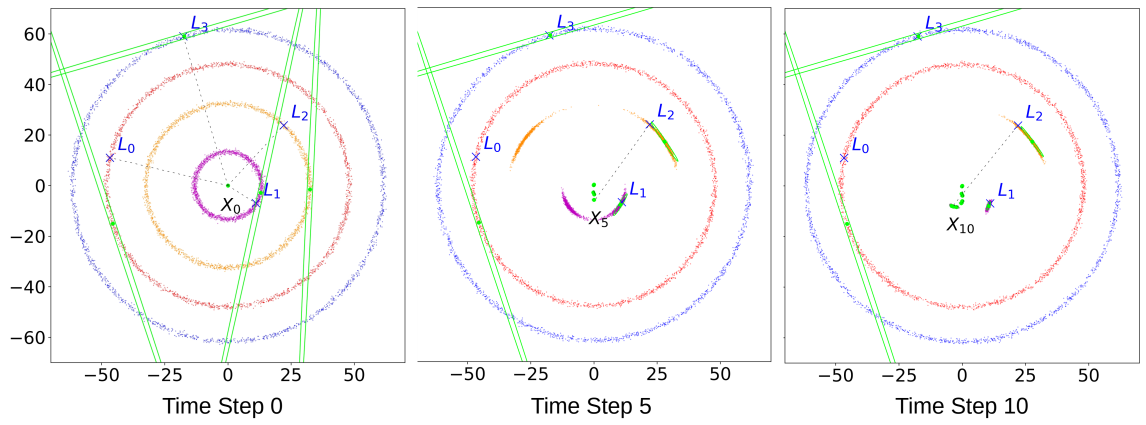


Figure 4-9: GAPSLAM results on the Plaza1 dataset for early time steps. Color dots are posterior samples and ellipses in green are 2-σ confidence intervals in Gaussian approximations.

Algorithm 6: GAPSLAM with the new criterion of Gaussian landmarks

```
Input: New factor $f$ of landmark $l$, Gaussian solver $g$, dictionary $\mathcal{D}$ of
       non-Gaussian landmark samples
1  if l not in the Gaussian solver then                                   // new landmark
2      Add $l$ to non-Gaussian landmarks $\mathcal{D}$ as a new key
3  Add $f$ to the Gaussian solver
4  if $l$ in non-Gaussian landmarks $\mathcal{D}$ then
5      LandmarkReinitializer($l$, $g$, $\mathcal{D}[l]$)                  // re-init. in the solver
6  Update the MAP estimate in the Gaussian solver
7  if $l$ in non-Gaussian landmarks $\mathcal{D}$ then
8      $\mathcal{D}[l]$ = LandmarkSampler($l$, $g$)                       // update samples
9      Compute empirical covariance matrix $C$ of $\mathcal{D}[l]$
10     Get covariance $\Sigma$ of $l$ in the Gaussian solver              // new criterion
11     if Correlation matrix distance between $C$ and $\Sigma$ is small
12     and largest eigenvalues of $C$ and $\Sigma$ are similar then       // new criterion
13         Delete $l$ from $\mathcal{D}$                   // no longer non-Gaussian landmarks
14 return $g$ and $\mathcal{D}$                             // updated solver and dictionary
```

subject to the non-Gaussian marginal of landmarks and, statistically, do not resemble a Gaussian.

Based on the investigation above, we update our criterion for determining the transition of Gaussian approximation to satisfy the following two conditions:

- the CMD between covariances derived from samples and Gaussians is below threshold $\alpha \in [0, 1]$;

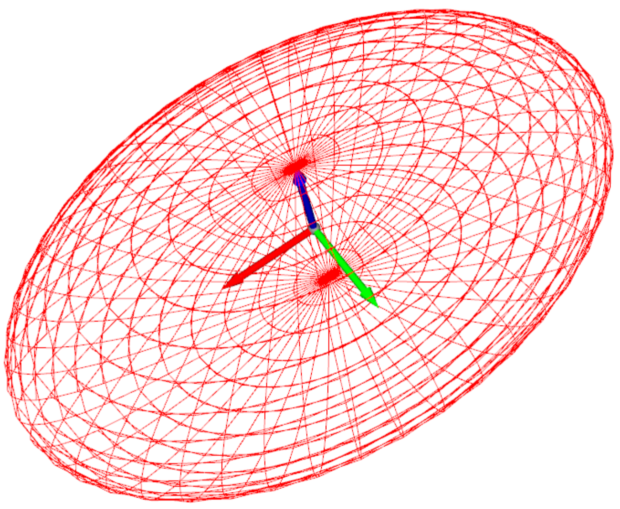


Figure 4-10: Ellipsoid derived from the covariance $diag(9, 4, 1)$. The lengths of the semi-axes of the ellipsoid are 3, 2, and 1, which are square roots of eigenvalues of the covariance. The unit vectors indicate the directions of the semi-axes, which are aligned with the eigenvectors of the covariance.

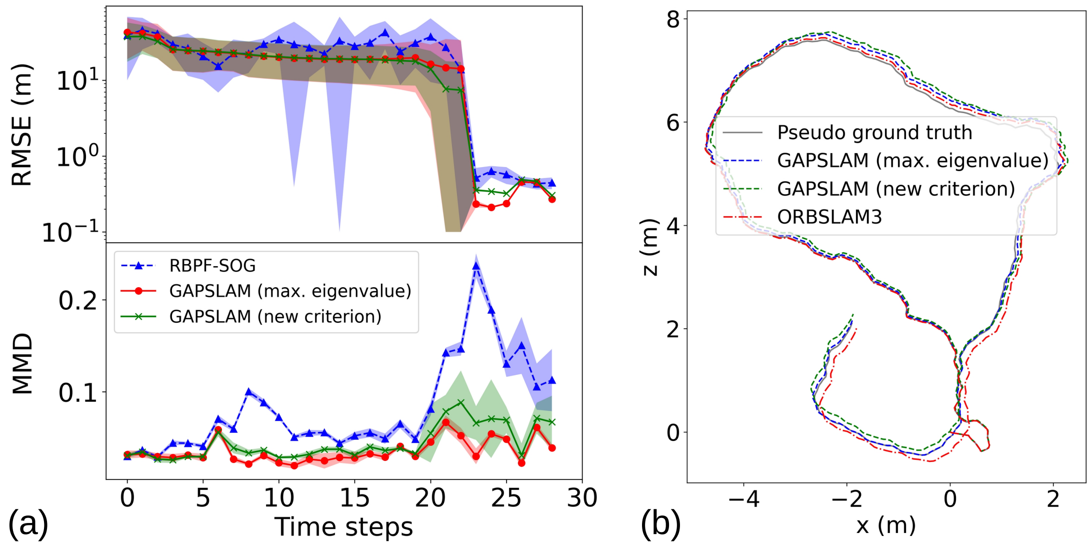


Figure 4-11: Results of the new criterion for determining the transition to Gaussians: (a) RMSE and MMD for the range-only SLAM experiment shown in Fig. 4-3c and (b) a comparison of trajectory estimates for the real-world object SLAM experiment shown in Fig. 4-4e.

- the ratio between largest eigenvalues of the covariances is in the range $[\frac{1}{\beta}, \beta]$ where $\beta > 1$.

Because the CMD only identifies similar covariances up to a scaling factor, we design the second condition to match scales of covariances derived from samples and Gaussians. We set the threshold α of CMD to 0.1 and the threshold β to 1.2 as the new criterion and perform experiments about range-only SLAM and object-based bearing-only SLAM again. Fig. 4-11 shows that, while the new criterion consistently works for both range-only SLAM and object-based bearing-only SLAM, the resulting performance such as MMD or RMSE is slightly worse than that using the old criterion, which sets thresholds for largest eigenvalues. This result indicates that if we know the environment *a priori*, we had better set predetermined thresholds of the largest eigenvalues to determine the transition from samples to Gaussians. However, the new criterion enjoys improved generalizability for various environments. Algorithm 6 updates the GAPSLAM algorithm (i.e., Alg. 4) with the new criterion of Gaussian landmarks.

4.6.2 Computational complexity

Time complexity of the GAPSLAM algorithm is dominated by solving the Gaussian approximation and drawing samples of non-Gaussian landmarks, as seen in the runtime profiling in Figs. 4-3 and 4-5. One can refer to [31, Ch. 4] for the complexity of MAP inference in SLAM via solving least-squares, which is crucial in the computation of the Gaussian approximation. Here, we focus on discussing the other computational crux in the GAPSLAM algorithm, i.e., sampling the marginal of a non-Gaussian landmark. The detailed version of the complexity in the sampling (i.e., Alg. 3) is $O(N^3 + KN^2 + MNK)$ where $N = \max_i |X_{\mathcal{V}_i}|$ denotes the maximal number of robot poses that observe the same non-Gaussian landmark, K is the number of robot path samples, and M is the number of landmark samples per path. The first two terms $N^3 + KN^2$ are the complexity of drawing K robot path samples from the Gaussian distribution via Cholesky decomposition of the covariance and matrix-vector production. The last term MNK is the complexity of using the importance sampling in Sec. 4.3.2 to generate KM landmark samples from a density which is the product of N factors. Note that, if a landmark is observed by many robot poses, the posterior belief of that landmark is generally very certain thus is highly likely being

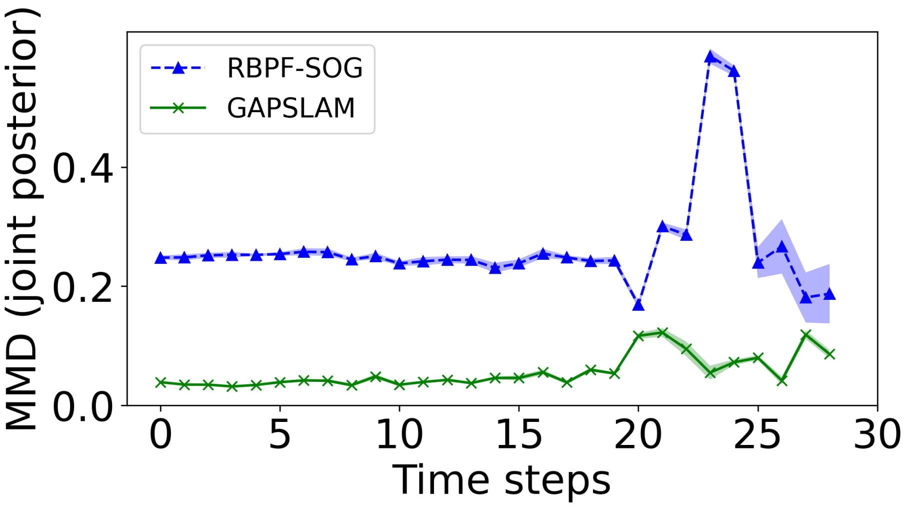


Figure 4-12: MMD of samples drawn from the joint posterior.

identified as a Gaussian landmark. Thus $N = \max_i |X_{\mathcal{V}_i}|$ can be sufficiently bounded by a constant which depends on the specific type of measurements. Therefore, the complexity of drawing KM samples of a non-Gaussian landmark can be reduced to $O(KM)$.

4.6.3 Samples of the joint posterior

The GAPSLAM algorithm can be extended to draw samples of the joint posterior at the expense of computation resources. As shown in (4.11) and (4.13) of Sec. 4.3.2, we are able draw M landmark samples $\{l_i^{(m,k)}\}_{m=1}^{M}$ from the conditional $p(L_i|X_{\mathcal{V}_i} = x_{\mathcal{V}_i}^{(k)}, z_{\mathcal{V}_i}, d_{\mathcal{V}_i})$, given a robot path sample $x_{\mathcal{V}_i}^{(k)}$. We can randomly select one of the landmark samples and joint it with the robot path sample to form a sample $(x_{\mathcal{V}_i}, l_i)^{(k)}$ of the marginal of variables $(X_{\mathcal{V}_i}, L_i)$. The time complexity of drawing K samples $\{(x_{\mathcal{V}_i}, l_i)^{(k)}\}_{k=1}^{K}$ is still $O(KM)$. Note that $x_{\mathcal{V}_i}$ is part (i.e., some components) of a sample from the high-dimensional Gaussian approximation $g(\mathbf{X}_t, L_{\mathcal{G}_t}|\mathbf{z}_t, \mathbf{o}_t, \mathbf{d}_t)$. If we want to sample the joint posterior of all variables, we can draw K samples from the Gaussian first and then sample conditionals of non-Gaussian landmarks separately in parallel. The complexity of drawing K joint posterior samples is $O(G^3 + KG^2 + KMNQ)$ where $G = |\mathbf{X}_t| + |L_{\mathcal{G}_t}|$ is the number of robot poses and Gaussian landmarks and $Q = |L_{\mathcal{N}_t}|$ is the number of non-Gaussian landmarks. Note that this time complexity covers the worst cases in computation. In practice, the runtime performance can be greatly optimized via efficient implementation strategies, including 1) utilizing matrix sparsity to alleviate the complexity $G^3 + KG^2$ in sampling the Gaussian and 2) parallelizing the sampling of non-Gaussian landmarks.

We draw samples from the joint posterior for the range-only SLAM dataset. These samples are compared with samples from RBPF-SOG in terms of MMD. Note that this MMD is computed using joint posterior samples provided by NSFG, so it is different from Fig. 4-3, which shows the MMD of samples drawn from marginals. Fig. 4-12 shows the MMD of samples drawn from the joint posterior. The lower MMD by GAPSLAM indicates that the samples by GAPSLAM resemble the joint posterior much better than those by RBPF-SOG.

4.7 Summary

We presented a real-time algorithm, GAPSLAM, that precisely infers non-Gaussian/multi-modal marginal posteriors encountered in SLAM. Our experiments justified the efficacy of the adaptive modeling strategy for efficient full posterior inference, and the uncertainty-aware re-initialization technique for explicitly correcting linearization points in nonlinear optimization solvers. Future work includes inferring the joint posterior of multiple variables, incorporating more complex likelihood of measurements (e.g., learned orientation distributions [94], multi-modal data association [34]), and applying GAPSLAM to realworld planning tasks based on the non-Gaussian belief of obstacles [54].

Chapter 5

Incremental inference of the joint posterior in SLAM via learning normalizing flows on the Bayes tree

The content in this chapter is mainly based on the following paper:

- Qiangqiang Huang, Can Pu, Kasra Khosoussi, David M. Rosen, Dehann Fourie, Jonathan P. How, John J. Leonard. Incremental Non-Gaussian Inference for SLAM Using Normalizing Flows. *IEEE Transactions on Robotics (T-RO)*, 2023.

5.1 Introduction

Chapter 4 develops a scalable, real-time algorithm for full posterior inference of marginals. Inferring the joint posterior is a much more challenging task due to the increasing dimensionality of the joint posterior encountered in SLAM. This chapter develops a scalable algorithm that incrementally updates a non-Gaussian, parametric approximation of the joint posterior, beyond the Gaussian approximation. The technical goal in this chapter is to find a computationally tractable density representation that has the necessary flexibility to approximate the joint posterior. Specifically, we aim to develop an algorithm that is able to perform the following tasks in non-

Gaussian settings:

Task 1. Solve for a distribution that effectively approximates the full posterior.

Task 2. Draw samples from the distribution to infer quantities of interest using Monte Carlo integration.

Task 3. Allow incremental updates of the distribution.

A key step for scalability is to extend the Bayes tree, which was proposed in iSAM2 [71] for analyzing the Gaussian approximation, to non-Gaussian settings. The Bayes tree algorithm converts a cyclic factor graph into an acyclic directed graph using a variable elimination game [57] and max-cardinality search [129]. From an information-theoretic standpoint, the Bayes tree shows how to factorize the original *high-dimensional* posterior into a *sequence* of *low-dimensional* conditionals that encode a tree-like graphical model. We propose to learn *non-Gaussian* models of conditionals that factorize the posterior using the Bayes tree. The learned non-Gaussian models in turn reconstruct the posterior. To perform inference, we draw samples sequentially from these models following the order governed by the Bayes tree.

We exploit normalizing flows to represent the non-Gaussian conditionals. Normalizing flows, as emerging tools for density modeling [108, 37, 66, 109, 76, 101], have shown strong expressive power for representing complex densities and support fast sampling. An important property is that conditionals of the modeled density can be extracted easily from a normalizing flow model, which perfectly matches our need for modeling conditionals. Therefore, the problem statement in this chapter is to find normalizing flows that can represent the joint posterior 2.1.

Contributions

We present a novel general solution, called normalizing flows for incremental smoothing and mapping (NF-iSAM), to model and sample the *joint posterior distribution* of general SLAM problems using the Bayes tree and the normalizing flow model. Key contributions of this work include:

1. NF-iSAM introduces normalizing flows to factor graph inference for robot perception.

2. NF-iSAM generalizes the Bayes tree to perform full (non-Gaussian) posterior estimation for the joint posterior distribution.

3. NF-iSAM augments normalizing flows from low-dimensional inference to high-dimensional cyclic factor graphs.

4. NF-iSAM achieves superior accuracy in comparison to state-of-the-art SLAM algorithms in describing the full posteriors encountered in highly non-Gaussian SLAM settings.

Outline

The rest of this chapter is organized as follows. Sec. 5.2 presents the problem statement and the high-level idea of the inference framework without digging into density modeling techniques. Sec. 5.3 delineates the formulation for modeling densities and the detailed algorithms. Sec. 5.4 summarizes our implementation and experimental setup. Sec. 5.5 provides experimental results and demonstrates the advantages of our algorithm in comparison with state-of-the-art algorithms. Finally, Sec. 5.6 concludes with a summary of the contributions of this paper and a discussion of future research directions.

5.2 Inference on the Bayes tree

5.2.1 Factor graphs and the Bayes tree

We provide a brief review of factor graphs and the Bayes tree which are the foundation of our inference method. Posterior distributions in SLAM problems are usually represented by factor graphs [31], which have been introduced in Chapter 2.1.2. Fig. 5-1a shows an example of factor graph.

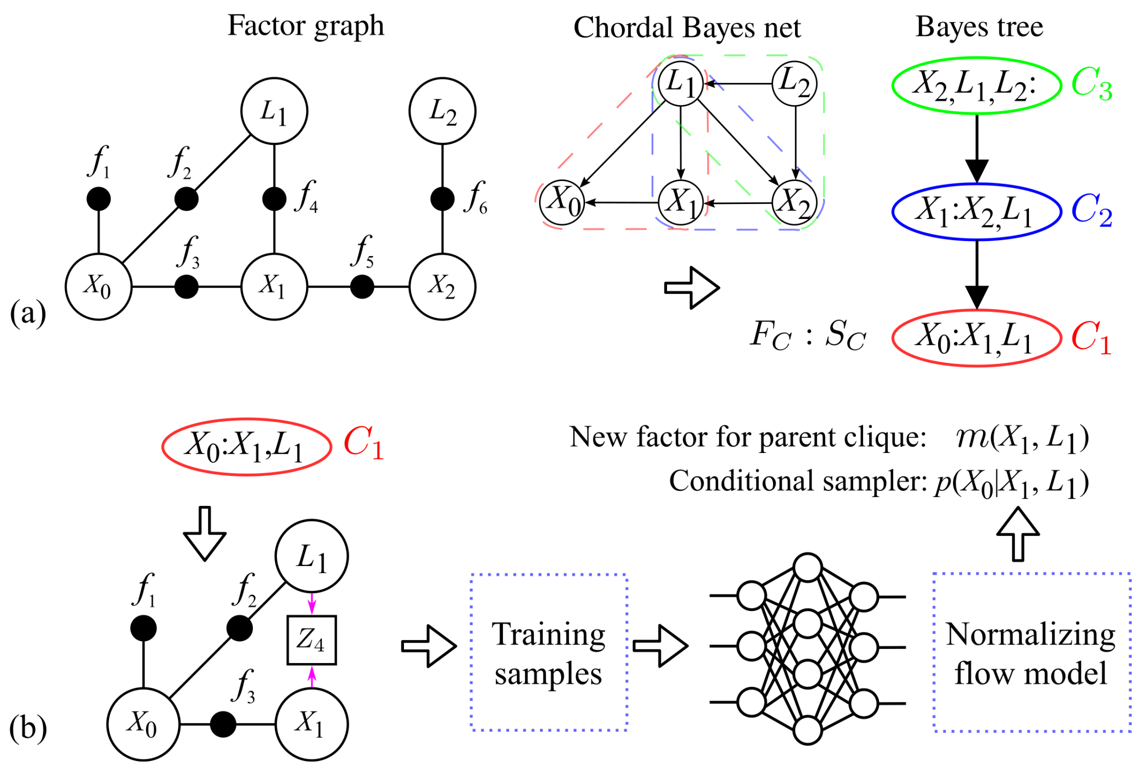


Figure 5-1: Illustration of core steps in NF-iSAM: (a) conversion from a factor graph to the Bayes tree with elimination ordering $(X_0, X_1, X_2, L_1, L_2)$ and (b) construction of clique conditional sampler via normalizing flows. The colon in a Bayes tree node splits frontal variables F_C and separator S_C. The normalizing flow model is learnt from training samples by neural networks. In factors of Bayes tree node C_1, the factor f_4 is reverted to a Bayes net where the measurement variable Z_4 is treated unobserved to enable ancestral sampling for rapidly simulating training samples (See Fig. 5-4 for more details).

The Bayes tree is a directed variant of the junction tree [31]. Given a variable elimination ordering, a factor graph can be converted to a Bayes tree by the variable elimination algorithm [71, Alg. 2] and the Bayes tree algorithm [71, Alg. 3]. Nodes on the Bayes tree represent cliques of variables as shown in Fig. 5-1(a). Variables in a clique shared with its parent clique are called the separator while the remainder are frontal variables. The Bayes tree factorizes the posterior by a sequence of conditionals [31, 44], as seen in

$$p(\mathbf{\Theta}|z) = \prod_{C\in\mathbf{C}} p(F_C|S_C, \mathbf{z}) = \prod_{C\in\mathbf{C}} p(F_C|S_C, z_C), \tag{5.1}$$

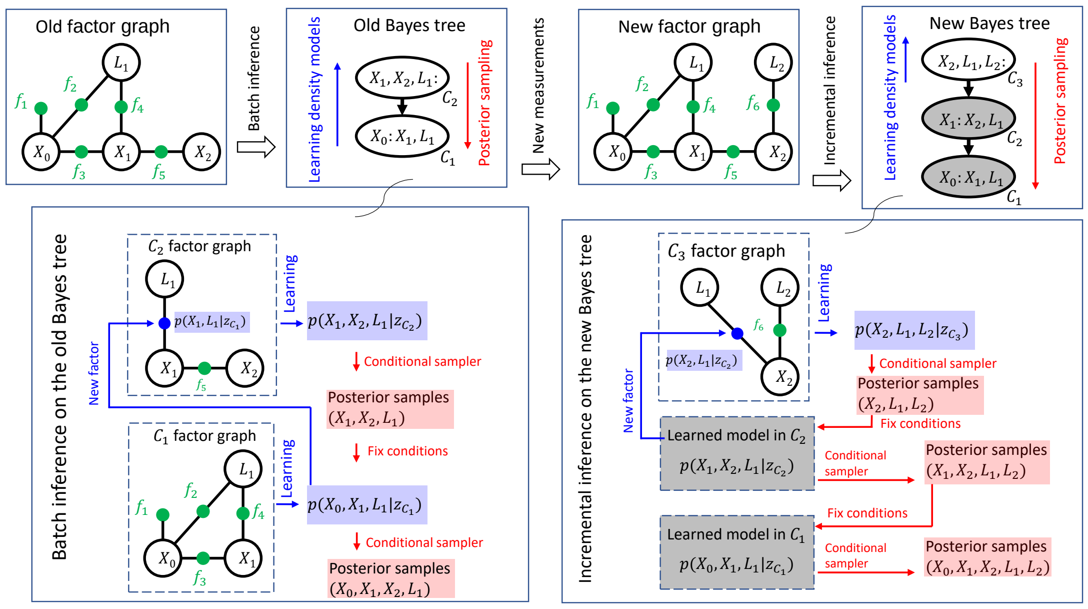

Old factor graph
Batch inference
Old Bayes tree
Learning density models
Posterior sampling
New measurements
New factor graph
Incremental inference
New Bayes tree
Batch inference on the old Bayes tree
C_2 factor graph
C_1 factor graph
New factor
Learning
$p(X_1, X_2, L_1|z_{C_2})$
Conditional sampler
Posterior samples (X_1, X_2, L_1)
Fix conditions
$p(X_0, X_1, L_1|z_{C_1})$
Posterior samples (X_0, X_1, X_2, L_1)
Incremental inference on the new Bayes tree
C_3 factor graph
$p(X_2, L_1, L_2|z_{C_3})$
Posterior samples (X_2, L_1, L_2)
Learned model in C_2 $p(X_1, X_2, L_1|z_{C_2})$
Learned model in C_1 $p(X_0, X_1, L_1|z_{C_1})$
Posterior samples (X_1, X_2, L_1, L_2)
Posterior samples $(X_0, X_1, X_2, L_1, L_2)$

Figure 5-2: Illustration of learning and inference procedures on the Bayes tree. The variable elimination ordering $(X_0, X_1, X_2, L_1, L_2)$ for constructing Bayes trees is consistent with that in Fig. 5-1. A Bayes tree node is denoted by C. The colon in a Bayes tree node splits frontal and separator variables. The white node on the new Bayes tree is the changes caused by new measurements. Updates of density models only occur to the changed part on the Bayes tree. Green factors in a clique factor graph are from user-defined factor graphs (e.g., old and new factor graphs here) while blue factors are separator densities from child cliques and are passed to parent cliques as new factors resulted from variable elimination.

where $\mathbf{C}$ is the collection of cliques, F_C denotes the set of frontal variables in clique C, S_C denotes the separator, and z_C denotes the set of observations in and below clique C on the Bayes tree (we designate the root clique as the top of the tree). The last equality in (5.1) is a result of applying the conditional independence relation $F_C \perp\!\!\!\perp (\mathbf{z} \setminus z_C)|S_C$. Note that S_C is the junction between clique C and its parent clique. Once S_C is fixed with a realization, the measurements above C, i.e., $\mathbf{z} \setminus z_C$, will not affect F_C. Thus, $\mathbf{z} \setminus z_C$ can be excluded from the condition in (5.1). The factorization (5.1) reflects the information-theoretical view of the Bayes tree we mentioned in Section 5.1: a decomposition of the original *high-dimensional* posterior into *a sequence* of *low-dimensional* clique conditionals $p(F_C|S_C, z_C)$. We will solve a sequence of *low-dimensional* density modeling problems to learn the clique conditionals. The learned conditionals in turn reconstruct the posterior, which enjoys better scalability than directly learning the *high-dimensional* posterior.

5.2.2 Inference using the clique conditionals

We introduce the main idea of our inference method for modeling clique conditionals and drawing samples from the full posterior distribution. The inference method consists of two steps: (i) performing the bottom-up belief propagation on the Bayes tree for learning the clique conditionals $p(F_C|S_C, z_C)$ and (ii) applying ancestral sampling [9, Ch. 8.1.2] to the learned conditionals for drawing posterior samples in a top-down traversal of the Bayes tree (see the batch inference in Fig. 5-2 for an example of the two steps).

The primary challenge is modeling the clique density $p(F_C, S_C|z_C)$ and extracting the clique conditional $p(F_C|S_C, z_C)$ in the belief propagation. The bottom-up belief propagation is the unidirectional sum-product message passing from the leaf nodes to the root node in a tree [9, Fig. 8.52] (also known as the Shafer-Shenoy algorithm [117] for junction trees). The clique density for any clique C, as a result of the product operation in the sum-product, is formulated by

$$p(F_C, S_C|z_C) \propto \prod_{q\in\mathcal{C}_C} p(S_q|z_q) \prod_{\substack{\Theta_{f_i}\subseteq C \\ \Theta_{f_i}\not\subseteq S_q}} f_i(\Theta_{f_i}), \tag{5.2}$$

where q denotes any child clique of clique C and $\mathcal{C}_C$ indicates the set of the child cliques. Relation (5.2) shows that the clique density contains some user-defined factors $f(\cdot)$ and separator densities $p(S_q|z_q)$ which are resulted from variable elimination (i.e. the sum operation) in the child cliques. We will introduce normalizing flows in the following section to model the clique density as well as extract the clique conditional and the separator density. The separator density $p(S_C|z_C)$ will be passed upwards as a new factor for joining in the sum-product in the parent clique of C.

In addition, we will also use normalizing flows to construct conditional samplers for each clique conditional $p(F_C|S_C = s_C, z_C)$ such that independent samples of F_C

can be drawn once the separator is fixed by a realization s_C. These conditional samplers will be created during the belief propagation and cached for performing the ancestral sampling in the top-down traversal.

5.3 SLAM via normalizing flows

We first briefly review normalizing flows in Section 5.3.1. We then present our novel technique for modeling and learning clique conditional $p(F_C|S_C, z_C)$ via normalizing flows in Section 5.3.2. Finally, in Section 5.3.3 we describe our *incremental* inference approach that generates joint posterior samples.

5.3.1 Normalizing flows

Normalizing flows have shown strong expressive power for modeling complex distributions. An extensive review can be found in [76, 101]. A normalizing flow is a *transformation* T that maps a D-dimensional target random variable $\mathbf{X} := (X_1, \ldots, X_D)$ onto another D-dimensional random variable $\mathbf{Y} := (Y_1, \ldots, Y_D)$ that follows a reference distribution $q(\mathbf{y})$. We choose the standard multivariate normal distribution $\mathcal{N}(0, I_D)$ as the reference distribution which is also a common choice in related literature for its advantages in computation [37, 66, 109, 76, 101]. We will explain those advantages in the later discussion of this subsection and, in particular, stress their connections with some properties of the chosen reference distribution including sampling efficiency, separability (i.e., independent components), and log-concavity. *Our objective* in this subsection is to use the reference distribution, $q(\mathbf{y})$, and the transformation, T, to model the target distribution, $p(\mathbf{x})$; see Fig. 5-3 for an example.

We take the transformation to be a lower-triangular map:

$$T(\mathbf{x}) = \begin{bmatrix} T_1(x_1) \\ T_2(x_1, x_2) \\ \vdots \\ T_D(x_1, x_2, \ldots, x_D) \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_D \end{bmatrix} = \mathbf{y}, \tag{5.3}$$

where each function T_d is differentiable, bijective, and increasing with respect to x_d [108, 37, 66, 101, 109]. In general, a transformation between random variables of two distributions without additional constraints is not unique. It has been proven, however, that triangular maps to a standard Gaussian exist and are unique for any non-vanishing densities [20, 14, 133]. Theoretically, the Knothe–Rosenblatt rearrangement provides a scheme to construct the triangular map by defining T_1 to T_D sequentially [133, Ch. 1]. However, it is computationally impractical to construct the exact Knothe–Rosenblatt rearrangement for modeling a general multivariate density. Thus, in practice, many works opt to estimate such a map by seeking the optimal one among a parameterized family of triangular maps [38, 102, 66]. We will follow the same practice and show how to solve for the optimal T. Before the optimization problem for T, we review *three* useful properties of triangular maps that have been widely exploited for constructing the map, drawing samples, and extracting marginals and conditionals.

Property 1: Since T_d is differentiable, lower-triangular, and increasing with respect to x_d, its Jacobian matrix is triangular with positive diagonals. The absolute value of Jacobian determinant is thus given by,

$$|T'(\mathbf{x})| = \prod_{d=1}^{D} \frac{\partial T_d}{\partial x_d}. \tag{5.4}$$

For any such T, by change of variables, we have $p(\mathbf{x}; T) = q(T(\mathbf{x}))\ |T'(\mathbf{x})|$, where $p(\mathbf{x}; T)$ denotes a density defined by $q(\mathbf{y})$ and T for modeling $p(\mathbf{x})$. Thus, with the triangular structure, $p(\mathbf{x}; T)$ can be expressed by

$$p(\mathbf{x}; T) = q(T(\mathbf{x})) \prod_{d=1}^{D} \frac{\partial T_d}{\partial x_d}. \tag{5.5}$$

The density model (5.5) can be evaluated once we can evaluate T and its Jacobian. Our goal is to find T that makes $p(\mathbf{x}; T)$ well approximate the target density $p(\mathbf{x})$.

Property 2: T_d essentially models the conditional probability $p(x_d|x_1, \ldots, x_{d-1})$ [66]. This idea is also referred to as autoregressive flows in literature [101, 76]. For

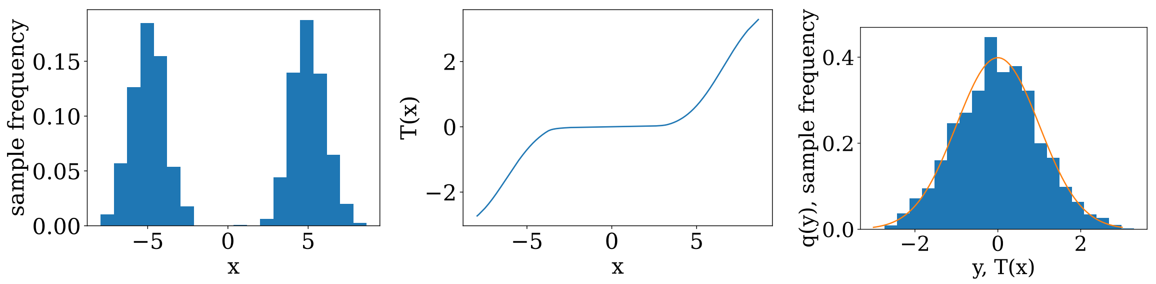


Figure 5-3: A one-dimensional example of transformation function: histogram of sample x (left), transformation function $T(x)$ (middle), and histogram of transformed samples and reference variable $y \sim N(0,1)$ (right).

any d where $2 \leq d \leq D$,

$$p(x_1, \ldots, x_{d-1}; T) = q(y_1, \ldots, y_{d-1}) \prod_{i=1}^{d-1} \frac{\partial T_i}{\partial x_i} \tag{5.6}$$

and

$$p(x_1, \ldots, x_d; T) = q(y_1, \ldots, y_d) \prod_{i=1}^{d} \frac{\partial T_i}{\partial x_i}. \tag{5.7}$$

Their quotient is simply

$$p(x_d | x_1, \ldots, x_{d-1}; T) = q(y_d | y_1, \ldots, y_{d-1}) \frac{\partial T_d}{\partial x_d}. \tag{5.8}$$

Furthermore, since we defined $q(\mathbf{y})$ as the standard multivariate normal distribution, (5.8) can be reduced to

$$p(x_d | x_1, \ldots, x_{d-1}; T) = q(y_d) \frac{\partial T_d}{\partial x_d}, \tag{5.9}$$

where $y_d = T_d(x_1, \ldots, x_d)$ and $q(y_d)$ is a one-dimensional normal distribution. Thus, T_d fully determines how we model the conditional $p(x_d | x_1, \ldots, x_{d-1})$. This important property enables extracting marginals and conditionals once T is learned. We will use this property to build desired clique conditional samplers on the Bayes tree in Section 5.3.2.

Property 3: The normalizing flow T provides the following simple procedure for generating samples from $p(\mathbf{x}; T)$:

1. Draw samples $\mathbf{y} \sim q(\mathbf{y})$;

2. Solve for $\mathbf{x}$ by inverting T, i.e.,

$$\mathbf{x} = T^{-1}(\mathbf{y}) = \begin{bmatrix} T_1^{-1}(y_1) \\ T_2^{-1}(y_2; x_1) \\ \vdots \\ T_D^{-1}(y_D; x_1, x_2, \ldots, x_{D-1}) \end{bmatrix}. \tag{5.10}$$

In the first step, we can directly draw samples from the standard multivariate normal distribution $q(\mathbf{y})$. Since T is lower triangular, in the second step one can use a forward substitution-type approach to solve for elements of $\mathbf{x}$ one by one. We denote $x_{<d} = (x_1, x_2, \ldots, x_{d-1})$. Specifically, $x_{<d}$ is solved before x_d so $x_{<d}$ can determine the one-dimensional function $T_d(\cdot\, ; x_{<d})$ that maps x_d to y_d. Since T_d was constructed to be invertible with respect to x_d, x_d can be solved by evaluating the inverse function of $T_d(\cdot\, ; x_{<d})$ at y_d. Computation of the inverse can be efficiently done numerically or analytically, depending on the specific parameterization of T_d [76, 101]. Thus, a sample of $\mathbf{y}$ can be transformed to a sample of $\mathbf{x} \sim p(\mathbf{x}; T)$ by computing components $x_1, \ldots, x_D$ recursively.

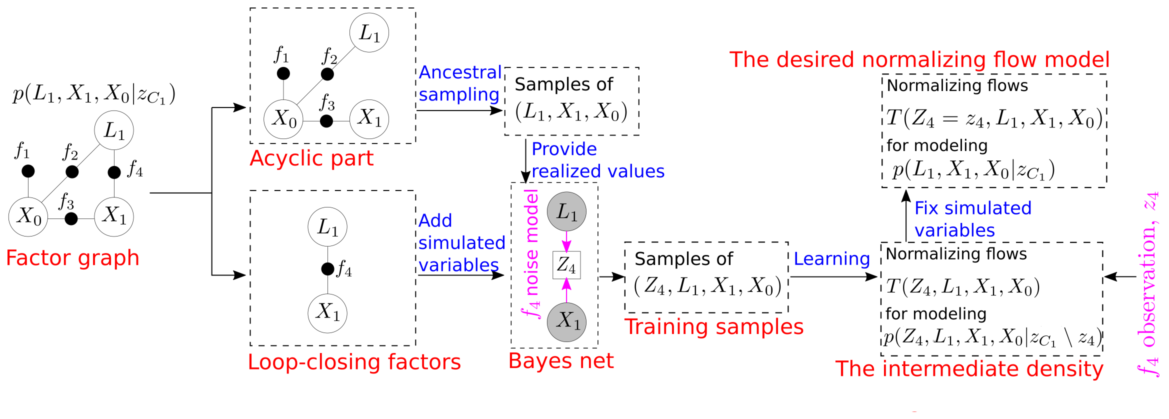
Optimal Normalizing Flow: It remains to explain how the triangular map T in (5.3) can be obtained. Given n i.i.d. training samples $\{\mathbf{x}^{(k)}\}_{k=1}^{n}$ from $p(\mathbf{x})$, we find T by minimizing the Kullback–Leibler (KL) divergence between $p(\mathbf{x})$ and $p(\mathbf{x}; T)$. In the following section (Sec. 5.3.2), we will present a simulation-based method for obtaining these training samples in the context of factor graph inference; in this section, we only focus on the procedure from training samples to normalizing flows in Fig. 5-1b. Assuming training samples are given, an optimal triangular map $T^{\star}$ is given by,

$$
\begin{aligned}
T^\star &\in \operatorname*{argmin}_{T\in\mathfrak{T}} D_{\mathrm{KL}}\left(p(\mathbf{x}) \parallel p(\mathbf{x};T)\right) && (5.11)\\
&= \operatorname*{argmin}_{T\in\mathfrak{T}} \int_{\mathbf{x}} p(\mathbf{x}) \log \frac{p(\mathbf{x})}{p(\mathbf{x};T)} \,\mathrm{d}\mathbf{x} && (5.12)\\
&= \operatorname*{argmin}_{T\in\mathfrak{T}} \int_{\mathbf{x}} -p(\mathbf{x}) \sum_{d=1}^{D} \left[\log q(T_d) + \log \frac{\partial T_d}{\partial x_d}\right] \mathrm{d}\mathbf{x} && (5.13)\\
&\approx \operatorname*{argmin}_{T\in\mathfrak{T}} -\sum_{k=1}^{n}\sum_{d=1}^{D} \left[\log q(T_d) + \log \frac{\partial T_d}{\partial x_d}\right]\Bigg|_{\mathbf{x}=\mathbf{x}^{(k)}} && (5.14)\\
&= \operatorname*{argmin}_{T\in\mathfrak{T}} \sum_{k=1}^{n}\sum_{d=1}^{D} \left(\frac{1}{2}T_d^2 - \log \frac{\partial T_d}{\partial x_d}\right)\Bigg|_{\mathbf{x}=\mathbf{x}^{(k)}}, && (5.15)
\end{aligned}
$$

where (5.13) follows from (5.9) and also used the training samples to approximate the expectation by Monte Carlo integration. $\mathfrak{T}$ denotes all triangular maps defined by (5.3). If the reference distribution is log-concave (e.g., Gaussian distributions here), the cost function from (5.14) is convex. As the feasible set is also convex, (5.15) turns out to be a convex optimization problem [6, Lemma 10]. Interested readers can find more discussion in [88, 75, 102]. While (5.15) offers a theoretical pathway for finding an optimal T, it is not practical to solve (5.15) directly as it requires searching among all admissible maps. To make this practical, we limit our search to admissible maps in a parameterized family of transformations $\mathcal{F} \subset \mathfrak{T}$,

$$
T^\star \in \operatorname*{argmin}_{T\in\mathcal{F}} \sum_{k=1}^{n}\sum_{d=1}^{D} \left(\frac{1}{2}T_d^2(\mathbf{x}^{(k)}) - \log \frac{\partial T_d(\mathbf{x}^{(k)})}{\partial x_d}\right). \qquad (5.16)
$$

The parameterization choice determines whether or not the above optimization problem is convex. For example, if we let $T_1(x_1) = ax_1 + b$ for a one-dimensional problem (i.e., $D = 1$), we can examine derivatives of the cost function to derive a close-formed globally optimal solution under the parameterization, $a^\star = 1/\sigma_{x_1}$ and $b^\star = -\bar{x}_1/\sigma_{x_1}$, where $\bar{x}_1$ and σ_{x_1} are empirical mean and standard deviation of x_1 samples. Obviously this affine transformation is not expressive enough for modeling non-Gaussian densities. Many flexible parameterizations of T_d have been proposed including polynomial



$p(L_1, X_1, X_0|z_{C_1})$
Factor graph
Acyclic part
Loop-closing factors
Ancestral sampling
Samples of (L_1, X_1, X_0)
Provide realized values
Add simulated variables
f_4 noise model
Bayes net
Samples of (Z_4, L_1, X_1, X_0)
Training samples
Learning
Normalizing flows $T(Z_4, L_1, X_1, X_0)$ for modeling $p(Z_4, L_1, X_1, X_0|z_{C_1} \setminus z_4)$
The intermediate density
f_4 observation, z_4
Fix simulated variables
Normalizing flows $T(Z_4 = z_4, L_1, X_1, X_0)$ for modeling $p(L_1, X_1, X_0|z_{C_1})$
The desired normalizing flow model

Figure 5-4: Flowchart for the generation of training samples and the desired normalizing flow. This factor graph is the C_1 factor graph in Fig. 5-2. The collection z_{C_1} denotes all measurements in this leaf clique. Z_4 and z_4 denote the measurement variable and value in factor f_4, respectively.

expansions [75, 38], sum-of-square polynomials [66], and splines [37]. T_d can be modeled as a one-dimensional function of x_d, $T_d(x_d; c_d(x_{<d}; \mathbf{w}_d))$, where $c_d(x_{<d}; \mathbf{w}_d)$ is the so-called conditioner network [101]. The conditioner network with weights $\mathbf{w}_d$ takes $x_{<d}$ as the input and then outputs a set of parameters such as polynomial coefficients or spline segments that determine a differentiable, bijective, and increasing function of x_d under the chosen parameterization [66, 37]. However, except for special cases like Gaussian conditionals, it is difficult to analyze and model the conditioner and then solve for its weights since T_d essentially encodes a high-dimensional conditional. Neural networks have been widely employed as universal functional approximators of conditioners in normalizing flows [76, 101]. In practice, one optimizes over all weights $\mathbf{w}_{<D+1}$ for a solution to (5.16) once a parameterization method and network configuration are designed; see Sec. 5.4.1 for our parameterization method and neural network configuration. It is an active research topic to construct a parameterization method that possesses convexity for leveraging the convex problem (5.15). Interested readers can find further discussions in [3].

Practical Considerations: A usual routine before training is standardizing raw samples by their means and standard deviations to regularize unbounded large values [65]. This standardizing step is equivalent to an affine transformation which makes training more efficient and does not alter the problem nor affect the non-Gaussianity in the raw samples. When the training is finished, the resulting triangular map will be transformed back to the space of raw samples by the inverse of the affine transformation. Samples of orientation variables in our SLAM experiments are transformed to $[-\pi, \pi]$ before being standardized since the experiments are treated in a planar environment; for samples in more general manifolds (e.g., SO(3)), one should resort to alternative standardizing methods (e.g., in vector spaces such as $\mathfrak{so}(3)$). Recently, more sophisticated treatments for orientation have been proposed for normalizing flows, which improves the robustness and expressive power of density estimation on complex manifolds [109]. As a widely used alternative of autoregressive flows, coupling flows impose additional structures on the triangular map at the expense of reduced expressive power, leading to improved efficiency in modeling and training normalizing flows [101]. While there are so many different normalizing flow parameterizations, we stress that the inference framework of NF-iSAM is generalizable as it is mostly governed by the Bayes tree and the triangular map structure. As what we will depict in the flowchart of Fig. 5-4, learning for a model of density just takes a small fraction in the flowchart. We emphasize that the specific parameterization for modeling a complex density is a replaceable part in the pipeline. Thus, more parameterization methods can be explored and exploited in future work for efficiency, expressiveness, and robustness in density modeling.

5.3.2 Clique conditional samplers via normalizing flows

This subsection focuses on learning the clique density $p(S_C, F_C|z_C)$ in (5.2). We will describe in detail how to learn a normalizing flow model of the clique density from which we can extract the separator density $p(S_C|z_C)$ and the clique conditional $p(F_C|S_C, z_C)$.

As suggested in Section 5.3.1, we can learn the normalizing flow

$$T_C(S_C, F_C) = \begin{bmatrix} T_{S_C}(S_C) \\ T_{F_C}(S_C, F_C) \end{bmatrix} \tag{5.17}$$

for the clique density $p(S_C, F_C|z_C)$ if we have training samples from $p(S_C, F_C|z_C)$. Then T_{S_C} is the normalizing flow for the separator density $p(S_C|z_C)$, and T_{F_C} is the normalizing flow for the clique conditional $p(F_C|S_C, z_C)$. There are many well-developed off-the-shelf implementations of MCMC sampling such as PyMC3 [114] or nested sampling such as dynesty [123]. However, even though variables in a clique are much fewer than those in the entire Bayes tree, those packages are still too slow for generating the training samples for robotics applications [63].

Inspired by the so-called forecast-analysis scenario in hidden Markov models [122] and simulating from a Bayes net model [31, Sec. 1.5], we propose the following two-step strategy for modeling the clique density $p(F_C, S_C|z_C)$:

Step 1. Draw training samples from an intermediate density $\widetilde{p}$, where sampling is efficient.

Step 2. Train normalizing flow $\widetilde{T}$ for $\widetilde{p}$, and retrieve T for $p(S_C, F_C|z_C)$.

In *Step* 1, we sample from the intermediate density $p(O_C, S_C, F_C|z'_C)$, where $z'_C = z_C \setminus o_C$. We can select a set of likelihood factors, whose measurements are o_C, that breaks the factor graph of the clique into an acyclic factor graph (for example, see f_4 in Fig. 5-4). We define these likelihood factors as loop-closing factors and convert them to Bayes nets where measurements are assumed as unobserved variables O_C (see [31, Sec. 1.7] for the recipe and the probabilistic interpretation of the conversion). Since both Bayes nets and the acyclic factor graph afford ancestral sampling, one can use ancestral sampling and measurement models to efficiently simulate samples of (O_C, S_C, F_C) which are distributed according to the intermediate density.

Algorithm 7 is our implementation of *Step* 1 for SLAM problems. In the algorithm, most of the loop-closing factors can be passively identified when we simulate samples. The prior factors $\mathcal{P}$ in the algorithm refer to either user-defined normalizable densities (e.g., the density of the first robot pose), from which we are typically able to draw samples directly, or separator densities modeled by normalizing flows, which enjoy fast sampling as well (*Property 3*, Sec. 5.3.1). Starting from samples in these priors (line 2), we iterate over binary factors to simulate other robot pose and landmark samples (line 6). If both variables adjacent to a binary factor have been sampled, virtual observations between samples of these variables will be simulated (line 8). All multi-modal data association factors are proactively treated as loop-closing factors for simulating measurements (line 14).

In *Step* 2, we use training samples from the intermediate density $p(O_C, S_C, F_C | z'_C)$ to learn the normalizing flow $\widetilde{T}_C$ for eventually modeling $p(S_C|z_C)$ and $p(F_C|S_C, z_C)$. According to Section 5.3.1, by ordering arguments in $\widetilde{T}_C$ to (O_C, S_C, F_C), we get the triangular map

$$\widetilde{T}_C(O_C, S_C, F_C) = \begin{bmatrix} \widetilde{T}_{O_C}(O_C) \\ \widetilde{T}_{S_C}(O_C, S_C) \\ \widetilde{T}_{F_C}(O_C, S_C, F_C) \end{bmatrix}. \tag{5.18}$$

When we fix O_C to its measured value o_C, $\widetilde{T}_{S_C}(O_C = o_C, S_C)$ gives the normalizing flow for the separator density $p(S_C|z_C)$, and $\widetilde{T}_{F_C}(O_C = o_C, S_C, F_C)$ gives the normalizing flow for the clique conditional $p(F_C|S_C, z_C)$ (see Algorithm 8). Thus, we can retrieve the desired normalizing flow model T_C from $\widetilde{T}_C$:

$$T_C(S_C, F_C) = \begin{bmatrix} \widetilde{T}_{S_C}(O_C = o_C, S_C) \\ \widetilde{T}_{F_C}(O_C = o_C, S_C, F_C) \end{bmatrix}, \tag{5.19}$$

which models the clique density $p(F_C, S_C|z_C)$.

5.3.3 Incremental inference on Bayes tree

Learning the full posterior distribution will start from leaf cliques C_L. By Section 5.3.2, we can learn normalizing flows for the separator density $p(S_{C_\mathrm{L}}|z_{C_\mathrm{L}})$ and the clique conditional $p(F_{C_\mathrm{L}}|S_{C_\mathrm{L}}, z_{C_\mathrm{L}})$. The separator density will be passed to the parent of clique C_L as a new factor as shown in Fig. 5-2 (i.e., $p(X_1, L_1|z_{C_1})$ and $p(X_2, L_1|z_{C_2})$). The normalizing flow for the clique conditional will be saved in clique C_L as a conditional sampler for sampling the joint posterior later. Our algorithm learns all normalizing flows during a single leaf-to-root traversal on the Bayes tree. The product of learned clique conditionals resolves *Task* 1 in Sec. 5.1.

As we described in Sec. 5.2.2, once all cliques have learned their conditional samplers of clique conditionals $p(F_C|S_C, z_C)$, we can draw components of a joint posterior sample from these conditional samplers by recursively applying ancestral sampling during a root-to-leaf traversal on the Bayes tree. Through the root-to-leaf traversal,

Algorithm 7: TrainingSampleSimulator

```
Input: Prior $\mathcal{P}$, binary measurement $\mathcal{B}$, and multi-modal data association
       $\mathcal{M}$ factors in a clique
Output: Samples and measured values
1  Initialize samples $\mathcal{S}$ and measured values $\mathcal{V}$ dictionaries
2  $\mathcal{S}[\Theta_i] \leftarrow$ Sample any variable $\Theta_i$ in $\mathcal{P}$
3  while queue $\mathcal{B} \neq \emptyset$ do
4      $f_i = p(z_i|\Theta_j, \Theta_k) \leftarrow$ Pop the first element in $\mathcal{B}$
5      if only one latent variable (e.g., $\Theta_k$) not in $\mathcal{S}$ then
6          $\mathcal{S}[\Theta_k] \leftarrow$ Simulate variable $\Theta_k$ using sample $\mathcal{S}[\Theta_j]$, measured value $z_i$,
            and measurement models
7      else if both latent variables $\Theta_j, \Theta_k$ in $\mathcal{S}$ then
8          $\mathcal{S}[Z_i] \leftarrow$ Simulate measurement $Z_i$ between samples $\mathcal{S}[\Theta_j]$ and $\mathcal{S}[\Theta_k]$
            using measurement models
9          $\mathcal{V}[Z_i] \leftarrow$ Measured value $z_i$
10     else
11         Push $f_i$ to the back of $\mathcal{B}$
12 for $f_i = p(z_i|\Theta_{f_i})$ in $\mathcal{M}$ do
13     $\mathcal{S}[Z_i] \leftarrow$ Simulate measurement $Z_i$ between samples $\mathcal{S}[\Theta_{f_i}]$ using
        measurement models
14     $\mathcal{V}[Z_i] \leftarrow$ Measured value $z_i$
15 return Samples $\mathcal{S}$, measured values $\mathcal{V}$
```

Algorithm 8: ConditionalSamplerTrainer

```
Input: Training samples and measured values $o$
Output: Clique conditional, separator density
1  Rearrange training samples to the order of observation ($O$), separator ($S$),
    and frontal variables ($F$)
2  Find $\widetilde{T}$ in (5.18) by minimizing the KL divergence according to (5.15) using
    the training samples
3  $T(S, F) \leftarrow \widetilde{T}(O = o, S, F)$                    // fix observations in (5.19)
4  $T_S, T_F \leftarrow$ partition $T(S, F)$ following (5.17)
5  Obtain samplers of $p(F|S)$, $p(S)$ from $T_S$ and $T_F$ by (5.10)
6  return Samplers of $p(F|S), p(S)$
```

Task 2 in Sec. 5.1 can be accomplished. Compared to learning normalizing flows during the leaf-to-root traversal, the computational cost of the root-to-leaf traversal is minimal since normalizing flows support fast sampling (*Property 3*, Sec. 5.3.1).

When performing incremental updates, we do not need to recompute normalizing flows for all cliques of the Bayes tree. Every time a new factor is added into the factor graph, the corresponding change in the Bayes tree is an exact and symbolic result from the Bayes tree algorithm [71, Alg. 3]. We just need to learn normalizing flows for cliques in the changed part to update posterior estimation (see clique C_3 in Fig. 5-2 for an example of incremental inference). We designate the changed part of the Bayes tree the sub-tree. The upward traversal starts from leaves of the sub-tree instead of the entire Bayes tree. Normalizing flows for cliques outside the sub-tree are not changed and can be reused directly. Thus, the computational cost for *incrementally* training normalizing flows depends only on the *sub-tree*, instead of the entire problem. To draw samples from the full joint posterior density, the downward traversal still needs to visit all cliques. However, as mentioned above, the computational cost for the downward sampling traversal is much lower than that for training normalizing flows. The detailed algorithm of NF-iSAM is summarized in Algorithm 9. At this point, all *Task* 1-*Task* 3 we proposed in Sec. 5.1 have been resolved. With the

```
Algorithm 9: NF-iSAM
   Input: New factors $\mathbf{f}$, factor graph $\mathcal{G}$, ordering $\boldsymbol{\Theta}$
   Output: Samples of the joint posterior distribution
 1 $\mathcal{T} \leftarrow \mathcal{G}$.update($\mathbf{f}, \boldsymbol{\Theta}$)                    // update the Bayes tree
 2 $\mathcal{T}_\Delta \leftarrow \mathcal{T}$.extract($\mathbf{f}, \boldsymbol{\Theta}$)         // extract the changed sub-tree of $\mathcal{T}$
 3 for clique $C$ in leaf-to-root traversal of $\mathcal{T}_\Delta$ do
 4     $\mathbf{x}$, $o \leftarrow$ TrainingSampleSimulator($C$)
 5     $p(F_C|S_C), p(S_C) \leftarrow$ ConditionalSamplerTrainer($\mathbf{x}$, $o$)
 6     Append $p(S_C)$ to the parent clique as a factor
 7 $\mathcal{D} \leftarrow \{\}$          // initialize an empty dictionary for posterior samples
 8 for clique $C$ in root-to-leaf traversal of $\mathcal{T}$ do
 9     $p(F_C|S_C) \leftarrow$ retrieve the conditional sampler in $C$
10     $s \leftarrow \mathcal{D}[S_C]$                               // retrieve samples of separator
11     $\mathcal{D}[F_C] \leftarrow$ draw samples from $p(F_C|S_C = s)$ using (5.10)
12 return $\mathcal{D}$
```

exception of normalizing flows for modeling non-Gaussian conditionals, our strategy for incremental updates is similar to iSAM2 where linear-Gaussian conditionals for the Gaussian approximation are partially updated as new measurements arrive. The back-substitution in iSAM2 for the least-squares solution also corresponds to a root-to-leaf traversal on the entire Bayes tree [31, Sec. 5.4.3].

Although the final output of our algorithms is samples for subsequent inference tasks requiring Monte Carlo integration, function evaluation of the approximate distribution can be conducted as well. While we wrap trained normalizing flows in the name of "sampler" in Algorithms 8 and 9, the density modeled by normalizing flows can be easily evaluated by (5.5). Instead of drawing samples at line 10 to line 11 in Algorithm 9, we can simply evaluate the conditionals modeled by normalizing flows and then return their product as the function evaluation of the approximate density of the joint posterior.

5.4 Implementation and experimental setup

5.4.1 Current implementation of NF-iSAM

We implemented Algorithms 7-9 as well as other building blocks including prior and measurement likelihood factors, factor graphs, and the Bayes tree in Python. In the current implementation of normalizing flows, we choose rational-quadratic (RQ) splines to parameterize the one-dimensional function of x_d, $T_d(x_d; c_d(x_{<d}; \mathbf{w}_d))$ [37], considering the flexibility of splines. A fully connected neural network (FCNN) with an input dimension $d-1$ is configured to model the conditioner network $c_d(x_{<d}; \mathbf{w}_d)$. If the spline is composed of K different rational-quadratic functions, the conditioner network possesses an output dimension $3K-1$ of which $2K-2$ units map to coordinates of spline knots on the x_d-y_d plane and $K+1$ units are for spline derivatives at those knots; see *RQ-NSF (AR)* in [37, Sec. 3] for the detailed parameterization. Our RQ spline flows are constructed and trained using PyTorch [103]. For the experiments in Sec. 5.5, we use two-layer FCNNs for every conditioner network. The

default number of hidden layer units in the FCNN is set to 8 and the default number of RQ splines, K, is set to 9. The default number of training samples is set to 2000. While we had tried some stopping criteria in [106] using test sets, for less training time, the training of the FCNNs stops when the loss (5.16) converges. We monitor the relative change between the average loss over the latest 50 iterations and that over the second latest 50 iterations. The ongoing training is terminated once the relative change is lower than 1%.

It is worth mentioning that [76, Table 3] has reported that spline-based flows outperform or are on par with other normalizing flows in terms of modeling accuracy. Another advantage of RQ splines is that it can be inverted by evaluating an analytically exact expression, which permits a fast solution to the inverse transformation problem (5.10) for sampling.

5.4.2 Other solvers and computation resources

We use a nested-sampling-based approach for factor graphs, NSFG [63], to obtain high-quality samples for some examples in Sec. 5.5 as reference solutions. NSFG is implemented in Python based on the dynamic nested sampling package, dynesty[123]. iSAM2 (provided by the GTSAM library in C++ [28]) and mm-iSAM (provided by Caesar.jl, v0.10.2 in Julia [22]) are tested in our experiments as well. Experiments are performed on a workstation with an AMD Ryzen ThreadRipper 3970X CPU with 32 cores and 64 threads, an NVIDIA RTX 3090 GPUs, and 125.7 GB of RAM running Ubuntu 20.04.1 LTS. Only the neural network training in NF-iSAM uses the GPU while other solvers and other computation in NF-iSAM rely on the CPU in our experiments.

5.4.3 Datasets and measurement likelihood models

In the following experiments, we use three simulated datasets and two real-world datasets for range-only SLAM problems with and without data association ambiguity. We apply a unified variable elimination ordering in Algorithm 9 when solving these datasets: eliminating poses along the robot trajectory first and then landmarks. The extensive experimental results form a parameter study that investigates how the performance of NF-iSAM is affected by conditions such as: i) the magnitude of measurement noise, ii) hyperparameters in normalizing flows, iii) the fraction of factors involving data association ambiguity, iv) the randomness of robot trajectories, v) random seeds for the algorithm, and vi) the dimensionality of the SLAM problems.

We define three types of likelihood models for measurements in the datasets. First, the measurement of transformation between robot poses T_i^w and $T_j^w \in \mathrm{SE}(d)$ is modeled as $\tilde{T}_i^j = T_w^j T_i^w \exp(\boldsymbol{\xi}^\wedge)$ where $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \Sigma)$ and T_i^w reads pose i in the world frame. Second, the range measurement between a robot pose T_i^w and a landmark location $\mathbf{l}_j^w$ with known data association is modeled as $\tilde{r}_i^j = \|\mathbf{t}_i^w - \mathbf{l}_j^w\|_2 + \xi$ where $\mathbf{t}_i^w$ denotes the translation vector in T_i^w and $\xi \sim \mathcal{N}(0, \sigma^2)$. Third, the range measurement with unknown data association, $\tilde{r}_i$, is modeled as $p(\tilde{r}_i | \mathbf{t}_i^w, \mathcal{L}_i) = \frac{1}{|\mathcal{L}_i|} \sum_{\mathbf{l}_j^w \in \mathcal{L}_i} p(\tilde{r}_i | \mathbf{t}_i^w, \mathbf{l}_j^w)$ where $\mathcal{L}_i$ denotes the set of possibly associated landmarks. Each component in the sum-mixture is simply a likelihood model of range measurement with a given data association. All components in the mixture are equally weighted given no prior information about data associations.

5.5 Results

5.5.1 Synthetic datasets

A small illustrative example

A small example is employed to illustrate capacities and performance of NF-iSAM on non-Gaussian inference (Fig. 5-5 and 5-6). We create a 2D environment where a robot performs a range-only SLAM task using odometry and range measurements to landmarks. A large fraction of the range measurements, however, have no identity information of landmarks, which implies that each ambiguous range measurement can (potentially) be associated with *all* landmarks. While this problem is relatively low dimensional, it is nevertheless still difficult to infer the posterior distribution of robot and landmark positions since the problem involves both nonlinear measurements (e.g., distance) and high-uncertainty non-Gaussian likelihood models (due to the multi-modal data association).

Fig. 5-5 and 5-6 show samples that are drawn from estimated posteriors of problems without and with data association ambiguity. The runtime and accuracy are shown in Fig. 5-7. In both cases, the robot moves from X_0 to X_5 following a clockwise trajectory during which it acquires five odometry measurements and eight distance measurements. In the case with ambiguity, however, distance measurements from $X_{\{1-4\}}$ are modeled as potentially associated with *all* detected landmarks with equal weights. Note that “No solution" tags in the figures indicate that we could not obtain a solution due to errors thrown by GTSAM.

Incorporating range measurements already presents a challenge due to strong nonlinearity, so we analyze the case without data association ambiguity first (see Fig. 5-5). According to the scatter plots and kernel density estimation, the solutions of NF-iSAM resemble reference solutions provided by NSFG for all steps. When a landmark owns two distinct distance measurements (e.g., landmark L_1 at time step 1 and landmark L_2 at time step 3), both NF-iSAM and NSFG are able to infer the bi-modal distribution of the landmark's location. Furthermore, uni-modal posterior distributions of the landmark are immediately recovered by them once the robot sights the landmark from three different poses (see landmark L_1 at time step 2 and landmark L_2 at time step 4). Caesar.jl was developed to estimate multi-modal marginal posteriors. Even though Caesar.jl pinpoints the landmarks at the last step, the uncertainty estimates are less accurate. Moreover, its estimation for earlier steps preserves many less-likely modes, which will certainly introduce errors in the evaluation of empirical mean as well as uncertainty. GTSAM leverages nonlinear least-squares (NLLS) optimization techniques to resolve Gaussian approximations of posterior distributions. For a newly detected landmark, we randomly pick a point on the circle projected by the range measurement from a robot pose, and supply it to GTSAM as the initial value of the landmark. At early steps, it cannot return a solution since the information matrix for NLLS is under-determined due to insufficient constraints. At step 3 and 4 even when each landmark possesses at least three distinct measurements, considerable drifts from the ground truth still exist as the NLLS optimization is subject to local optima in this non-convex optimization problem.

As shown in Fig. 5-7a, our quantitative analysis of this case follows the qualitative analysis above. We use the root-mean-square error (RMSE) to gauge the difference between the empirical mean of our posterior samples and the ground truth. We choose to compute the empirical mean rather than an MAP point among samples since the empirical mean can reflect errors incurred by spurious modes in estimated distributions. Here, since we aim to infer the full posterior distribution instead of a point estimate, *maximum mean discrepancy (MMD)* [52] is actually a more reasonable choice. Given samples from two densities, MMD is a metric to evaluate how far the two distributions are apart. Therefore, a lower MMD from a solution to the NSFG solution indicates a more "accurate" approximation of the posterior. As the reference solver, the RMSE of NSFG outperforms others at the expense of computation time. Note that before time step 4, the landmark belief is supposed to be bi-modal or donut-shaped distributions so it is reasonable to have large RMSE at those time steps. The plot of RMSE of NF-iSAM follows the same trend as that of NSFG and it is noticeably lower than mm-iSAM and GTSAM. We extract samples from joint and marginal distributions to compute the joint MMD and the marginal MMD respectively. The MMD plots indicate the superior accuracy of NF-iSAM in capturing the *entire* "shape" of the true posterior. The lower MMD of landmark L_1 from GTSAM at time step 3 and 4 is a coincidence as the random initial value of landmark L_1 happens to be around the ground truth (see green dots in scatter plots of GTSAM). However, the initial value of landmark L_2 unluckily locates away from the ground truth so the final estimate of GTSAM is inevitably distorted, resulting in the large MMD of landmark L_1 at the final time step.

Ambiguous data association of range sensing makes the estimation problem even more difficult, as shown in the highly uncertain posteriors of NSFG solutions in Fig. 5-6. At time step 2, two of the three distance measurements to landmark L_1 are associated with L_2 as well, leading to a more uncertain distribution of landmark

L_1 than the counterpart in the ambiguity-free case. The uni-modal distributions of landmark L_1 and L_2 are not resolved until new ambiguity-free measurements added at the last step. NF-iSAM precisely captures the same trend in all scatter, kernel density estimation (KDE), and RMSE plots. The MMD plots in Fig. 5-7b indicate that NF-iSAM consistently infers more accurate estimates of the true posterior than other solvers in the setting with ambiguity. Note that iSAM2 provided by GTSAM is extended with max-mixture factors [100, 35] for dealing with multi-modal data association so GTSAM is replaced by max-mixtures in the legend.

Given samples from the posterior distribution of the robot and landmark positions, $p(\mathbf{\Theta}|\mathbf{z})$, one can evaluate the posterior belief of different data associations following

$$p(D|\mathbf{z}) = \int_{\mathbf{\Theta}} \frac{p(\mathbf{z}|\mathbf{\Theta}, D)p(D)}{\sum_{D\in\mathcal{D}} p(\mathbf{z}|\mathbf{\Theta}, D)p(D)} p(\mathbf{\Theta}|\mathbf{z}) \tag{5.20}$$

$$\approx \frac{1}{N}\sum_{i=1}^{N} \frac{p(\mathbf{z}|\mathbf{\Theta}=\boldsymbol{\theta}^{(i)}, D)p(D)}{\sum_{D\in\mathcal{D}} p(\mathbf{z}|\mathbf{\Theta}=\boldsymbol{\theta}^{(i)}, D)p(D)}, \tag{5.21}$$

where $\boldsymbol{\theta}^{(i)}$ is one of the N samples drawn from $p(\mathbf{\Theta}|\mathbf{z})$, and D is a possible data association in the set of all associations, $\mathcal{D}$. $p(D)$ is subject to a uniform distribution over $\mathcal{D}$ as we have no prior knowledge about those associations. $p(\mathbf{z}|\mathbf{\Theta}, D)$ is actually a binary factor under the association D so it is known when we formulate the problem.

Fig. 5-8 shows the posterior belief of groundtruth data associations so a good estimate should approach 1 as more measurements arrive. It is clear that both NF-iSAM and NSFG manage to identify true data associations eventually.

Medium-scale problems in the Manhattan world with range measurements

Here we simulate a variety of scenarios to investigate whether NF-iSAM performs consistently well over a range of settings such as different noise magnitudes, the fraction of measurements with ambiguous data associations, and robot trajectories. We consider these to be "medium-scale" problems since NSFG can still converge within tens of minutes and return samples of posteriors as reference solutions for the extensive parameter study. We implement a simulator named "Manhattan world with range measurements" to synthesize odometry and distance measurements along Manhattan-world-like trajectories (i.e., the robot operates in grid environments where it can only move to adjacent vertices by fixed step length). Fig. 5-9 shows a navigation task using range sensing along a lawnmower path in the simulator. Problems with random trajectories can be found in Fig. 5-13a. In all the cases, the robot starts from time step zero and then proceeds step-by-step until time step 15, during which three landmarks will be sighted. At each time step, a distance measurement is acquired with or without data association ambiguity. Hence, unknowns at the final step consist of 16 poses and three landmark positions, resulting in a 54-dimensional posterior distribution.

There are several settings related to noise magnitudes and the fraction of ambiguous range measurements for the lawnmower path experiment. The default standard deviation of range sensing noise is set to 2 meters while the default covariance of odometry noise is $diag(0.04, 0.0016, 0.0004)$ where the diagonal entries correspond to longitudinal, lateral, and heading measurements. The default probability for generating ambiguous data association factors is set to 40%. We are interested in investigating how those settings affect the performance of NF-iSAM and other solvers. Before varying and interrogating those settings, posterior samples for the default setting are presented in Fig. 5-9a to visualize the scenario and the solutions. The posterior distribution is resolved incrementally step-by-step. An interesting point in Fig. 5-9c is that the evidently linear curve of NSFG on the log-scale runtime plot indicates the exponential growth of computation time with increased dimensionality. On the other hand, while less accurate, other solvers are able to retain a roughly constant computation time per step by exploiting incremental inference techniques, which makes full posterior inference more tractable.

We conduct an empirical study over measurement noise and the fraction of ambiguous range measurements. Only one of the settings mentioned above varies for each point in Fig. 5-10. Runtime plots of max-mixtures are neglected as they are faster than others by at least two orders of magnitude in our experiments; however, solutions of max-mixtures deviate considerably from the ground truth due to bad initial values and local optima. The computation time of NF-iSAM is consistently lower than mm-iSAM (Caesar.jl) and the reference solution (NSFG) while its RMSE is almost at the same order of magnitude as the reference solution in various settings. The lower value of average MMD from NF-iSAM indicates that its estimated posterior distribution resembles the reference posterior distribution better than mm-iSAM and GTSAM, which demonstrates the superior accuracy of NF-iSAM for full posterior estimation in non-Gaussian settings.

The normalizing flow model in NF-iSAM is primarily characterized by two predetermined hyperparameters: the number of RQ functions on the spline for fitting the one-dimensional transformation map, and the number of hidden units in the fully connected neural networks that output locations and derivatives of the spline knots (see Sec. 5.4 for more about hyperparameters). The former one controls the flexibility of the spline while the latter one is the width of neural networks. Both of them have a great impact on the expressiveness of the normalizing flow model so it is worth investigating how they influence the solutions of NF-iSAM. It is not surprising to find that greater numbers of spline knots and hidden units in general lead to higher computation time as there are more parameters being trained in the neural networks (Fig. 5-11). Another considerable change is spotted in the plot of joint MMD versus the number of knots. More spline functions evidently allow for a finer fit to the shape of the posterior, decreasing the joint MMD between NF-iSAM and reference solutions. In contrast, these parameters overall present little influence on the RMSE, which implies that the enhanced expressiveness only marginally improves point estimates. The low sensitivity of RMSE also reflects our previous comment that MMD is a more reasonable choice for evaluating the quality of full posterior estimation. Furthermore, this implies an important fact that the accuracy evaluation simply using means or modes can be ineffective especially for non-Gaussian posteriors.

Because training samples are self-generated in NF-iSAM, the number of training samples (i.e., n in (5.16)) is a predetermined hyperparameter in NF-iSAM as well. Fig. 5-12 shows how the number of training samples affects the results for the lawnmower path problem. It is evident that small sample sizes (e.g., 500 and 1000 samples) cause less favorable performance in both computation time and accuracy.

This can be explained by the plot of training loss where fewer samples incur slower and more unstable convergence of training loss. An extremely large number of training samples increases both the total computation time and the fraction of the time for sample generation; furthermore, it does not greatly improve accuracy of the results. Further investigation is needed to design an adaptive strategy that determines good hyperparameters in the training.

Fig. 5-13 presents posterior samples and performance of different algorithms under randomly generated cases. NF-iSAM behaves robustly for those cases and provides solutions almost as accurate as the reference solutions while possessing superior scalability, due to the use of incremental updates.

Large-scale problems in the Manhattan world with range measurements

We demonstrate scalability of NF-iSAM and repeatability of its solutions given randomness in algorithms. We simulate a relatively large-scale range-only SLAM problem where the robot follows a path similar to that in the Plaza1 dataset [33]. The entire trajectory involves 136 robot poses, 4 landmarks, 135 odometry measurements, and 136 range measurements among which 59 measurements are associated with multiple landmarks (i.e., they will be modeled by multi-modal data association factors). The posterior of robot poses and landmark locations incurs a 416-dimensional latent variable at the end of the sequence, to which a reference solution via sampling techniques is generally not available. Hence, we only compute RMSE of NF-iSAM estimates versus ground truth as the accuracy metric.

Fig. 5-14a shows the posterior samples resolved by NF-iSAM at a few important time steps. The odometry and groundtruth trajectories are respectively shown in gray and black in the figure as well. At time step 6, the robot nearly moves along a line and, as a result, the belief of landmark locations inferred by distance measurements is subject to a distribution mirrored across the line being tracked by the robot. The highly uncertain distribution of landmark location results in the significant RMSE at time step 6 as shown in Fig. 5-14b. As the robot proceeds and turns left to time step 19, distance measurements acquired along the asymmetric trajectory can disambiguate landmark locations, thus the landmark distribution collapses to uni-modal. The steep decline of RMSE at time step 19 is a direct consequence of the disambiguation of landmark locations. Although the landmarks are basically pinpointed at this time step, the accumulative error in odometry can still incur inaccurate estimation. As seen around the bottom part of the groundtruth path, the odometry trajectory considerably deviates from the ground truth by more or less a block, visibly twisting the estimated trajectory. This is reflected by the sharp increase of RMSE at time step 39 as well. However, as shown in the estimated trajectory at time step 135, the deviation at the bottom of the trajectory is corrected via fusing the full-time history of measurements, which demonstrates the smoothing capacity of NF-iSAM.

In practice NF-iSAM learns probability density functions from a finite number of training samples via stochastic optimization (see Algorithms 7 and 8), so it is not a deterministic algorithm. Therefore, it is necessary to check if NF-iSAM can achieve consistent results in the presence of inherent randomness in algorithms. As seen in Fig. 5-14b, the width of the error band of RMSE is comparatively much smaller than the mean of RMSE. Thus, the accuracy of NF-iSAM is marginally affected by randomness in algorithms.

5.5.2 Real-world datasets

We also evaluate the scalability and error of NF-iSAM using a larger real SLAM dataset. The Plaza dataset provides time stamped range and odometry measurements $(\delta x, \delta\theta)$ of a vehicle moving in a planar environment [33]. Two of its sequences, Plaza1 and Plaza2, are available in the GTSAM software distribution. There are four unknown landmarks in each of the sequences. As noted in [33], the error of distance measurement is strongly correlated with the true distance, leading to a non-zero mean in the distribution of errors. Therefore, we use least squares to fit an affine function characterizing the relation between the measurement error and the true distance. We then compute calibrated distances via subtracting the affine function from measured distances. This is a valid calibration process. If we were provided with the same sensor, we could make a collection of range measurements independently, and fit the linear model using our own data. As indicated by the histograms in Fig. 5-15a, the error of the calibrated distances obeys zero-mean Gaussian distributions well and no evident outliers appear in the datasets. To tackle datasets where outliers present, the Gaussian noise model in Sec. 5.4.3 can be replaced by outlier-robust distributions (e.g., a mixture model with a null hypothesis that the measurement is wrong [100]). NF-iSAM is applicable to those situations since there is no assumption on noise models in our algorithms (Algorithms 7-9).

The range-only dataset is challenging for the state-of-the-art SLAM techniques (e.g., iSAM2) that rely on the Gaussian approximation obtained by linearization around an MAP estimate. Since range-only SLAM is a highly non-convex problem and good initial values are usually not available in advance, those techniques are prone to find local optima. For a newly detected landmark, we randomly pick a point on the circle projected by the range measurement, and supply it to GTSAM as the initial value of the landmark. The GTSAM solutions in Fig. 5-15c clearly show that range-only measurements pose difficulties for the MAP estimation especially when a good initialization is not available.

As shown on the leftmost side in Fig. 5-15b, NF-iSAM can solve both sequences and return accurate estimates on robot trajectories and landmark positions. Moreover, we consider data association ambiguity in these datasets and evaluate the performance of NF-iSAM for multi-modal data association problems. Note that we randomly choose a faction of range measurements, wipe the ID information of the detected landmark, and designate the measurement to associate with all landmarks. The rightmost end in Fig. 5-15b is the most challenging case where 60% of range measurements are acquired with no landmark information such that they are formulated by multi-modal factors. The estimated trajectories of NF-iSAM resemble the ground truth well in all the cases with data association ambiguity. As seen in Fig. 5-16a, although a higher fraction of data association ambiguity causes a higher RMSE, the RMSE is still at the same order of magnitude as that with no data association ambiguity. We also provide the profiling of NF-iSAM's runtime in Fig. 5-16b. The time for sample generation and subsequent training remains roughly even across key poses since the dimension of variables on the sub-tree for incremental updates is fairly consistent. As we mentioned in Sec. 5.3.3, sampling posteriors takes much shorter time than training normalizing flows. While posterior sampling is fast, it runs through the entire Bayes tree. Thus, the time for posterior sampling increases with the dimensionality of the joint posterior. To alleviate this issue for larger-scale problems, one may consider to control the size of the Bayes tree via strategies including fixed-lag smoothing[31, Sec. 5.3.2].

5.6 Summary

We presented a novel algorithm, NF-iSAM, that provides a promising foundation for estimating the full posterior distribution encountered in SLAM. NF-iSAM utilizes the Bayes tree coupled with normalizing flows to achieve efficient incremental updates in non-Gaussian distribution estimation of the full posterior. We demonstrated the advantages of the approach over alternative state-of-the-art point and distribution estimation techniques for SLAM, with synthetic datasets and real datasets. Our approach showed an improved estimate of the full posterior in highly non-Gaussian settings due to nonlinear measurement models and non-Gaussian (e.g., multi-modal) factors. Currently, for real SLAM problems with non-Gaussian and nonlinear models, our approach can be used to i) understand how the posterior distribution evolves over time, ii) provide reference estimates to approximate distributions found by other estimation techniques, and iii) perform inference tasks that require an estimate of the full posterior, e.g., estimating the posterior belief of data association or various expectations with respect to the posterior.

We conclude the paper by noting that NF-iSAM warrants further research as a promising and generalizable algorithmic framework. Its generalizability includes two aspects: i) the parameterization of transformation maps in normalizing flows can be replaced by other forms ii) and, more significantly, normalizing flows can be replaced by other probabilistic modeling techniques. For instance, if restricting the parameterization to affine transformations or the probabilistic modeling to Gaussian, our approach can recover iSAM2 as a special case. On the practical side, our experiments employ complex transformations to express non-Gaussian distributions, which in part incurs greater computation cost than iSAM2. Further research is needed to explore more efficient implementation strategies that can lead us closer to real-time operation for higher dimensional problems (e.g., 3D SLAM), including: (1) leveraging more efficient density modeling techniques such as coupling flows [101] and (2) utilizing faster incremental update strategies on the Bayes tree such as marginalization operations and variable elimination orderings with heuristics [31, 44].

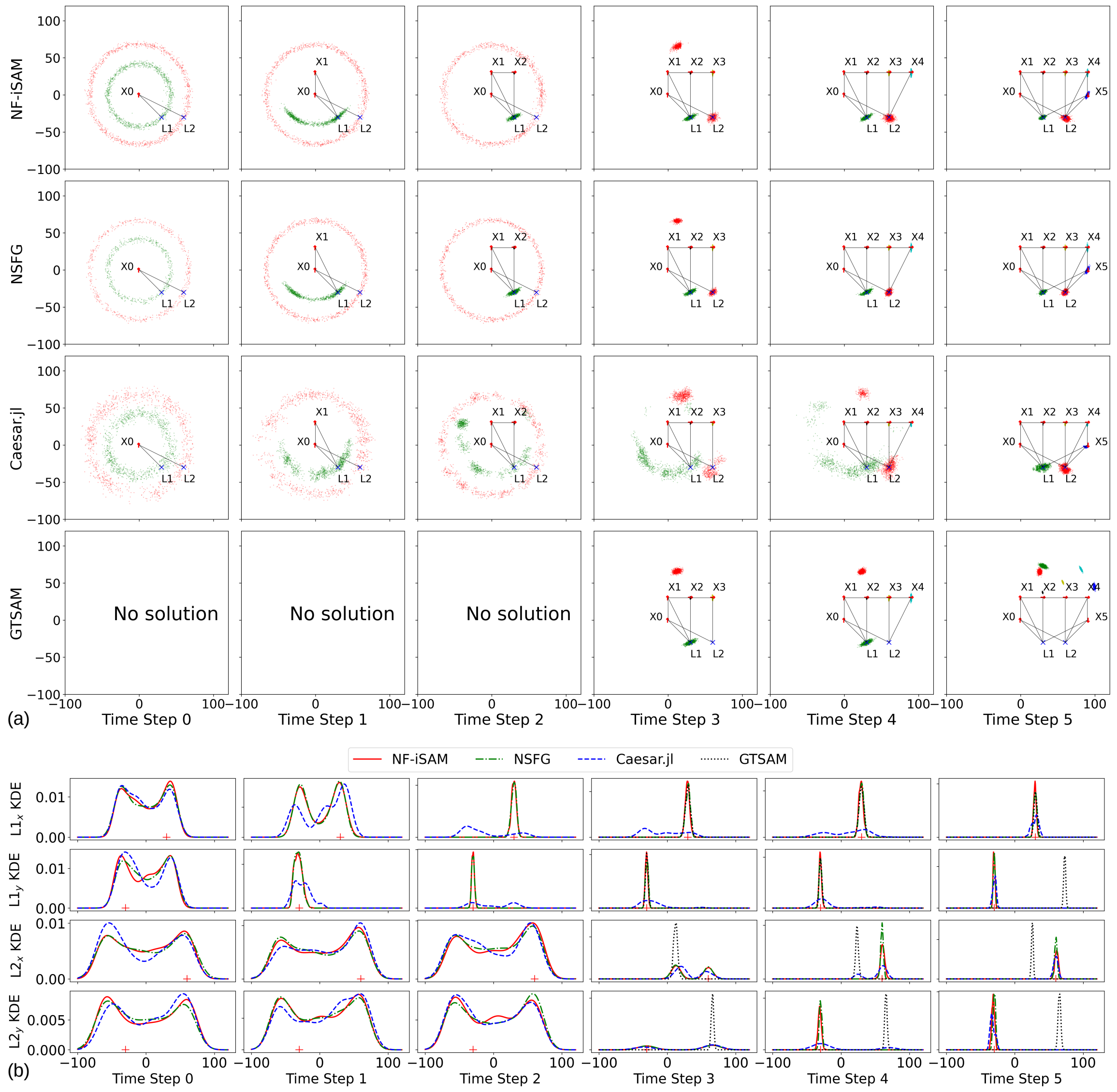
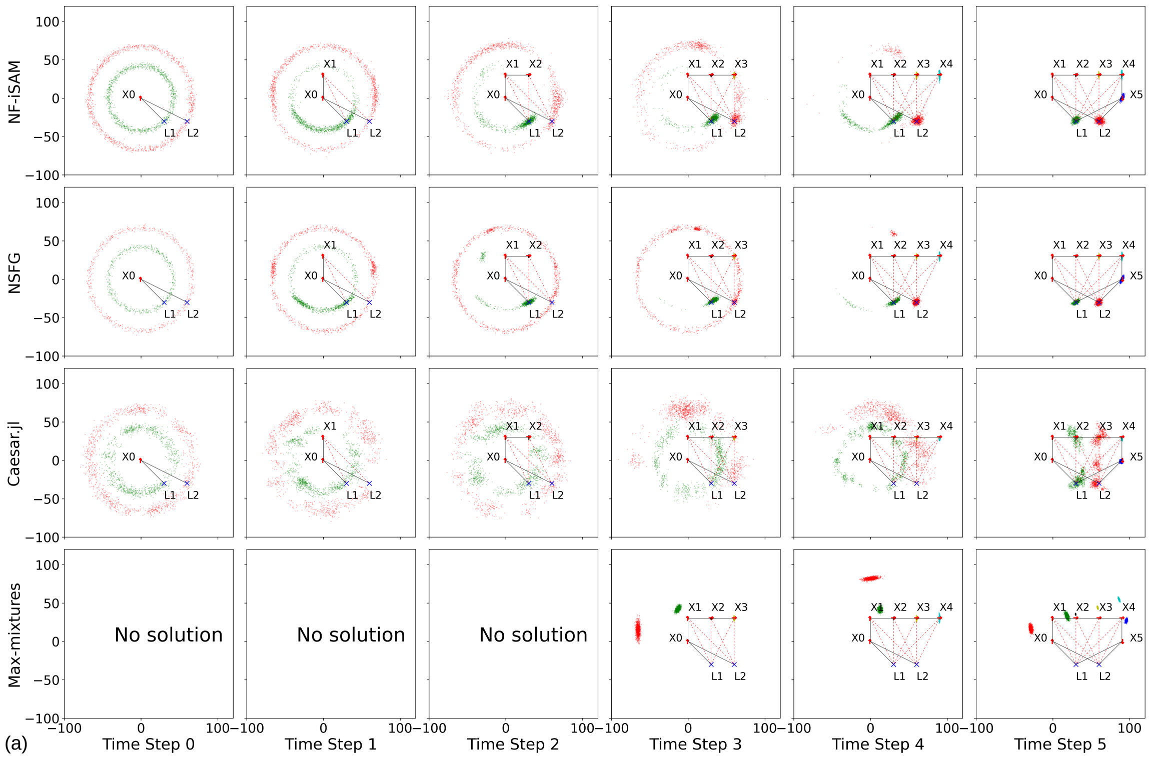


Figure 5-5: Results for the small range-only problem without data association ambiguity: a) samples from joint posteriors and b) kernel density estimation. The robot moves clockwise from $X0$ to $X5$ and measures its distances to the landmarks $L1$ and $L2$. The black lines in (a) mark the odometry and range measurements with certain data association. Groundtruth coordinates of landmark locations are marked by '+' on KDE plots.



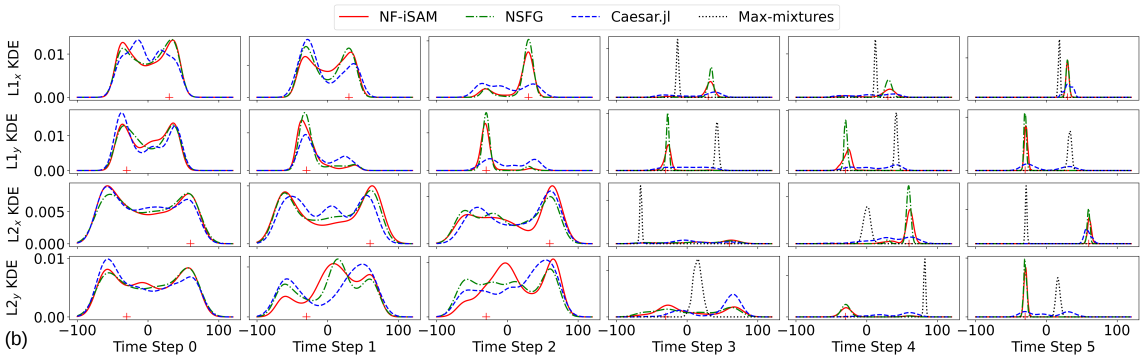


Figure 5-6: Results for the small range-only problem with data association ambiguity: a) samples from joint posteriors and b) kernel density estimation. The robot moves clockwise from $X0$ to $X5$ and measures its distances to the landmarks $L1$ and $L2$. The black lines in (a) mark the odometry and range measurements with certain data association while the red lines in (a) between a robot pose and k landmarks indicate a range measurement that is potentially associated with k landmarks. Ground truth landmarks and poses in (a) are marked by '$\times$' and arrows, respectively. On the KDE plots, groundtruth landmark coordinates are marked by '+' .

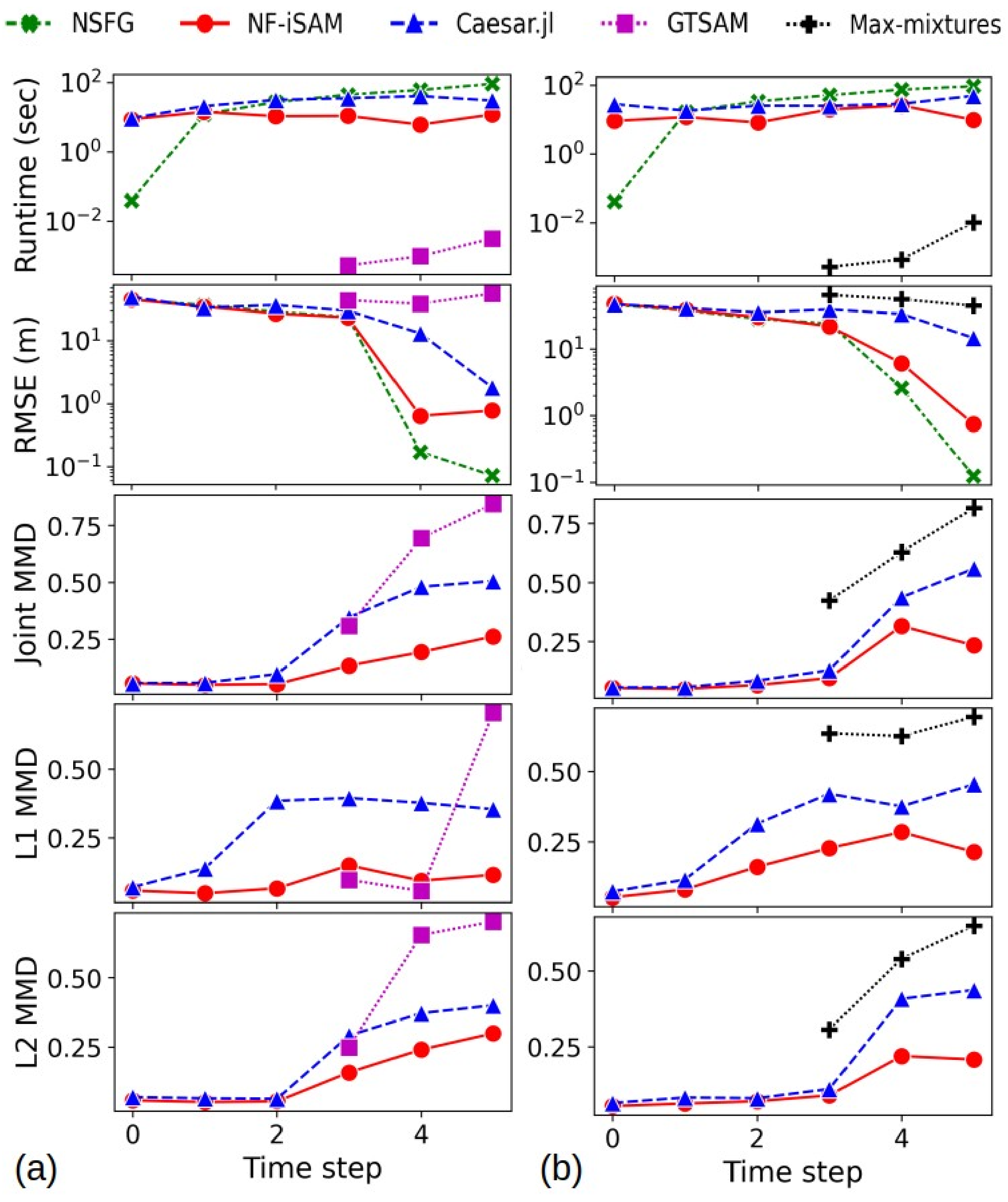


Figure 5-7: Performance of different solvers for the small illustrative range-only problem: (a) data associations are given and (b) data associations are unknown from time step 1 to 4. The performance metrics include computation time, RMSE w.r.t. the ground truth, and maximum mean discrepancy of estimated marginal and joint posteriors to NSFG solutions.

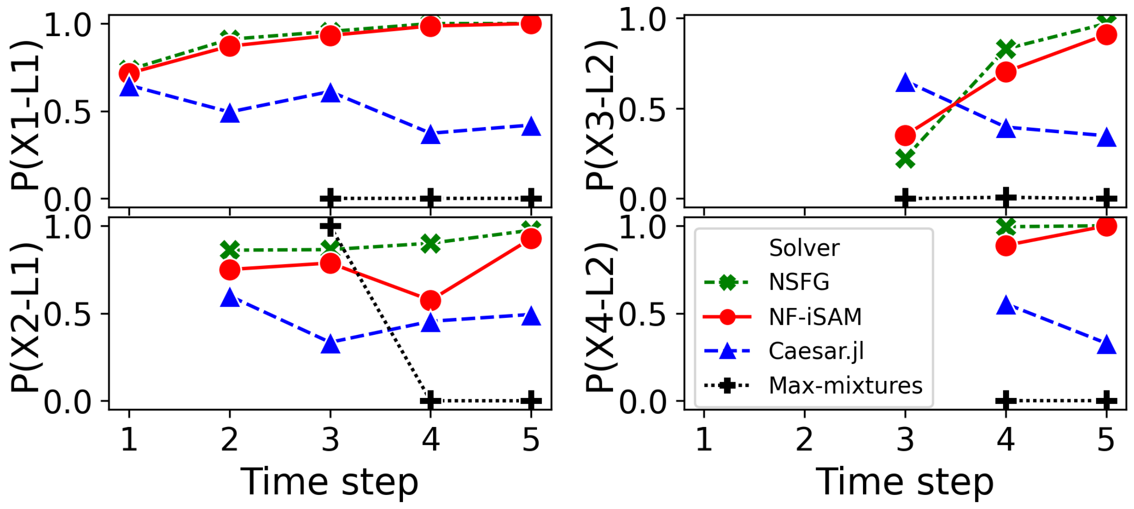


Figure 5-8: Estimated posterior belief of groundtruth data associations for the small problem with data association ambiguity.

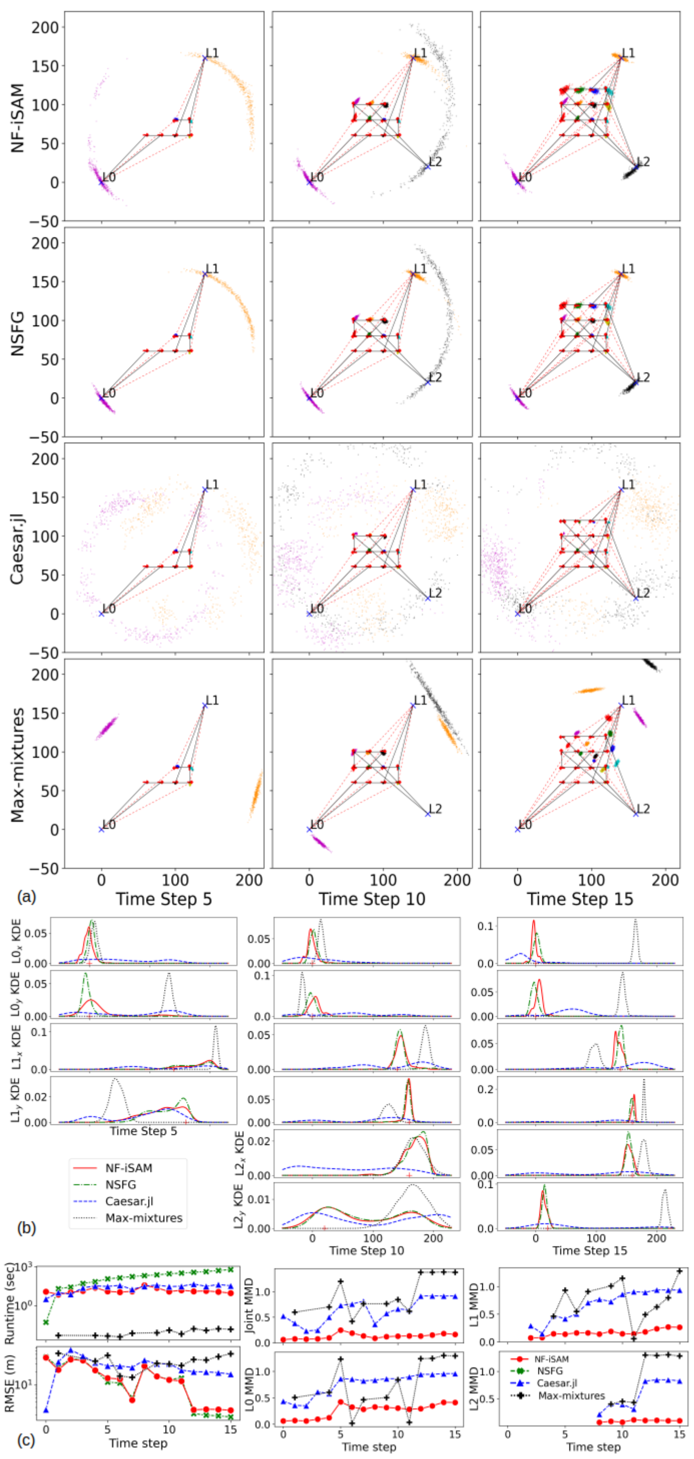


Figure 5-9: Results for the lawnmower path problem on the 4-by-4 grid: a) samples from joint posteriors, b) kernel density estimation, and c) performance. See Fig. 5-6 for our convention about markers in (a).

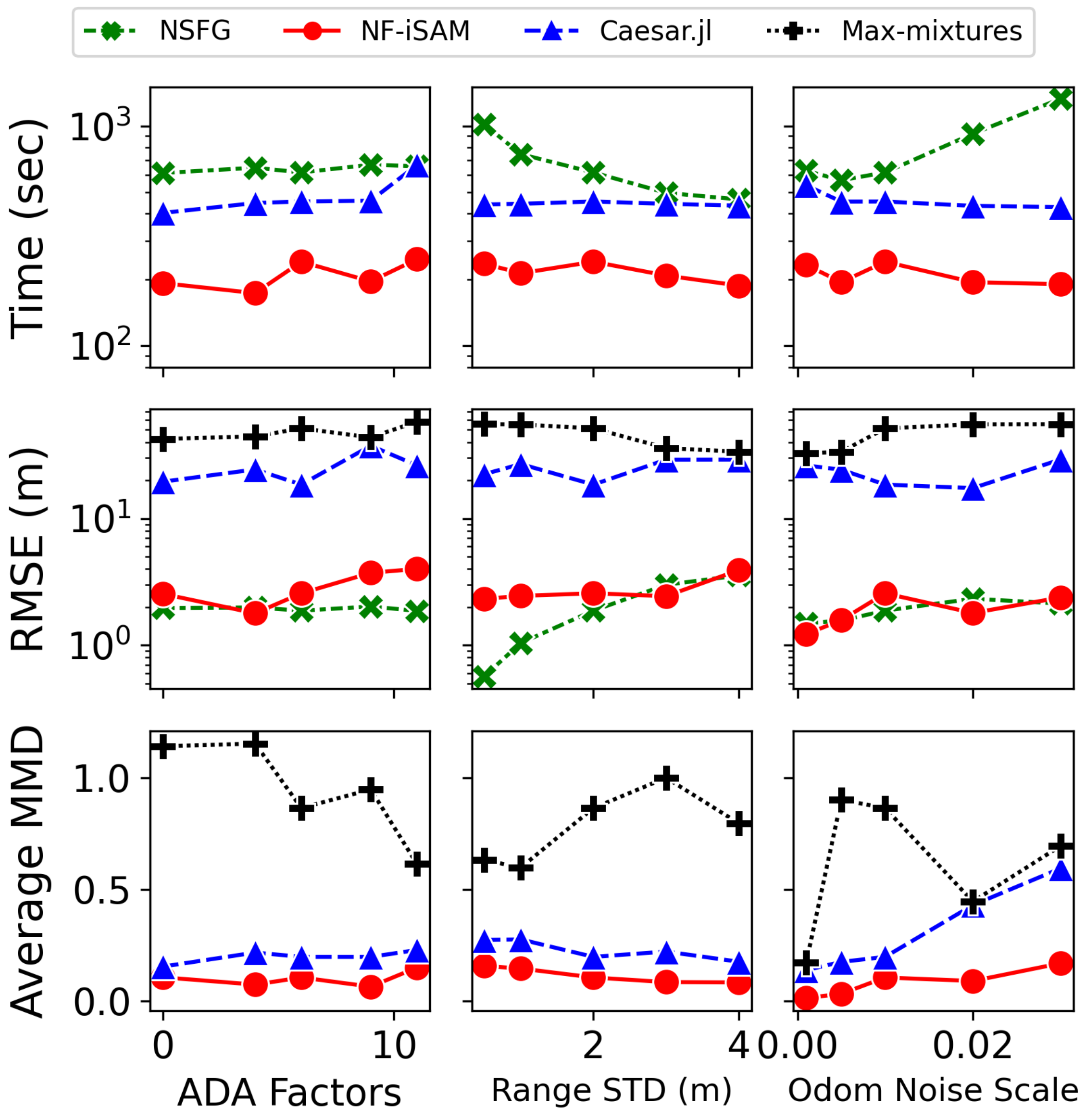


Figure 5-10: Error and accumulated computation time for posterior estimation at the final step of the lawnmower path problem with different measurement noise and numbers of ambiguous data association factors. For each column, only one of the settings varies from the default setting. Average MMD means the average of MMDs for all estimated marginals.

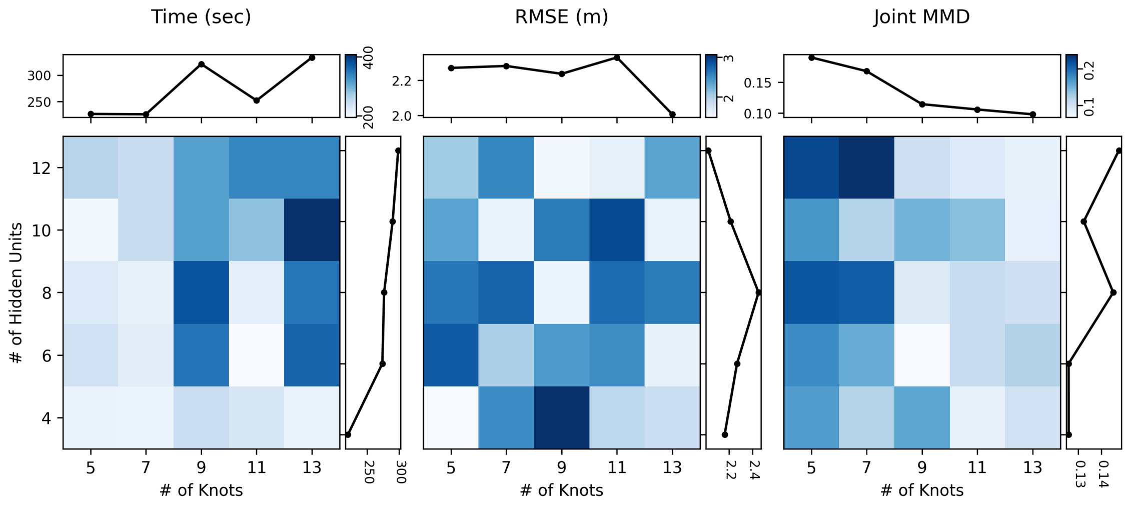


Figure 5-11: Error and accumulated computation time of posterior estimation at the final time step with various hyper-parameters for learning normalizing flows. The parameter study is performed with the medium-scale problem with the default setting regarding noise models and data association ambiguity. Column-wise and row-wise means are shown beside the grids.

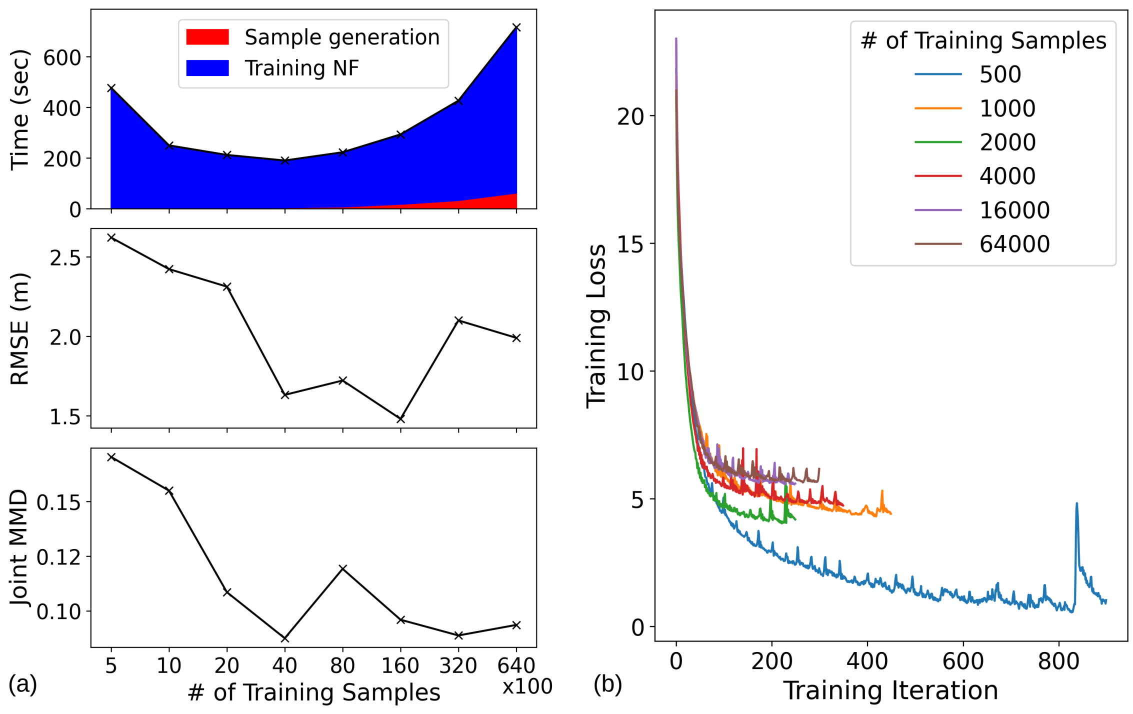


Figure 5-12: Effects of training sample numbers in NF-iSAM results: (a) accumulated time, RMSE, and joint MMD at the final time step of the lawnmower path problem on the 4-by-4 grid, and (b) evolution of loss for training normalizing flows at the final time step.

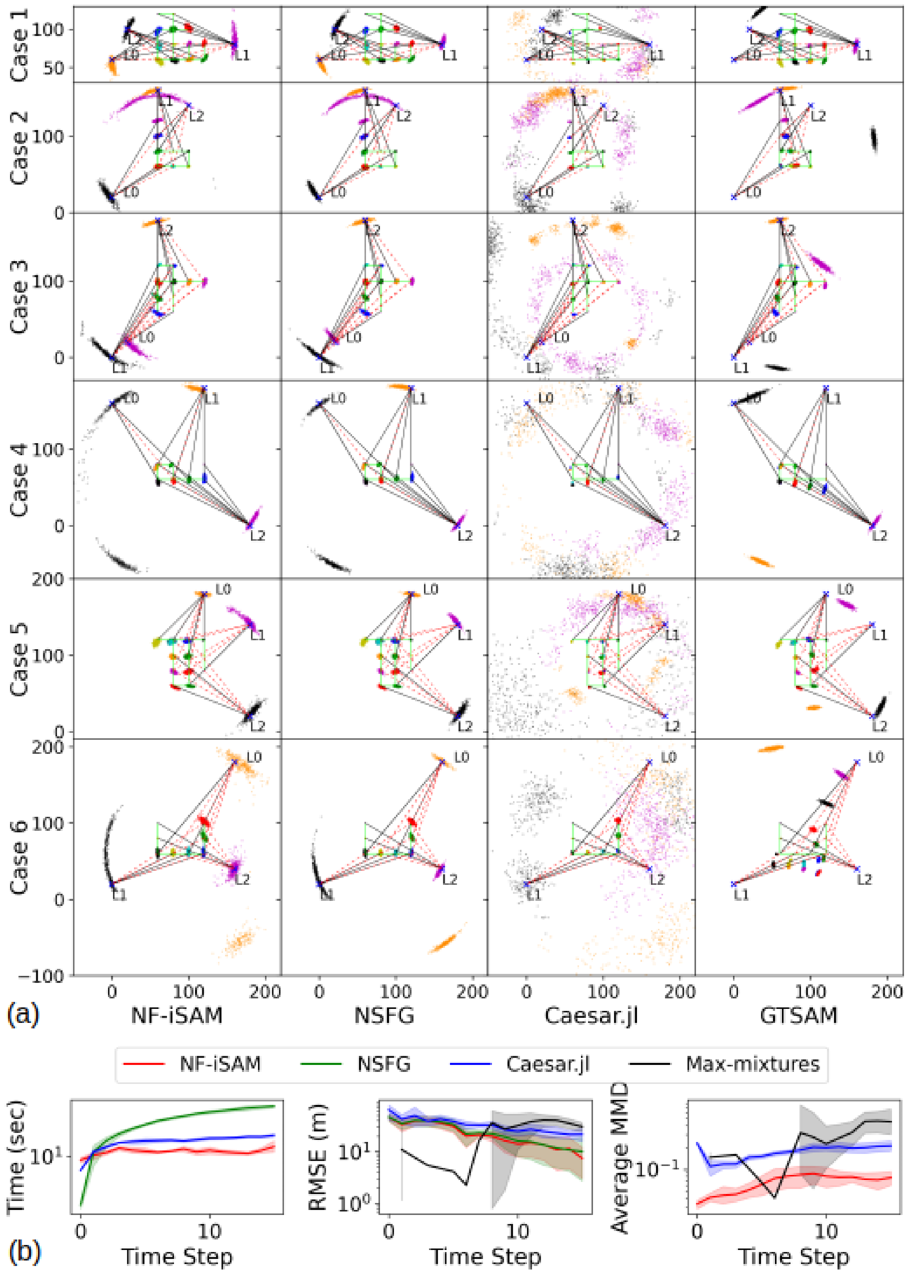


Figure 5-13: Randomly generated cases: (a) samples of estimated posteriors and (b) error bands (95% confidence interval) of performance by different methods for randomly generated cases. Runtime of GTSAM is not shown as it makes the computation time of other solvers less distinguishable in the figure. See Fig. 5-6 for our convention about markers in (a).

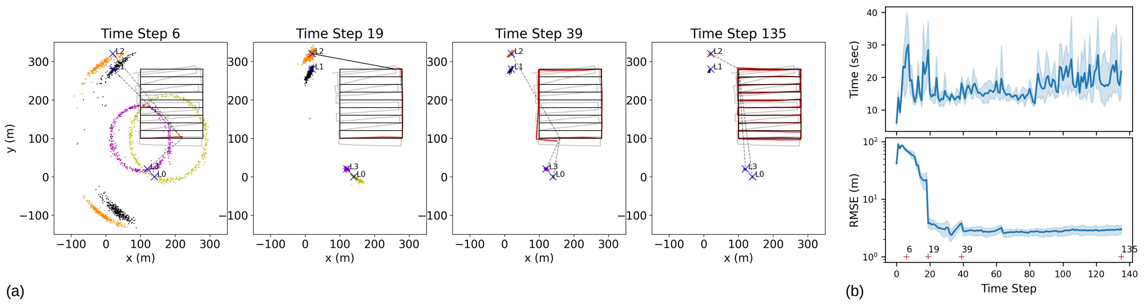


Figure 5-14: NF-iSAM results for the simulated Plaza1 dataset: (a) posterior estimation where robot trajectories are shown in red and estimated by averages of samples, black lines and blue X markers are the ground truth, and gray lines are odometry trajectories and (b) error bands (95% confidence interval) of computation time per incremental update and RMSE of six NF-iSAM solutions to the simulated Plaza1 dataset. NF-iSAM was initialized with different random seeds to get those solutions.

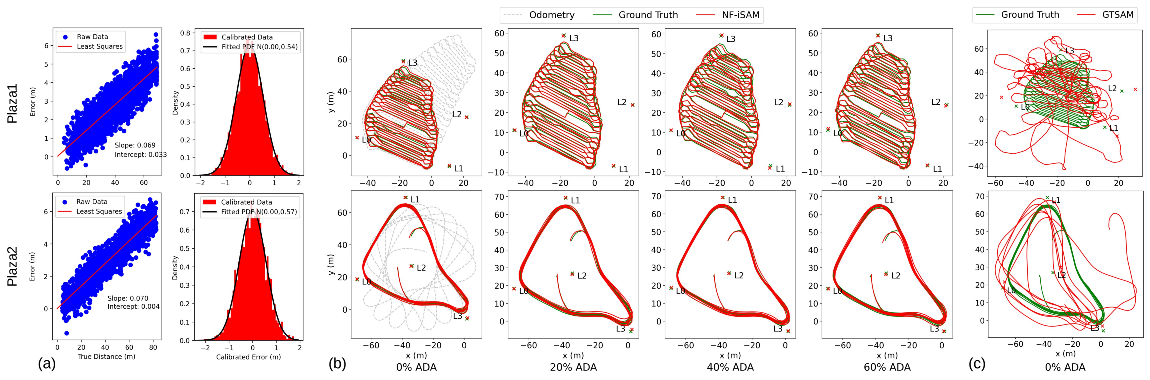


Figure 5-15: Plaza datasets: (a) least squares for modeling distance-dependent bias in range measurements and error distributions (fitted to N(mean, standard deviation)) of the calibrated data which is obtained by subtracting least-squares-predicted bias from the raw data, (b) NF-iSAM's results for the datasets mingled with different fractions of ambiguous data association (ADA) factors, and (c) maximum a posteriori estimation by GTSAM. Trajectories by NF-iSAM are formed by the average of posterior samples. They have been processed by the Kabsch-Umeyama algorithm for trajectory alignment.

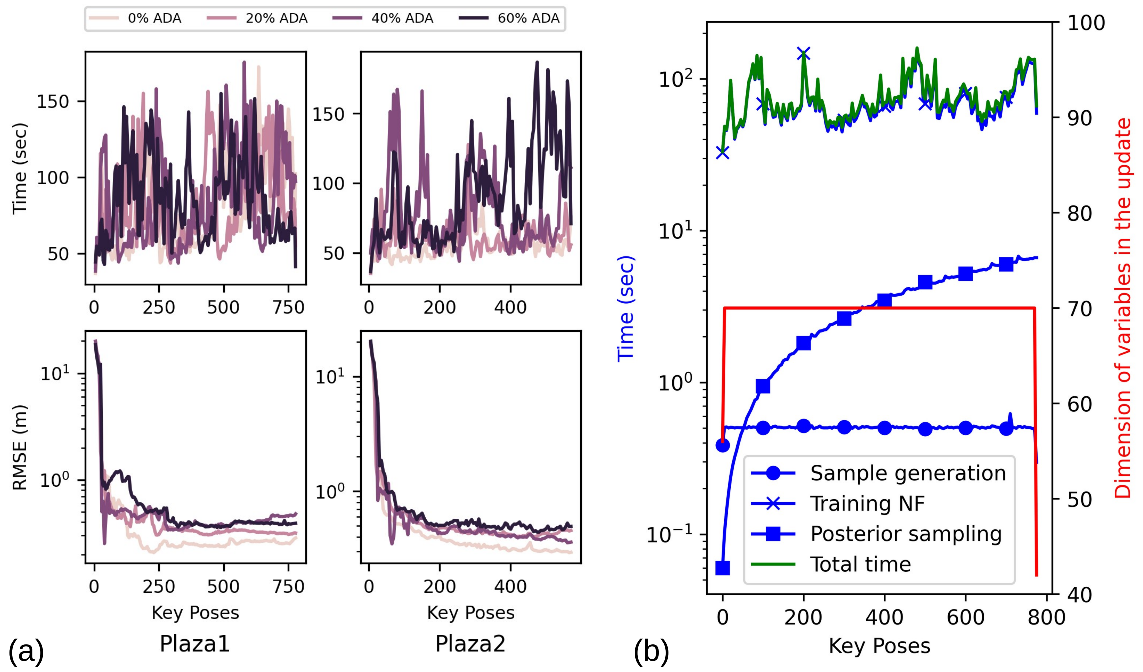


Figure 5-16: Performance evaluation of NF-iSAM for the Plaza datasets: (a) runtime and RMSE with ADA factors and (b) decomposed runtime of NF-iSAM for the Plaza1 sequence without data association ambiguity. We also plot the dimension of variables involved in learning normalizing flows for each incremental upate.

Chapter 6

Full posterior inference for improved visual localization via optimizing fiducial marker placement

The content in this chapter is mainly based on the following paper:

- Qiangqiang Huang, Joseph DeGol, Victor Fragoso, Sudipta N. Sinha, John J. Leonard. **O**ptimizing Fiducial **M**arker **P**lacement for Improved Visual Localization. *IEEE Robotics and Automation Letters (RA-L)* & *IEEE/RSJ International Conference on Intelligent Robots and System (IROS)*, 2023.

6.1 Introduction

Chapters 3-5 focuses on developing full posterior inference algorithms for SLAM. This chapter presents a new application of full posterior inference. Chapter 1.1.3 have introduced that visual localization systems can be fragile in scenes that are weakly textured or present repetitive structures. In visual localization, we have a pre-built map of the environment in hand. *Can we analyze the map to identify difficult areas for visual localization before deploying visual localizaion systems to real-world?* This chapter answers this question by defining camera localizability scores over the entire environment and then applying full posterior inference to compute the camera localizability.

As seen in Fig. 1-6, adding fiducial markers to the environment is a common solution for improved accuracy of visual localization. Thus a follow-up question is *what the optimal positions are for adding markers*. We propose an automatic approach to optimizing marker placement such that 1) the resulting marker positions yield improved accuracy in visual localization and 2) a human user will be able to place markers at positions planned by the approach (e.g., no markers on the ceiling). Specifically, the approach computes optimized marker positions, given a predetermined set of markers and a scene model. The key contributions of this work include:

1. We propose a novel framework that models localizability of camera poses in a scene and computes localizability scores.
2. We develop optimized marker placement (OMP), a greedy algorithm that optimizes marker positions with the goal of increased localizability scores.
3. This is the first work that optimizes marker placement for visual localization based on scene features and fiducial markers.
4. We design a simulation framework for testing marker placement algorithms on 3D scene models that enables others to reproduce and build on our work.
5. We demonstrate that optimized marker placement by our approach can improve the localization rate by up to 20 percent on four different scenes.

6.2 Problem statement

We aim to compute k 3D locations in the scene for placing k fiducial markers such that after marker placement, the camera localization performance improves for query images from anywhere within the scene. In summary, we solve the global search of optimal k locations by a greedy algorithm that seeks one marker placement each time.

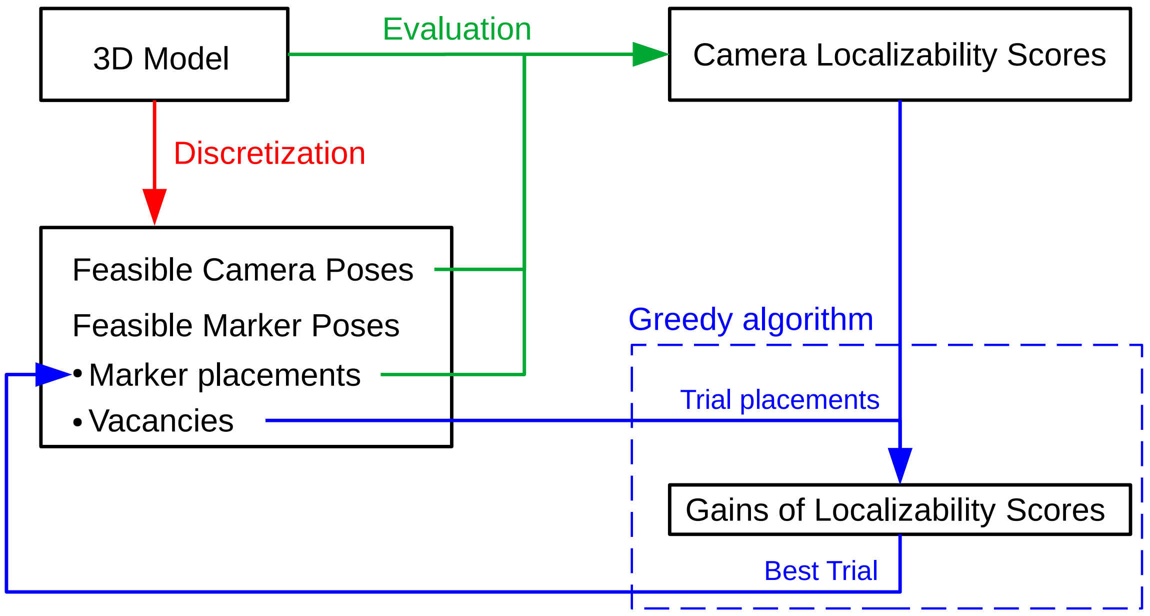


Figure 6-1: Overview of the OMP algorithm. We first create a set of feasible camera poses and marker poses by discretizing space in the 3D model. Then we evaluate localizability scores of the feasible camera poses and update the scores once a feasible marker pose is selected to place a marker. The marker placement is selected by a greedy algorithm as the best trial out of trial placements in the vacancies (unselected marker poses). These trial placements are ranked by gains of localizability scores.

6.3 Approach

6.3.1 Assumptions

This work makes two assumptions: 1) A textured 3D model of the scene is available, and 2) markers and cameras are located on a 3D plane parallel to the ground plane at roughly the eye level of a person with average height. Note that the textured model can be a 3D simulation environment or a dense reconstruction of scenes. We will collect images (e.g., RGB, depth, and surface normal) and corresponding camera poses from the model and take them as input to our approach for optimizing marker placement. The second assumption ensures that our marker placement will be reachable to a human user and constrains the number of feasible camera and marker locations for computational efficiency.

6.3.2 Key elements of the approach

Fig. 6-1 shows an overview of our approach, which is composed of three key elements: 1) discretization, 2) evaluation of camera localizability, and 3) a greedy algorithm for selecting marker placements.

Discretization

We first convert the ground plane in the 3D model to a discretized space of camera and marker poses, as shown in Fig. 6-2. The conversion is implemented by occupancy grid mapping. Centers of unoccupied grid cells are designated as feasible camera locations (dots in Fig. 6-2) while scan points form the perimeter of the free space (lines in Fig. 6-2). We uniformly downsample the scan points to generate a set of feasible marker poses $\mathcal{M}$ (arrows in Fig. 6-2) whose orientations are determined by surface normals in the 3D model. Note that one can choose other ways to select feasible marker poses and then still apply our marker placement algorithm. For example, the feasible marker poses can be further refined by incorporating semantics and physical constraints. It is possible that the algorithm could produce a marker placement in an infeasible location, although we found this was rare. Even so, we have done a sensitivity study showing that we can place the marker nearby the exact location and still get most of the gain.[1] We derive a set of feasible camera poses $\mathcal{C}$ from the feasible camera locations. Each of the camera locations yields n camera poses whose optical axes are parallel to the ground plane and evenly spaced in $[0, 2\pi]$ (e.g., the default $n = 8$).

Camera localizability score

We compute camera localizability scores by evaluating uncertainty in localizing feasible camera poses. Specifically, for any feasible camera pose $\mathbf{c} \in \mathcal{C}$ (the corresponding random variable is C), we synthesize measurements z to create a camera localization problem, estimate the distribution of the camera pose $p(C|z)$, and define the localiz-

[1]A sensitivity study about the influence of position and size deviations of markers on localization performance is available in Sec.6.6.

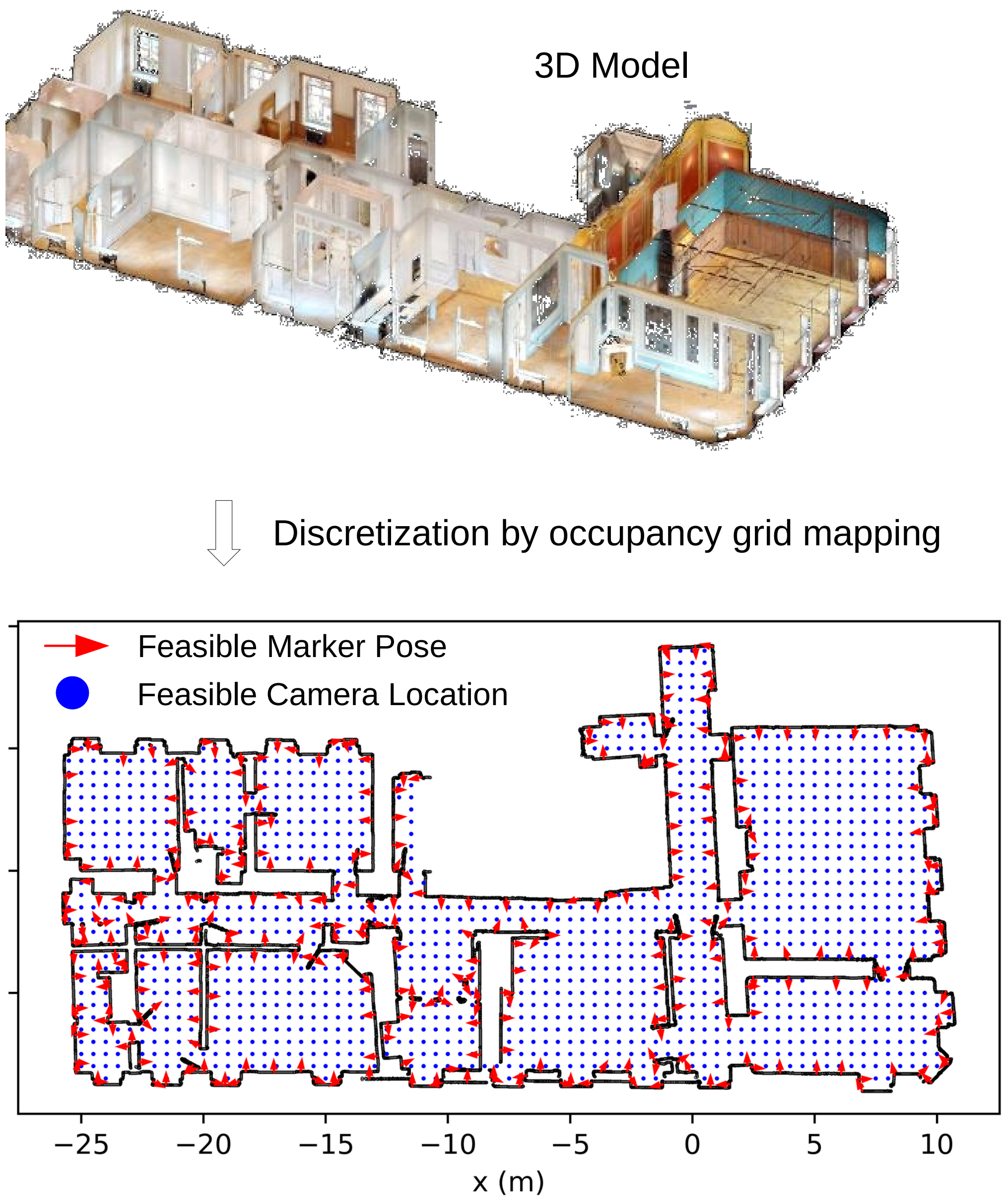


Figure 6-2: Discretization of a model from the Habitat-Matterport 3D dataset [107]. We select a ground plane in the 3D model at roughly eye level of a human user. The discretized space of the ground plane consists of feasible marker poses (red arrows), which are sampled from scan points on the ground plane perimeter, and feasible camera locations (blue dots), which are centers of unoccupied cells in the 2D discrete grid.

ability score of the camera pose $l(\mathbf{c})$ as the negation of the entropy of the distribution, as shown in

$$l(\mathbf{c}) = -H(p(C|z)) = \mathbb{E}[\ln p(C|z)]. \tag{6.1}$$

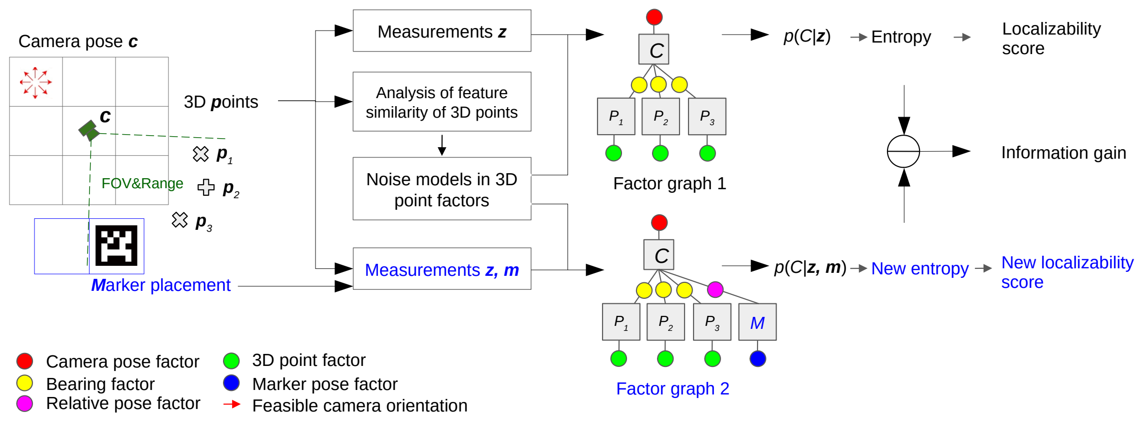


Figure 6-3: Evaluation of localizability scores and the information gain brought by a marker placement. On the left we show a grid of feasible camera poses. Feasible camera poses are positioned at cell centers with orientations shown as the red arrows. The field of view of camera pose **c** covers points p_1, p_2, and p_3 in the 3D model and a marker placement on the discretized perimeter of the level set of the ground plane. We synthesize measurements z of the points to create a camera localization problem using scene features. The problem is represented by factor graph 1 and distribution $p(C|z)$ by which we can compute the entropy as well as the localizability score of the camera pose seeing no markers. We penalize contributions of repetitive structures on the localizability score via the analysis of feature similarity. With additional measurements **m** to the marker, we create another localization problem which is represented by factor graph 2 and distribution $p(C|z, \mathbf{m})$. The new problem leads to a new entropy and a new localizability score.

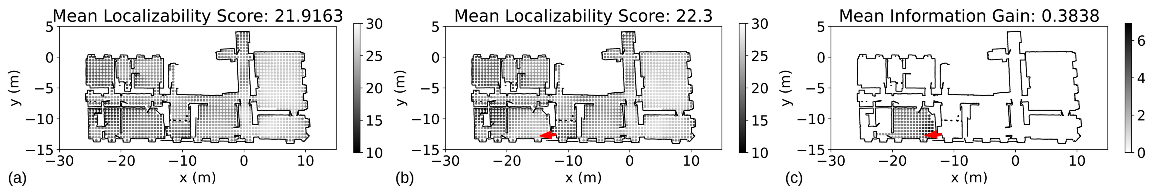


Figure 6-4: Results of localizability scores: (a) no markers, (b) a trial marker placement (red arrow), and (c) the information gain. The score (or gain) at a dot is the mean score (or gain) of camera poses at the dot with all feasible orientations. Darker dots stress low localizability scores in (a) and (b) and high information gains in (c). This trial turns out to be the first marker in the optimized placement (see the Apartment in Fig. 6-9).

If a new fiducial marker is added in the field of view (FOV) and range of the camera pose, the new synthetic measurement regarding the marker will change the entropy of the camera pose distribution, resulting in an information gain that quantifies the impact of the marker placement. Fig. 6-3 summarizes steps for evaluating the localizability score and the information gain. These steps are explained in detail in following paragraphs.

Synthesized data for computing the localizability score: The leftmost part of Fig. 6-3 illustrates 3D points and a feasible marker pose (i.e., trial marker placement) that are in the FOV/range of a feasible camera pose[2]. We collect RGB and depth images at the camera pose in the 3D model. These images will be used to compute 3D points and descriptors of features (e.g., SIFT [84]). We use these known poses and points to synthesize measurements and estimate probability density functions (PDFs) of the camera pose variable. Measurements $\mathbf{z}$ in Fig. 6-3 contain the camera pose, the 3D points, and bearings between them. Thus the PDF $p(C|\mathbf{z})$, which is represented by factor graph 1, expresses the distribution of the camera pose constrained by the 3D points. Placing a marker in the FOV/range of the camera leads to new synthetic measurements $\mathbf{m}$ of the marker pose and the relative pose between the marker and the camera. As a result, the camera pose is further constrained by measurements $\mathbf{m}$ thus is described by a new PDF $p(C|\mathbf{z}, \mathbf{m})$ represented by factor graph 2 in Fig. 6-3. We use an approach that is similar to the one proposed by Stachniss et al. [124] to define the information gain of a marker placement. The information gain is defined as the change of entropy that the marker placement $\mathbf{m}$ yields at the camera pose $\mathbf{c}$, as seen in

$$I(\mathbf{m}, \mathbf{c}) = H(p(C|z)) - H(p(C|z, \mathbf{m})). \tag{6.2}$$

Fig. 6-4a shows localizability scores of camera poses in the original ground plane with no marker placement. Note that the score at a dot in the figure is the mean score of camera poses with all feasible orientations. Fig. 6-4b shows localizability scores after adding a marker (the arrow) to the ground plane perimeter. The scores increase in the region around the marker, indicated by the brighter dots in the region in Fig. 6-4b and the information gain in Fig. 6-4c.

[2] In practice, one can further refine marker poses in the FOV by considering marker sizes and rejecting corner cases that may fail the detection of markers. The cases include marker poses that are too close to the boundary of the view frustum of the camera.

Algorithm 10: Optimized Marker Placement (OMP)

```
Input: The number of markers $k$, the list of feasible marker poses $\mathcal{M}$, the
       ground plane space $\mathcal{S}$
Output: $k$ marker poses
1  Initialize an empty list for storing selected marker poses $\mathcal{O}$
2  repeat $k$ times
3      Initialize the best marker pose $T^\star = \emptyset$
4      Initialize the highest localizability gain $g^\star = -\inf$
5      Evaluate localizability scores $\mathcal{L}^\star$ of camera poses in space $\mathcal{S}$
6      for Pose $T$ in $\mathcal{M}$ do
7          Place a marker at pose $T$ in space $\mathcal{S}$
8          Evaluate localizability scores $\mathcal{L}$ of camera poses
9          Compute information gains $\mathcal{I} = \mathcal{L} - \mathcal{L}^\star$
10         Evaluate localizability gain $g$ of the marker by (6.6)
11         if $g > g^\star$ then
12             $T^\star = T$
13             $g^\star = g$
14         Remove the marker from space $\mathcal{S}$
15     Push $T^\star$ to $\mathcal{O}$
16     Place a marker at pose $T^\star$ in space $\mathcal{S}$
17     Remove $T^\star$ from $\mathcal{M}$
18 return List of marker poses $\mathcal{O}$
```

Analysis of feature similarity of 3D points: Repetitive structures in scenes cause similar features across RGB images and can result in localizing to a wrong location. To reduce the contribution of repetitive structures to localizability scores, we penalize the localizability score if similar features appear in the FOV of the camera. Specifically, when modeling 3D points with similar features in factor graphs, we set greater uncertainty in noise models of 3D point factors to encode the fact that similar 3D points are ambiguous and less informative. (6.3) shows the 3D point factor that formulates the difference between the noisy 3D location $\tilde{\mathbf{p}}$ and true 3D location $\mathbf{p}$ using a Gaussian distribution

$$p(\tilde{\mathbf{p}}|\mathbf{p}) = \mathcal{N}(\tilde{\mathbf{p}} - \mathbf{p}; \mathbf{0}, \Sigma_{\mathbf{p}}) \tag{6.3}$$

where $\Sigma_{\mathbf{p}}$ is the covariance we set for modeling noise. For example, in the leftmost

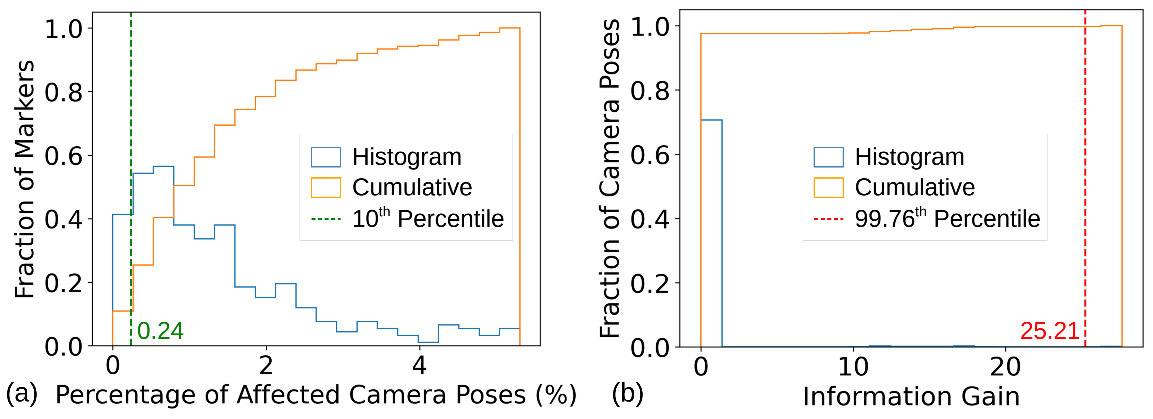


Figure 6-5: Histograms for the HM3D apartment model: (a) percentage of affected camera poses and (b) information gains at camera poses yielded by a marker. The most visible 90% markers (i.e., $v = 90$) means 10^{th} percentile in (a), determining the percentile $q = 99.76$ by (6.9). The 99.76^{th} percentile in (b) indicates a localizability gain 25.21 of the marker by (6.6).

part of Fig. 6-3, points $\mathbf{p}_1$ and $\mathbf{p}_3$ are visually similar, so we set big covariances in 3D point factors of $\mathbf{p}_1$ and $\mathbf{p}_3$. Informally, factors with big covariances impose loose constraints on the camera pose distribution, leading to lower contributions on the localizability score.

We perform an analysis of feature similarity of 3D points to determine noise models in 3D point factors (i.e., $\Sigma_{\mathbf{p}}$ in (6.3)), as shown in the flow chart in Fig. 6-3. The analysis is to count the number of similar 3D points to any 3D point. The resulting covariance $\Sigma_{\mathbf{p}}$ is formulated as

$$\Sigma_{\mathbf{p}} = (1 + n_{\mathbf{p}})\Sigma_0 \tag{6.4}$$

where Σ_0 is a base covariance (e.g., $diag(2.5, 2.5, 2.5) \times 10^{-3}\ m^2$ in our experiments) and $n_{\mathbf{p}}$ denotes the number of similar 3D points to the query point $\mathbf{p}$. 3D points observed by all feasible camera poses are filtered to select similar ones of the query point. The selection is determined by two criteria: 1) the selected points have similar descriptors to the query point and 2) the selected points are not too close to the 3D location of the query point. The intuition is that, if two areas in the scene look similar but they are far away from each other, a wrong place recognition would incur a huge localization error.

Estimation of camera pose distributions: We use the Laplace approximation [9, Ch. 4.4] to estimate a Gaussian distribution that approximates the camera pose distribution encountered in the synthetic localization problem. The mean of the Gaussian is the known feasible camera pose so the covariance Σ is the only unknown. The covariance can be approximated by an estimated Hessian of the negative logarithm of the camera pose distribution at the mean (see [70, Sec. 2] for the estimation of the covariance). Thus the entropy encountered in the synthetic localization problem can be approximated by

$$H(p(C|\cdot)) \approx \frac{1}{2}\ln|\Sigma| + \frac{d}{2}(1+\ln(2\pi)) \tag{6.5}$$

where the dimensionality d is 6 for 6DOF poses.

The greedy algorithm

The algorithm sequentially selects k poses from feasible marker poses $\mathcal{M}$ (see Algorithm 10). The algorithm executes k loops to search the best k poses. In each loop, we update localizability scores, tentatively place a marker at any feasible marker pose, and compute localizability gains of trial marker placements. The best pose that earns the highest localizability gain will be removed from feasible marker poses and be permanently occupied by a marker. The marker will influence future updates of localizability scores.

The crux of computation in Algorithm 1 is evaluating localizability scores. To avoid redundant computation, we update the localizability score of a camera pose only if the newly added marker is covisible to the camera pose. The complexity of Algorithm 1 is $O(|\mathcal{C}|+k|\mathcal{M}|\max_{\mathbf{m}}(|\mathcal{C}_{\mathbf{m}}|))$ where $O(1)$ is the complexity for evaluating the localizability score of a camera pose. $\max_{\mathbf{m}}(|\mathcal{C}_{\mathbf{m}}|)$ denotes the maximal number of covisible camera poses to a marker so it generally increases with the FOV and range of the camera. The first term $|\mathcal{C}|$ indicates the cost for initializing localizability scores over all feasible camera poses while $\max_{\mathbf{m}}(|\mathcal{C}_{\mathbf{m}}|)$ in the second term bounds the cost for evaluating the localizability gain brought by a marker. For each of the k loops, we evaluate localizability gains of all $|\mathcal{M}|$ markers. Note that the time complexity can be further reduced because it is not necessary to re-evaluate localizability gains for all markers in each loop (lines 8-10 in Algorithm 1). For example, if the covisible camera poses of an unselected marker have not been affected by all selected markers, we do not need to re-compute the localizability gain of that unselected marker.

We discuss the possibility of generalizing the greedy algorithm by re-defining the localizability score. For example, one can use the pose estimation error from a visual localization system (e.g., Fig. 8) to replace the localization score and keep the rest of the algorithm the same. The new marker placement based on the error may enjoy advantages in localization experiments using the same localization system since the marker placement is directly optimized for the system. However, updating the error along with trial marker placements is computationally much more expensive than evaluating the localization score since we need to add markers to the 3D scene model, generate new map and test images, update the map in the localization system, estimate camera poses of test images using the system, and compute the pose estimation error. In contrast, updating the localizability score just needs to re-estimate camera pose distributions, as shown in Fig. 6-3.

We summarize information gains at all feasible camera poses in the scene, using a single scalar quantity that we refer to as localizability gain. Informally, one could think of the localizability gain as the reward for placing an additional marker at a specific position. The localizability gain of any marker placement $\mathbf{m}$ is defined as the q^{th} percentile of information gains that marker $\mathbf{m}$ yields at all feasible camera poses $\mathcal{C}$, as seen in

$$g(\mathbf{m}) = \inf\{i \in \mathbb{R} : F_I(i) \geq \frac{q}{100}\}, \tag{6.6}$$

where $F_I(\cdot)$ is the cumulative distribution function (CDF) after sorting the information gains at all camera poses

$$\mathcal{I} = \{I(\mathbf{m}, \mathbf{c}) : \mathbf{c} \in \mathcal{C}\}. \tag{6.7}$$

The choice of percentile $q \in [0, 100]$ is crucial and dependent on environments (i.e., the ground plane). For example, in a large environment where any marker is only visible to a small fraction of feasible camera poses, a low percentile q would likely incur zero localizability gains for all markers since camera poses seeing no markers receive zero information gains and constitute a great portion of the information gain distribution $\mathcal{I}$.

We use an adaptive approach to determine the percentile q before computing the localizability gain. The approach introduces a hyperparameter $v \in [0, 100]$ and ensures that the most visible v percent of markers earn nonzero localizability gains. A high v allows more markers, even the ones stuck in corners, to effectively join in the selection of best marker while a low v favors the most visible ones among feasible marker poses. In the ground plane space, for any marker $\mathbf{m}$, we can find a set of affected camera poses $\mathcal{C}_{\mathbf{m}}$ that are supposed to see the marker (i.e., nonzero info. gain). We can derive a CDF $F_P(p)$ using percentages of affected camera poses for all markers

$$\mathcal{P} = \left\{ \frac{|\mathcal{C}_{\mathbf{m}}|}{|\mathcal{C}|} \times 100 : \mathbf{m} \in \mathcal{M} \right\}. \tag{6.8}$$

To ensure only the most visible v percent of markers earn nonzero localizability gains, the percentile q is determined by the $(100 - v)^{th}$ percentile in percentages of affected camera poses, as seen in

$$q = 100 - \inf \left\{ p \in [0, 100] : F_P(p) \geq \frac{100 - v}{100} \right\}. \tag{6.9}$$

(6.9) indicates q is a non-decreasing function of v. When v approaches 100, q approaches 100 as well so only markers that earn a greater maximum in information gains will be considered in the best marker selection (see (6.6)); when v approaches 0, q approaches 0 as well so the best marker will only be selected from markers that influence large areas. Thus the choice of hyperparameter v can reflect the trade-off between helping the worst single camera pose and influencing the most camera poses.

Fig. 6-5 shows an example for computing the percentile q and the localizability gain for the marker placement in Fig. 6-4. We set $v = 90$ as the default setting so the

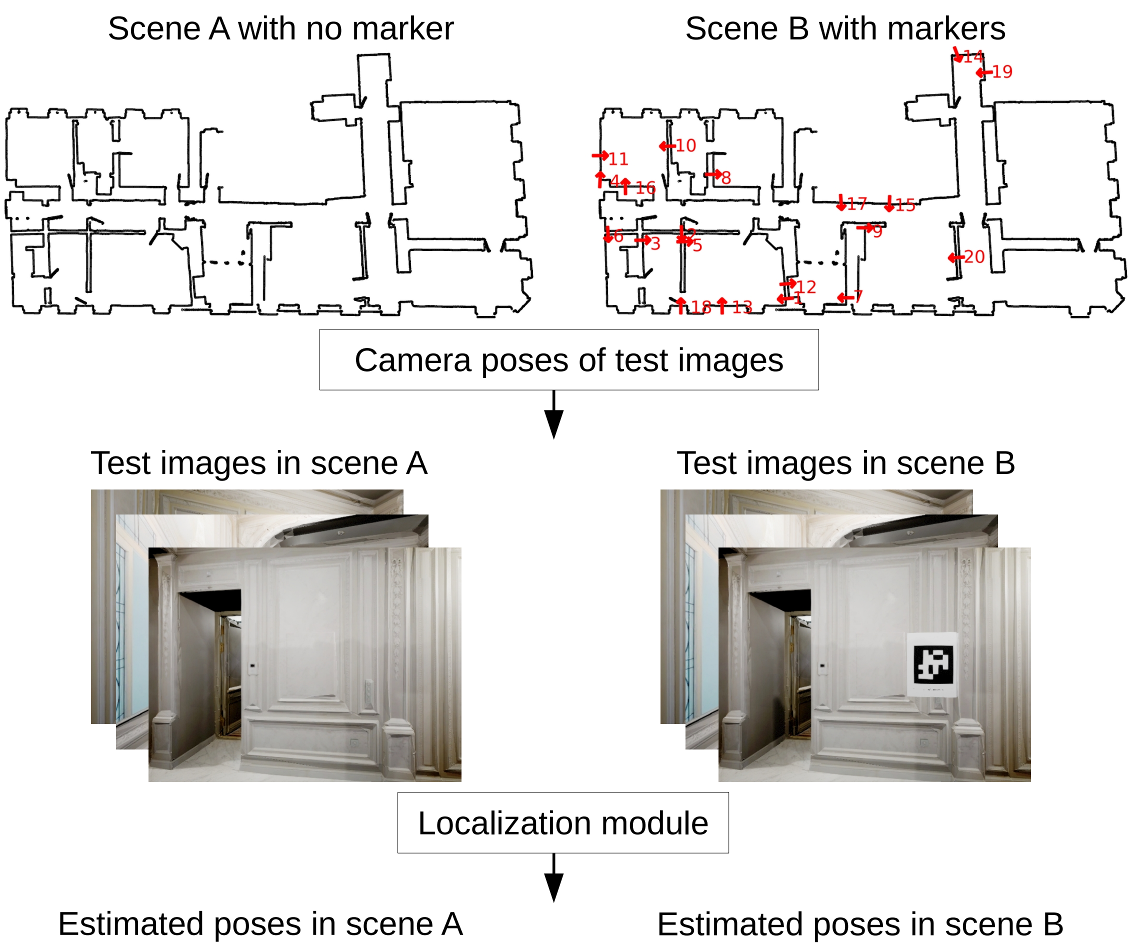


Figure 6-6: The flowchart of our system for performing camera localization experiments. Scenes with different marker placements share the same set of camera poses for acquiring test images and the same localization module.

most visible 90% markers receive nonzero localizability gains and are effective best marker candidates. This setting results in a marker placement strategy that tends to support worst camera poses instead of area coverage, as shown in the optimized marker placement for the apartment model in Fig. 6-9. No markers are placed in the two big rooms on the right of the apartment since (i) camera poses in these rooms already enjoyed good localizability scores (see Fig. 6-4a) and (ii) a large hyperparameter v does not emphasize area coverage.

6.4 Experimental setup

6.4.1 Implementation

We implemented all three key elements and Algorithm 10 in Sec. 6.3.2 in Python with assistance of a few open source software packages. We used the Unreal Engine 4.27 [39] and the AirSim library (v1.8.1) [118] to simulate and collect images from 3D models. We used the Open3D library [141] to downsample scan points to get candidate marker locations. We used the GTSAM library [29] to create factor graphs and estimate covariances in Gaussian approximations of camera pose distributions. The SIFT feature [84] was used throughout our experiments.

Additionally, we implemented a simulation system for testing marker placement algorithms and a camera localization module for estimating camera poses of test images. Fig. 6-6 presents a flowchart of the system. The system adds markers to a scene model at positions planned by marker placement algorithms and then generates test images from the same set of camera poses for different marker placements for the fairness in comparison. We stress three advantages of the simulation system over real world pipelines for performing camera localization experiments: 1) reproducible data collection by other researchers for future development of marker placement algorithms, 2) a large number of test images that cover the scene, 3) consistent camera poses for generating test images in scenes with different marker placements.

6.4.2 Evaluation

Methods for comparison

We compare our algorithm OMP with 1) no marker placement, 2) random marker placements, 3) uniform marker placements, and 4) markers placed by a human. Random marker placements refer to uniformly weighted samples from feasible marker poses. Uniform placements distribute the markers roughly uniformly along the perimeter of the environment (see [89] for details). We generated 5 versions of random and uniform placements for each scene and all placements were manually inspected in

Table 6.1: Specifics of scenes

Model	Area (m^2)	# of map images	# of test images
Apartment	339.3	10856	10000
Studio	149.6	2832	3000
Office	108.3	1768	2000
Room	21.0	250	200

scene models to ensure reasonable quality. The comparison with humans is only conducted in the real experiment. The human prioritizes centers in less textured areas.

Scenes

The method comparison is performed on four scenes: apartment, studio, office, and room, as seen in Fig. 6-9. The first two are pre-built dense maps of realworld spaces, provided by the Habitat-Matterport 3D (HM3D) Research Dataset [107], while the third model is an Unreal Engine simulation environment that resembles typical realworld offices[3]. The first three are for simulated experiments. The last one is a motion capture room at MIT for the real experiment (see Fig. 6-8). The textured mesh of the room was created by fusing RGB-D images from groundtruth poses, using the volumetric fusion [23] and marching-cubes algorithms and the screened Poisson surface reconstruction [73]. Table 6.1 lists specifics of these models.

The localization module

Fig. 6-7 presents the flowchart of our localization module. The localization module is similar to standard approaches [115] but with an extra function of fiducial marker detection, provided by the AprilTag library [99]. The tag detection and VLAD descriptors [67] were sequentially employed to find matched images in the map data. Camera poses were estimated using P3P [48] with RANSAC [41] followed by Levenberg-Marquardt optimization [15]. The rotation error δ_R is defined as the

[3]The serial number of the apartment model is 00770-NBg5UqG3di3 in the HM3D dataset and that of the studio model is 00254-YMNvYDhK8mB. We inspected all scenes in the dataset and chose these two as representatives of medium and large scenes with textureless areas and potential perceptual aliasing. The office model is the ThreeDee Office project in the Unreal Engine Marketplace.

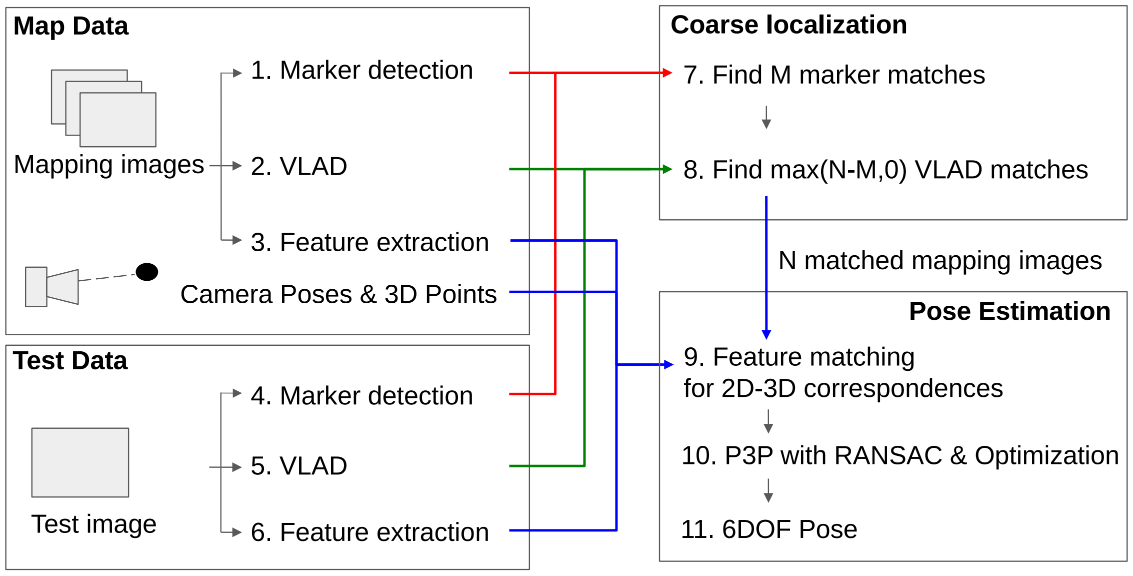


Figure 6-7: The localization module using fiducial marker detection. The numbers indicate the order of different operations.

angular distance between the estimated rotation matrix $\hat{R}$ and the groundtruth rotation R while the translation error δ_t is defined as the Euclidean distance between the estimated translation $\hat{t}$ and the groundtruth translation t, as seen in

$$\delta_R = \left|\arccos\left(\frac{\mathrm{tr}(\hat{R}^\intercal R) - 1}{2}\right)\right|, \tag{6.10}$$

$$\delta_t = \left\|\hat{t} - t\right\|_2. \tag{6.11}$$

The map and test data

In simulated experiments, the camera was set to a FOV of 90 degrees and a range of 10 meters (RGB res. 600×450, depth res. 300×225). The camera poses for collecting the map data are the same as the feasible camera poses in the ground plane space. The camera poses for collecting test images are sampled from the feasible camera poses with weights and then perturbed by translation and rotation noises that are subject to a uniform distribution in $[-0.5, 0.5]$. We intend to sample more densely from the difficult areas, which are of our interest, so the weights in the sampling correlate with localizability scores for generating more test images around low-scoring camera poses[4]. Let $\mathcal{L} = \{l(\mathbf{c}) : \mathbf{c} \in \mathcal{C}\}$ be the set of localizability scores of feasible camera poses in the ground plane space with no markers. The weights are defined as

$$\mathcal{W} = \{2l^\star - \bar{l} - l(\mathbf{c}) : \mathbf{c} \in \mathcal{C}\}, \tag{6.12}$$

where $l^\star$ is the maximal score in $\mathcal{L}$ and $\bar{l}$ is the mean of all scores. Thus all weights will be non-negative and a lower score incurs a greater weight. In the real experiment, we used the Realsense L515 camera for RGB-D data (image res. 1280×720) and the OptiTrack system for groundtruth poses. The map and test data were sampled along two lawn-mower paths around feasible camera poses, as seen in Fig. 6-8:

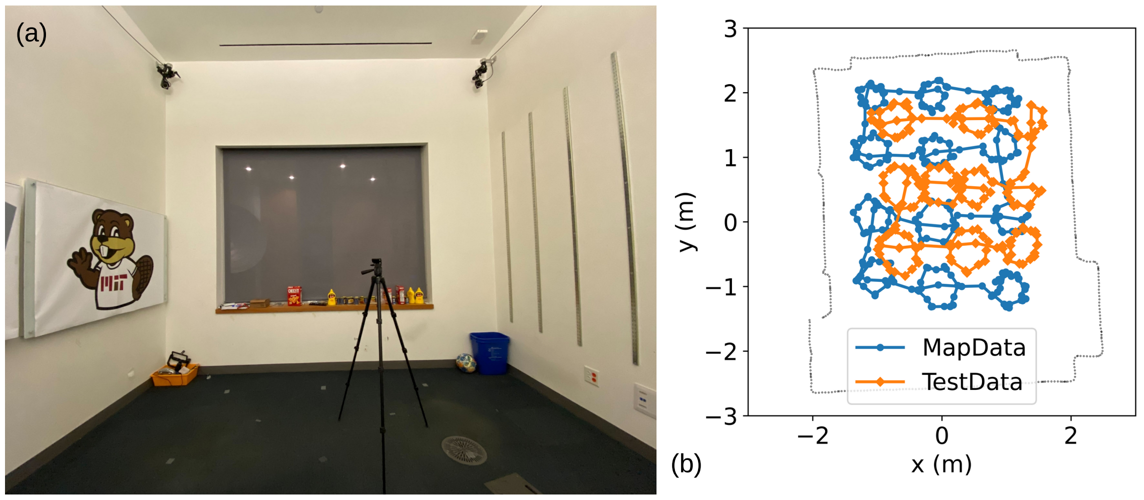


Figure 6-8: Real experiment: (a) the motion capture room and (b) paths for collecting data.

6.5 Results

We present two sections of results. In the first section, we present results comparing different marker placement methods. Next, we show a parameter study about factors that can affect our algorithm and the localization performance. The main metric we analyze is the recall, which is defined as the percent of test images localized within certain thresholds of errors: (5 cm, 5 deg) for simulated experiments and (30 cm, 10

[4]Results on test images uniformly sampled are available in Sec. 6.4.2.

deg) for the real experiment considering errors in the dense map, sensor noise, and large textureless areas. The default hyperparameter v is 90.

6.5.1 Comparison of marker placement methods

As optimized marker placements in Fig. 6-9 show, our algorithm focuses on placing markers around low-scoring areas and improves mean localizability scores by a large margin. For example, the largest room in the studio model only receives a single marker (marker 9 on the top right of the studio) since the room already possesses good localizability scores even with no markers.

Optimized marker placements consistently outperform no marker placement, random placements, and uniform placements on the recall. After placing 20 markers, our algorithm improves the recall by over 1.5 percentages for the apartment model, 3.0 percentages for the studio model, 20.0 percentages for the office model, and over 20.0 percentages for the room scene. Note that the area of the apartment model is very big and the model has attained a high recall 85% with no assistance of markers so the increment of recall for the apartment model was expected to be lower than that for the other models. The real experiment in the room scene shows that our algorithm is on par with markers placed by a human. Although our experiment demonstrates the efficacy of optimizing marker placements in 3D models for realworld applications, we emphasize that the efficacy relies on the similarity between rendered and real images. Vision features in rendered images can be affected by many factors including mesh quality and lighting. For example, we covered the glass door in the room by a well-textured poster to reduce the difficulty in 3D reconstruction. In addition, if one has quality real RGB-D data at feasible camera poses, the textured mesh is not needed for using our marker placement algorithm.

6.5.2 Parameter study

We design four experiment groups and change one of the default parameters in each experiment group. The experiment groups are 1) different values of v in the greedy

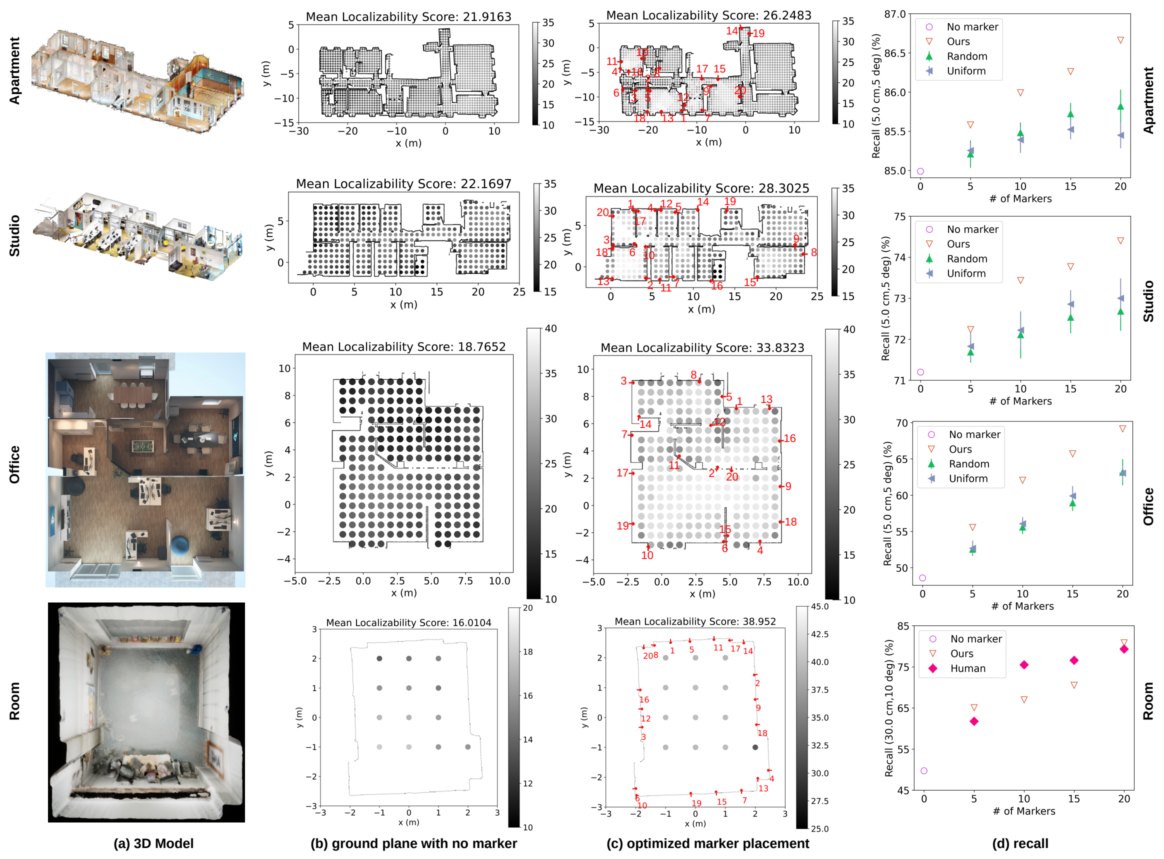


Figure 6-9: Results for all scenes: (a) 3D models, (b) ground plane space with no markers where darker dots indicate lower localizability scores, (c) optimized marker placements where the red arrows represent optimized marker placements and the numbers beside the arrows indicate the order of marker placements, and (d) the recall in camera localization experiments. We exclude camera poses near the bottom of the room where a table occupies.

algorithm, 2) enabling/disabling marker detection in the localization module, 3) low-scoring/uniform test data and 4) enabling/disabling the analysis of feature similarity, as seen in Table 6.2. The default setting is with $v = 90$, marker detection enabled, the low-scoring test data that has more test images in low-scoring areas in the ground plane, and the similarity analysis where similar 3D points are downweighted in the localizability score. For the parameter study, we use the office model.

Too large or small values of hyperparameter v incur lower improvements of the recall. As explained in Sec. 18, lower v favors markers that cover larger areas while greater v tends to stress the worst single camera pose. Table 6.2 shows that the default value ($v = 90$) consistently outperforms small value 50 and large value 99,

Table 6.2: Parameter study about the hyperparameter v, the test data, enabling/disabling marker detection, and enabling/disabling the similarity analysis.

Experiment group	Recall of test images with k markers (%)				
	$k = 0$	5	10	15	20
$v = 90$ (df.)	**48.6**	**55.5**	**62.1**	**65.7**	**69.2**
$v = 99$	48.6	55.5	60.4	64.5	67.4
$v = 70$	48.6	54.8	61.1	63.2	66.6
$v = 50$	48.6	54.3	57.6	63.9	66.8
Marker detect. on (df.)	**48.6**	**55.5**	**62.1**	**65.7**	**69.2**
Marker detect. off	48.6	55.2	60.7	64.2	67.5
Low-scoring data (df.)	48.6	55.5	62.1	65.7	69.2
Unif. test data	**57.4**	**63.7**	**68.4**	**72.1**	**74.8**
Similarity analysis (df.)	**48.6**	**55.5**	**62.1**	**65.7**	**69.2**
Sim. analysis disabled	48.6	55.4	61.8	65.0	67.8

indicating that the default attains a good balance between area coverage and helping the worst cases.

The localizability score can be a good indicator of localization errors. Table 6.2 shows that uniform test samples enjoy greater recall than test samples that stress low-scoring areas by at least 5 percentages. Fig. 6-10b indicates a statistically significant, negative correlation between the localizability score and the localization error. In addition, we show recall of uniformly sampled test images in Table 6.3.

Both the visual appearance and decoded label of markers are helpful for localization. We disable marker detection in the localization module (Fig. 6-7) to investigate its impact on the recall. Table 6.2 shows that markers still improve the recall even though the detector is turned off. The reason is that the visual appearance of markers is still helpful for coarse localization and pose estimation in the localization module.

Deactivating the analysis of feature similarity decreases the recall. Fig. 6-10a presents the marker placement after disabling the similarity analysis (i.e., no scaling in (6.4)). The first five markers remain in the same positions as those guided by the similarity analysis (Fig. 6-9c). Thus the recall does not change significantly

Table 6.3: Recall (%) of test images that are uniformly sampled. Rand. refers to random marker placements and Unif. refers to uniform marker placements.

Scene (NoMarker)	Method	Mean±STD with k markers			
		$k = 5$	$k = 10$	$k = 15$	$k = 20$
Apt. (88.2)	Ours	**88.6**	**88.8**	**89.2**	**89.5**
	Rand.	88.5±0.1	88.8±0.1	89.0±0.1	89.1±0.2
	Unif.	88.4±0.1	88.6±0.1	88.7±0.1	88.9±0.1
Studio (80.3)	Ours	**81.4**	**82.7**	**83.0**	**83.1**
	Rand.	80.8±0.3	81.4±0.4	81.2±0.4	81.4±0.7
	Unif.	80.9±0.2	81.2±0.3	81.6±0.5	81.8±0.2
Office (57.4)	Ours	**63.7**	**68.4**	**72.1**	**74.8**
	Rand.	60.7±0.7	63.0±1.0	66.7±1.5	69.1±1.8
	Unif.	60.0±0.6	63.0±0.7	67.6±0.9	69.9±1.3

until placing 10 markers, as shown in the last group in Table 6.2. The decrease in the recall with no similarity analysis justifies the efficacy of downweighting similar features in the computation of the localizability score.

6.6 Sensitivity study of marker sizes and positions

It is quite likely that a user will not be able to place fiducial markers exactly at the positions computed by the OMP algorithm; meanwhile, different users may print fiducial markers with different sizes. Thus we investigate the impact of position deviations and marker sizes on the recall. For the sensitivity study, we used the office model.

Enlarging markers up to a certain size keeps increasing the recall. Fig. 6-11a shows that, under 50 cm, larger tag widths lead to greater recall (note that the threshold 50 cm should correlate with environments). Excessively large sizes can degrade the recall because the markers become too big to be detected from nearby views.

Mild position deviations slightly degrade the performance of the optimized marker placement. All 20 markers planned by the OMP algorithm were

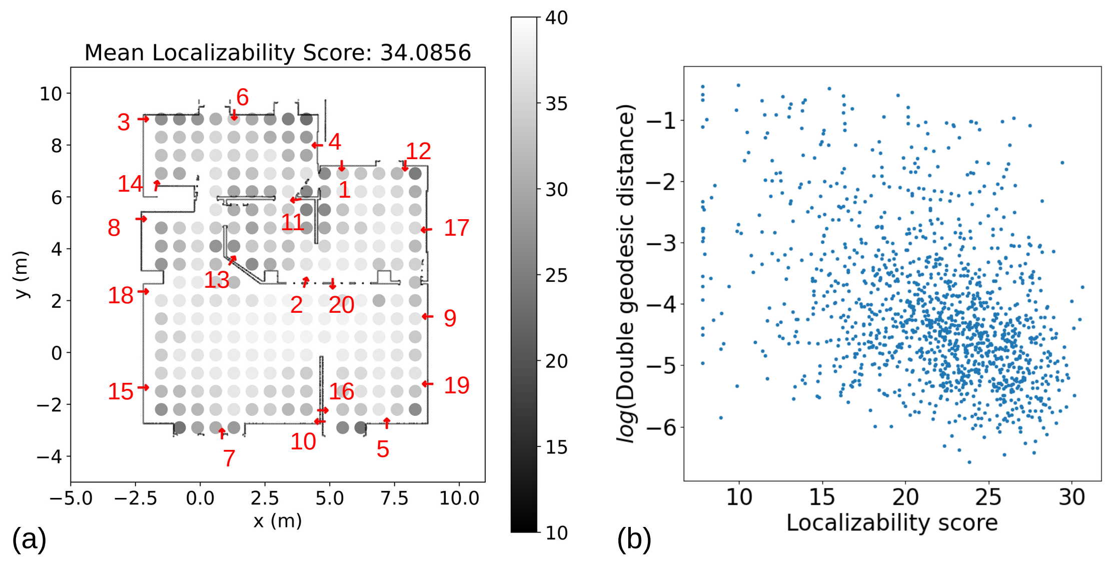


Figure 6-10: Parameter study: (a) the optimized marker placement after disabling the similarity analysis and (b) scatter plot of the localizability score and the log of estimation error of test images. The error is computed as the double Geodesic distance, $\sqrt{\delta_R^2 + \delta_t^2}$. To avoid outliers, samples are admitted to the plot only if the translation and rotation errors are within $(50cm, 50deg)$. The Pearson correlation coefficient and p-value for testing non-correlation is $(-0.41, 2.4 \times 10^{-55})$.

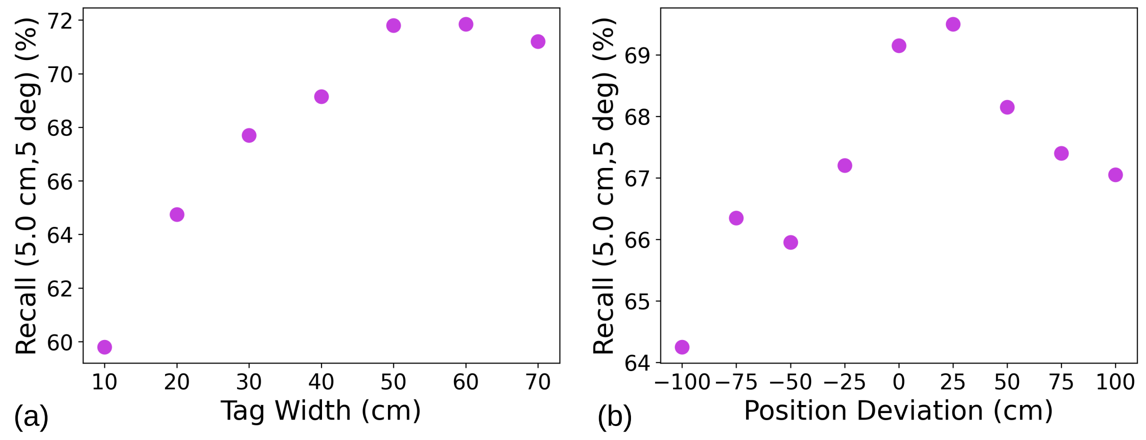


Figure 6-11: Sensitivity study: (a) re-sizing tags and (b) applying different position deviations to marker poses planned by the OMP algorithm.

moved left or right by certain distances to implement position deviations. Fig. 6-11b shows the recall can decrease by 2% in the presence of ±0.25 meters position deviations and by 5% in the presence of ±1 meter position deviations, compared with zero position deviation. However, marker placements with the position deviations still outperform no marker placement by a large margin (∼ 15 percentages in the recall).

6.7 Summary

This work provides a promising foundation for optimizing and evaluating marker placement for improved visual localization. Our OMP algorithm defines localizability scores for different areas in the scene and uses a greedy algorithm to find the best marker placements in the sense of increased localizability scores. We applied the OMP algorithm to four scenes and demonstrated that OMP consistently improves camera localization recall compared to random and uniform marker placements. We believe that our marker placement approach is also useful for SLAM. However, our approach could be further extended to compute optimal marker placement for specific tasks in SLAM. One potential idea involves extending the localizability score to a trackability score that incorporates uncertainty propagation along a robot path while restricting feasible camera poses to the operating area of the robot.

The OMP algorithm only considers placing markers in a scene model (i.e., mapped areas in the scene), however, regions in the scene which are challenging for mapping are also likely to be good locations for placing markers. Thus, it would be worth exploring ways to extend the algorithm to prioritize marker placements in regions that are either partially or inadequately mapped. Further research is also needed to compute more accurate localizability scores and explore more efficient optimization methods beyond the greedy algorithm, including: (1) joint optimization of marker poses and sizes, (2) extending the single-layer ground plane to multi-layer planes for deploying markers in multi-storey structures, (3) using non-Gaussian distribution estimation techniques to compute localizability scores, and (4) applying submodular optimization to jointly select multiple best markers together with fewer iterations.

Chapter 7

Conclusion

This thesis seeks scalable, expressive density representations, beyond the Gaussian distribution, for the improved approximation of posterior distributions encountered in SLAM. We achieve scalable full posterior inference for SLAM in highly non-Gaussian settings, by leveraging sparsity structures in factor graphs and various density representations to address the scalability challenge and the expressivity challenge, respectively. Our algorithms enable the robot to continuously reason the uncertainty in localization and mapping even with few, ambiguous observations about the world. In addition to SLAM algorithms, we apply full posterior inference to identify difficult areas for visual localization in an environment.

Chapter 3 introduces nested sampling methods to directly draw samples from the posterior distribution encountered in SLAM problems, pursuing the *bona fide* shape of the posterior at the expense of computation resources. We demonstrate that NSFG can serve as a reference solution for the posterior, aiding accuracy evaluation of density approximations found by scalable inference algorithms. In Chapter 4, we present a real-time algorithm, GAPSLAM, that precisely infers non-Gaussian/multi-modal marginal posteriors encountered in SLAM. In addition, we develop a streaming platform that bridges mobile devices and servers via web applications to conduct live demo of object-based SLAM, featured by the sharing of mapping results among online peers and the continuous visualization of the uncertainty in localization and mapping. Chapter 5 introduces a novel algorithm, NF-iSAM, that achieves incremental updates for modeling and sampling the joint posterior encountered in SLAM. Our approach show an improved approximation of the joint posterior in highly non-Gaussian settings due to nonlinear measurement models and non-Gaussian (e.g., multi-modal) factors. Chapter 6 provides a promising foundation for optimizing and evaluating marker placement for improved visual localization. Our OMP algorithm defines localizability scores, via performing full posterior inference in visual localization, and uses a greedy algorithm to find the best marker placements in the sense of increased localizability scores. We have evaluated OMP within this testbed and demonstrate an improvement in the localization rate by up to 20 percent on four different scenes.

7.1 Future work

Contributions in this thesis can be extended further in several ways. First, while the NF-iSAM algorithm is scalable owing to incremental inference, our implementation has not reached real-time operation since training neural networks to model normalizing flows is time-consuming. Further research is needed to explore more efficient implementation strategies that can lead us closer to real-time operation, including: 1) leveraging more efficient learning-based density models and 2) utilizing hardware accelerators for training neural networks. Second, this thesis focuses on non-Gaussian posteriors that are incurred by physical measurements and model-based likelihoods. Future work on full posterior inference can explore how to incorporate data-driven observations and learned, highly uncertain likelihoods. Last, there are a number of opportunities to utilize the uncertainty in perception results to improve autonomous systems, including the detection of anomaly in the perception pipeline and improved decision-making for active perception. We believe that advances in uncertainty-aware perception will assume an increasingly vital role in propelling the advancement of life-long, self-diagnosing, and self-improving autonomous systems.

Bibliography

[1] Pratik Agarwal, Gian Diego Tipaldi, Luciano Spinello, Cyrill Stachniss, and Wolfram Burgard. Robust map optimization using dynamic covariance scaling. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 62–69, 2013.

[2] Sameer Agarwal, Keir Mierle, and The Ceres Solver Team. Ceres Solver, 3 2022.

[3] Brandon Amos, Lei Xu, and J Zico Kolter. Input convex neural networks. In *Proc. Int. Conf. Mach. Learn.*, pages 146–155, 2017.

[4] M Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Trans. Signal Process.*, 50(2):174–188, 2002.

[5] T. Bailey and H.F. Durrant-Whyte. Simultaneous localisation and mapping (SLAM): Part I. *IEEE Robot. Autom. Mag.*, 13(2):99–110, Jun. 2006.

[6] Ricardo Baptista, Olivier Zahm, and Youssef Marzouk. On the representation and learning of monotone triangular transport maps. *arXiv preprint arXiv:2009.10303*, 2020.

[7] Maximilian Beinhofer, Jörg Müller, and Wolfram Burgard. Effective landmark placement for accurate and reliable mobile robot navigation. *Robot. Auton. Syst.*, 61(10):1060–1069, Oct. 2013.

[8] Michael Betancourt. A conceptual introduction to hamiltonian monte carlo, 2018.

[9] Christopher M Bishop. *Pattern recognition and machine learning*, chapter 8, page 365. Springer, New York, USA, 2006.

[10] Christopher M Bishop. *Pattern recognition and machine learning*, chapter 4, page 213. Springer, New York, USA, 2006.

[11] Jose Blanco, Juan-Antonio Fernández-Madrigal, and Javier González. Efficient probabilistic range-only SLAM. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 1017–1022. IEEE, 2008.

[12] Jose Blanco, Javier González, and Juan-Antonio Fernández-Madrigal. A pure probabilistic approach to range-only SLAM. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 1436–1441, 2008.

[13] Jose-Luis Blanco, Javier González, and Juan-Antonio Fernández-Madrigal. Optimal filtering for non-parametric observation models: applications to localization and slam. *The International Journal of Robotics Research*, 29(14):1726–1742, 2010.

[14] Vladimir Igorevich Bogachev, Aleksandr Viktorovich Kolesnikov, and Kirill Vladimirovich Medvedev. Triangular transformations of measures. *Sbornik: Mathematics*, 196(3):309–335, 2005.

[15] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[16] Rodney Brooks. A human in the loop: AI won't surpass human intelligence anytime soon. *IEEE Spectrum*, 58(10):48–49, 2021.

[17] Johannes Buchner. Nested sampling methods. *Statistic Surveys*, 17:169–215, 2023.

[18] C. Cadena et al. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Trans. Robot.*, 32(6):1309–1332, Dec. 2016.

[19] Carlos Campos, Richard Elvira, Juan J. Gómez, José M. M. Montiel, and Juan D. Tardós. ORB-SLAM3: An accurate open-source library for visual, visual-inertial and multi-map SLAM. *IEEE Trans. Robot.*, 37(6):1874–1890, 2021.

[20] Guillaume Carlier, Alfred Galichon, and Filippo Santambrogio. From Knothe's transport to Brenier's map and a continuation method for optimal transport. *SIAM J. Math. Anal.*, 41(6):2554–2576, 2010.

[21] Yingying Chen, John-Austen Francisco, Wade Trappe, and Richard P Martin. A practical approach to landmark deployment for indoor localization. In *Proc. IEEE Commun. Soc. Sens. Ad Hoc Commun. Netw.*, volume 1, pages 365–373, 2006.

[22] Contributors and Dependencies. Caesar.jl v0.10.2, 2021.

[23] Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proc. Annu. Conf. Comput. Graph. Interact. Tech.*, pages 303–312, 1996.

[24] RALPH D'agostino and Egon S Pearson. Tests for departure from normality. *Biometrika*, 60(3):613–622, 1973.

[25] Andrew J Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proc. IEEE Int. Conf. Comput. Vis.*, volume 3, pages 1403–1403, 2003.

[26] Joseph DeGol, Timothy Bretl, and Derek Hoiem. Chromatag: A colored marker and fast detection algorithm. In *Proc. IEEE Int. Conf. Comput. Vis.*, pages 1481 – 1490, 2017.

[27] Joseph DeGol, Timothy Bretl, and Derek Hoiem. Improved structure from motion using fiducial marker matching. In *Proc. Eur. Conf. Comput. Vis.*, pages 281–296, 2018.

[28] Frank Dellaert. Factor graphs and GTSAM: A hands-on introduction. Technical report, Georgia Institute of Technology, 2012.

[29] Frank Dellaert et al. borglab/gtsam, May 2022.

[30] Frank Dellaert and Michael Kaess. Square root SAM: Simultaneous localization and mapping via square root information smoothing. *Int. J. Robot. Res.*, 25(12):1181–1203, 2006.

[31] Frank Dellaert and Michael Kaess. Factor graphs for robot perception. *Found. Trends Robot.*, 6(1-2):1–139, 2017.

[32] Xinke Deng, Arsalan Mousavian, Yu Xiang, Fei Xia, Timothy Bretl, and Dieter Fox. PoseRBPF: A Rao–Blackwellized particle filter for 6-D object pose tracking. *IEEE Trans. Robot.*, 37(5):1328–1342, Oct. 2021.

[33] Joseph Djugash, Bradley Hamner, and Stephan Roth. Navigating with ranging radios: Five data sets with ground truth. *J. Field Robot.*, 26(9):689–695, 2009.

[34] Kevin Doherty, Dehann Fourie, and John J Leonard. Multimodal semantic slam with probabilistic data association. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 2419–2425, 2019.

[35] Kevin J Doherty, David P Baxter, Edward Schneeweiss, and John J Leonard. Probabilistic data association via mixture models for robust semantic SLAM. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 1098–1104, 2020.

[36] Arnaud Doucet, Adam M Johansen, et al. A tutorial on particle filtering and smoothing: Fifteen years later. *Handbook of nonlinear filtering*, 12(656-704):3, 2009.

[37] Conor Durkan, Artur Bekasov, Iain Murray, and George Papamakarios. Neural spline flows. In *Proc. Adv. Neural Inform. Process. Syst.*, volume 32, 2019.

[38] Tarek A El Moselhy and Youssef M Marzouk. Bayesian inference with optimal maps. *J. Comput. Phys.*, 231(23):7815–7850, 2012.

[39] Epic Games. Unreal engine.

[40] F Feroz, MP Hobson, and M Bridges. Multinest: an efficient and robust bayesian inference tool for cosmology and particle physics. *Monthly Notices of the Royal Astronomical Society*, 398(4):1601–1614, 2009.

[41] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[42] D. Fourie, P. V. Teixeira N. R. Rypkema, S. Claassens, E. Fischell, and J. Leonard. Towards real-time non-Gaussian SLAM for underdetermined navigation. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Oct. 2020.

[43] Dehann Fourie. *Multi-modal and inertial sensor solutions for navigation-type factor graphs*. PhD thesis, Massachusetts Institute of Technology, 2017.

[44] Dehann Fourie, Antonio T. Espinoza, Michael Kaess, and John J. Leonard. Characterizing marginalization and incremental operations on the Bayes tree. In *Proc. Int. Workshop Algorithmic Found. Robot.*, pages 227–242, 2020.

[45] Dehann Fourie, John J Leonard, and Michael Kaess. A nonparametric belief solution to the Bayes tree. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 2189–2196. IEEE, 2016.

[46] Jiahui Fu, Qiangqiang Huang, Kevin Doherty, Yue Wang, and John J Leonard. A multi-hypothesis approach to pose ambiguity in object-based SLAM. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 7639–7646, 2021.

[47] Xiang Gao, Rui Wang, Nikolaus Demmel, and Daniel Cremers. LDSO: Direct sparse odometry with loop closure. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 2198–2204, 2018.

[48] Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang, and Hang-Fei Cheng. Complete solution classification for the perspective-three-point problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(8):930–943, Aug. 2003.

[49] S. Garrido-Jurado, R. Muñoz-Salinas, F.J. Madrid-Cuevas, and M.J. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognit.*, 47(6):2280–2292, 2014.

[50] Gregor Gebhardt, Andras Kupcsik, and Gerhard Neumann. The kernel Kalman rule—efficient nonparametric inference with recursive least squares. In *Proc. AAAI Conf. Artif. Intell.*, volume 31, 2017.

[51] J. González, J. L. Blanco, C. Galindo, A. Ortiz-de Galisteo, J. A. Fernández-Madrigal, F. A. Moreno, and J. L. Martínez. Mobile robot localization based on Ultra-Wide-Band ranging: A particle filter approach. *Robot. Auton. Syst.*, 57(5):496–507, 2009.

[52] Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. A kernel two-sample test. *J. Mach. Learn. Res.*, 13(1):723–773, 2012.

[53] Paul D Groves. Shadow matching: A new GNSS positioning technique for urban canyons. *The journal of Navigation*, 64(3):417–430, 2011.

[54] Weiqiao Han, Ashkan Jasour, and Brian Williams. Non-gaussian risk bounded trajectory optimization for stochastic nonlinear systems in uncertain environments. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 11044–11050, 2022.

[55] WJ Handley, MP Hobson, and AN Lasenby. Polychord: next-generation nested sampling. *Monthly Notices of the Royal Astronomical Society*, 453(4):4384–4398, 2015.

[56] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[57] Pinar Heggernes and Pontus Matstoms. Finding good column orderings for sparse QR factorization. In *Proc. SIAM Conf. Sparse Matrices*, pages 1–28, 1996.

[58] M. Herdin, N. Czink, H. Ozcelik, and E. Bonek. Correlation matrix distance, a meaningful measure for evaluation of non-stationary MIMO channels. In *2005 IEEE 61st Vehicular Technology Conference*, volume 1, pages 136–140, 2005.

[59] Matthew D Hoffman, Andrew Gelman, et al. The no-u-turn sampler: adaptively setting path lengths in hamiltonian monte carlo. *J. Mach. Learn. Res.*, 15(1):1593–1623, 2014.

[60] Ming Hsiao and Michael Kaess. MH-iSAM2: Multi-hypothesis iSAM using Bayes tree and hypo-tree. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 1274–1280, 2019.

[61] Qiangqiang Huang et al. Incremental non-Gaussian inference for SLAM using normalizing flows. *IEEE Trans. Robot.*, pages 1–18, 2022.

[62] Qiangqiang Huang and John J. Leonard. GAPSLAM: Blending Gaussian approximation and particle filters for real-time non-Gaussian SLAM. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2023.

[63] Qiangqiang Huang, Alan Papalia, and John J. Leonard. Nested sampling for non-Gaussian inference in SLAM factor graphs. *IEEE Robot. Autom. Lett.*, 7(4):9232–9239, Oct. 2022.

[64] Qiangqiang Huang, Can Pu, Dehann Fourie, Kasra Khosoussi, Jonathan P. How, and John J. Leonard. NF-iSAM: Incremental smoothing and mapping via normalizing flows. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 1095–1102, 2021.

[65] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. Int. Conf. Mach. Learn.*, pages 448–456, 2015.

[66] Priyank Jaini, Kira A Selby, and Yaoliang Yu. Sum-of-squares polynomial flow. In *Proc. Int. Conf. Mach. Learn.*, pages 3009–3018, 2019.

[67] Hervé Jégou, Florent Perronnin, Matthijs Douze, Jorge Sánchez, Patrick Pérez, and Cordelia Schmid. Aggregating local image descriptors into compact codes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(9):1704–1716, Sep. 2012.

[68] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. YOLO by Ultralytics, Jan. 2023.

[69] Damien B Jourdan and Nicholas Roy. Optimal sensor placement for agent localization. *ACM Trans. Sens. Netw.*, 4(3):1–40, May 2008.

[70] Michael Kaess and Frank Dellaert. Covariance recovery from a square root information matrix for data association. *Robot. Auton. Syst.*, 57(12):1198–1210, Dec. 2009.

[71] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. iSAM2: Incremental smoothing and mapping using the Bayes tree. *Int. J. Robot. Res.*, 31(2):216–235, 2012.

[72] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Trans. Robot.*, 24(6):1365–1378, Dec. 2008.

[73] Michael Kazhdan and Hugues Hoppe. Screened Poisson surface reconstruction. *ACM Trans. Graph.*, 32(3):1–13, 2013.

[74] Kee Eung Kim and Hyun Soo Park. Imitation learning via kernel mean embedding. In *Proc. AAAI Conf. Artif. Intell.*, volume 32, pages 3415–3422, 2018.

[75] Sanggyun Kim, Rui Ma, Diego Mesa, and Todd P Coleman. Efficient Bayesian inference methods via convex optimization and optimal transport. In *Proc. IEEE Int. Symp. Inf. Theory*, pages 2259–2263, 2013.

[76] Ivan Kobyzev, Simon Prince, and Marcus A Brubaker. Normalizing flows: An introduction and review of current methods. *IEEE Trans. Pattern Anal. Mach. Intell.*, 43(11):3964–3979, 2020.

[77] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g2o: A general framework for graph optimization. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 3607–3613, 2011.

[78] N.M. Kwok and G. Dissanayake. An efficient multiple hypothesis filter for bearing-only SLAM. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 736–741, 2004.

[79] John Lafferty, Han Liu, and Larry Wasserman. Sparse nonparametric graphical models. *Stat. Sci.*, 27(4):519–537, 2012.

[80] Kevin Lai, Liefeng Bo, and Dieter Fox. Unsupervised feature learning for 3D scene labeling. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 3050–3057, 2014.

[81] Zike Lei, Xi Chen, Ying Tan, Xiang Chen, and Li Chai. Optimization of directional landmark deployment for visual observer on SE(3). *IEEE Trans. Ind. Electron.*, pages 1–10, 2022.

[82] Thomas Lemaire, Simon Lacroix, and Joan Sola. A practical 3D bearing-only SLAM algorithm. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 2449–2454, 2005.

[83] Andrew W Long, Kevin C Wolfe, Michael J Mashner, and Gregory S Chirikjian. The banana distribution is Gaussian: A localization study with exponential coordinates. *Proc. Robot.: Sci. Syst.*, pages 265–272, 2013.

[84] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.*, 60(2):91–110, Nov. 2004.

[85] Ziqi Lu, Qiangqiang Huang, Kevin Doherty, and John J. Leonard. Consensus-informed optimization over mixtures for ambiguity-aware object SLAM. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 5432–5439, 2021.

[86] Fahira Afzal Maken, Fabio Ramos, and Lionel Ott. Stein icp for uncertainty estimation in point cloud matching. *IEEE Robot. Autom. Lett.*, 7(2):1063–1070, Apr. 2022.

[87] John D Martin, Kevin Doherty, Caralyn Cyr, Brendan Englot, and John J Leonard. Variational filtering with copula models for SLAM. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 5066–5073, 2020.

[88] Diego Mesa, Sanggyun Kim, and Todd Coleman. A scalable framework to transform samples from one continuous distribution to another. In *Proc. IEEE Int. Symp. Inf. Theory*, pages 676–680, 2015.

[89] Daniel Meyer-Delius, Maximilian Beinhofer, Alexander Kleiner, and Wolfram Burgard. Using artificial landmarks to reduce the ambiguity in the environment of a mobile robot. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 5173–5178, 2011.

[90] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. Fast-SLAM: A factored solution to the simultaneous localization and mapping problem. In *Proc. AAAI Natl. Conf. Artif. Intell.*, pages 593–598, 2002.

[91] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. Fast-SLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proc. Int. Joint Conf. Artif. Intell.*, pages 1151–1156, 2003.

[92] Rafael Munoz-Salinas and Rafael Medina-Carnicer. UcoSLAM: Simultaneous localization and mapping by fusion of keypoints and squared planar markers. *Pattern Recognit.*, 101:107193, May 2020.

[93] Raul Mur-Artal and Juan D Tardós. ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Trans. Robot.*, 33(5):1255–1262, Oct. 2017.

[94] Kieran A Murphy, Carlos Esteves, Varun Jampani, Srikumar Ramalingam, and Ameesh Makadia. Implicit-PDF: Non-parametric representation of probability distributions on the rotation manifold. In *Proc. Int. Conf. Mach. Learn.*, pages 7882–7893, 2021.

[95] José Neira and Juan D Tardós. Data association in stochastic mapping using the joint compatibility test. *IEEE Trans. Robot. Autom.*, 17(6):890–897, 2001.

[96] Paul Newman and John Leonard. Pure range-only sub-sea SLAM. In *Proc. IEEE Int. Conf. Robot. Autom.*, volume 2, pages 1921–1926, 2003.

[97] Lachlan Nicholson, Michael Milford, and Niko Sünderhauf. QuadricSLAM: Dual quadrics from object detections as landmarks in object-oriented SLAM. *IEEE Robot. Autom. Lett.*, 4(1):1–8, 2018.

[98] Kyel Ok, Katherine Liu, Kris Frey, Jonathan P How, and Nicholas Roy. Robust object-based SLAM for high-speed autonomous navigation. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 669–675, 2019.

[99] Edwin Olson. AprilTag: A robust and flexible visual fiducial system. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 3400–3407, 2011.

[100] Edwin Olson and Pratik Agarwal. Inference on networks of mixtures for robust robot mapping. *Int. J. Robot. Res.*, 32(7):826–840, 2013.

[101] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *J. Mach. Learn. Res.*, 22(57):1–64, 2021.

[102] Matthew D Parno and Youssef M Marzouk. Transport map accelerated Markov chain Monte Carlo. *SIAM/ASA J. Uncertain. Quantif.*, 6(2):645–682, 2018.

[103] Adam Paszke, , et al. Pytorch: An imperative style, high-performance deep learning library. *Proc. Adv. Neural Inform. Process. Syst.*, 32, 2019.

[104] Tim Pfeifer, Sven Lange, and Peter Protzel. Advancing mixture models for least squares optimization. *IEEE Robot. Autom. Lett.*, 6(2):3941–3948, Apr. 2021.

[105] Emilia Pompe, Chris Holmes, and Krzysztof Łatuszyński. A framework for adaptive mcmc targeting multimodal distributions. *The Annals of Statistics*, 48(5):2930–2952, 2020.

[106] Lutz Prechelt. Early stopping — but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer, 1998.

[107] Santhosh Kumar Ramakrishnan et al. Habitat-matterport 3d dataset (HM3D): 1000 large-scale 3d environments for embodied AI. In *Proc. Conf. Neural Inform. Process. Syst. Dataset. Benchmark. Track (Round 2)*, 2021.

[108] Danilo Rezende and Shakir Mohamed. Variational inference with normalizing flows. In *Proc. Int. Conf. Mach. Learn.*, pages 1530–1538, 2015.

[109] Danilo Jimenez Rezende et al. Normalizing flows on tori and spheres. In *Proc. Int. Conf. Mach. Learn.*, pages 8083–8092, 2020.

[110] David M. Rosen, Kevin J. Doherty, Antonio Terán Espinoza, and John J. Leonard. Advances in inference and representation for simultaneous localization and mapping. *Annu. Rev. Control Robot. Auton. Syst.*, 4(1):215–242, Jan. 2021.

[111] David M Rosen, Michael Kaess, and John J Leonard. Robust incremental online inference over sparse factor graphs: Beyond the Gaussian case. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 1025–1032, 2013.

[112] Cosimo Rubino, Marco Crocco, and Alessio Del Bue. 3D object localisation from multi-view image detections. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(6):1281–1294, 2017.

[113] John Salvatier, Thomas V Wiecki, and Christopher Fonnesbeck. Probabilistic programming in Python using PyMC3. *PeerJ Computer Science*, 2:e55, 2016.

[114] John Salvatier, Thomas V Wiecki, and Christopher Fonnesbeck. Probabilistic programming in Python using PyMC3. *PeerJ Computer Science*, 2:1–24, 2016.

[115] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pages 12716–12725, 2019.

[116] Paul-Edouard Sarlin, Mihai Dusmanu, Johannes L Schönberger, Pablo Speciale, Lukas Gruber, Viktor Larsson, Ondrej Miksik, and Marc Pollefeys. Lamar: Benchmarking localization and mapping for augmented reality. In *Proc. Eur. Conf. Comput. Vis.*, pages 686–704, 2022.

[117] Glenn R Shafer and Prakash P Shenoy. Probability propagation. *Ann. Math. Artif*, 2(1):327–351, 1990.

[118] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Proc. Field Serv. Robot.*, pages 621–635. Springer, 2018.

[119] Roshan Shariff, András György, and Csaba Szepesvári. Exploiting symmetries to construct efficient MCMC algorithms with an application to SLAM. In *Artificial Intelligence and Statistics*, pages 866–874. PMLR, 2015.

[120] John Skilling. Nested sampling for general Bayesian computation. *Bayesian Anal.*, 1(4):833–859, 2006.

[121] J. Sola, A. Monin, M. Devy, and T. Lemaire. Undelayed initialization in bearing only SLAM. In *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 2499–2504, 2005.

[122] Alessio Spantini, Ricardo Baptista, and Youssef Marzouk. Coupling techniques for nonlinear ensemble filtering. *arXiv preprint arXiv:1907.00389*, 2019.

[123] Joshua S Speagle. dynesty: a dynamic nested sampling package for estimating Bayesian posteriors and evidences. *Monthly Notices Roy. Astronomical Soc.*, 493(3):3132–3158, 2020.

[124] Cyrill Stachniss, Giorgio Grisetti, and Wolfram Burgard. Information gain-based exploration using rao-blackwellized particle filters. In *Proc. Robot.: Sci. Syst.*, volume 2, pages 65–72, 2005.

[125] Erik B Sudderth, Alexander T Ihler, William T Freeman, and Alan S Willsky. Nonparametric belief propagation. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, pages 605–612, 2003.

[126] Niko Sünderhauf, Marcus Obst, Sven Lange, Gerd Wanielik, and Peter Protzel. Switchable constraints and incremental smoothing for online mitigation of non-line-of-sight and multipath effects. In *Proc. IEEE Intell. Veh. Symp.*, pages 262–268, 2013.

[127] Martin Sundermeyer, Tomas Hodan, Yann Labbe, Gu Wang, Eric Brachmann, Bertram Drost, Carsten Rother, and Jiri Matas. Bop challenge 2022 on detection, segmentation and pose estimation of specific rigid objects. *arXiv:2302.13075*, 2023.

[128] Hyungsuk Tak, Xiao-Li Meng, and David A van Dyk. A repelling–attracting metropolis algorithm for multimodality. *Journal of Computational and Graphical Statistics*, 27(3):479–490, 2018.

[129] Robert E Tarjan and Mihalis Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM J. Comput.*, 13(3):566–579, 1984.

[130] Zachary Teed and Jia Deng. Droid-SLAM: Deep visual SLAM for monocular, stereo, and RGB-D cameras. *Proc. Adv. Neural Inform. Process. Syst.*, 34:16558–16569, 2021.

[131] Sebastian Thrun. Probabilistic robotics. *Commun. ACM*, 45(3):52–57, 2002.

[132] Péter Torma, András György, and Csaba Szepesvári. A Markov-chain Monte Carlo approach to simultaneous localization and mapping. In *Proc. Int. Conf. Artif. Intell. Stat.*, pages 852–859, 2010.

[133] Cédric Villani. *Optimal transport: old and new.* Springer, 2008.

[134] Michael P Vitus and Claire J Tomlin. Sensor placement for improved robotic navigation. *Proc. Robot.: Sci. Syst.*, pages 217–224, 2011.

[135] Lei Wang. *Investigation of shadow matching for GNSS positioning in urban canyons.* PhD thesis, UCL (University College London), 2015.

[136] Heng Yang, Jingnan Shi, and Luca Carlone. TEASER: Fast and certifiable point cloud registration. *IEEE Trans. Robot.*, 37(2):314–333, Apr. 2021.

[137] Shichao Yang and Sebastian Scherer. CubeSLAM: Monocular 3-D object SLAM. *IEEE Trans. Robot.*, 35(4):925–938, 2019.

[138] Ruqi Zhang. *Scalable and Reliable Inference for Probabilistic Modeling.* PhD thesis, Cornell University, 2021.

[139] Zhuming Zhang, Yongtao Hu, Guoxing Yu, and Jingwen Dai. DeepTag: A general framework for fiducial marker design and detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2022.

[140] Zichao Zhang, Torsten Sattler, and Davide Scaramuzza. Reference pose generation for long-term visual localization via learned features and view synthesis. *Int. J. Comput. Vis.*, 129(4):821–844, 2021.

[141] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.

[142] Xingyi Zhou, Rohit Girdhar, Armand Joulin, Philipp Krähenbühl, and Ishan Misra. Detecting twenty-thousand classes using image-level supervision. In *Proc. Eur. Conf. Comput. Vis.*, pages 350–368, 2022.