Visual and lidar-based SLAM by variational bayesian methods

Jiang, Xiaoyue

2020

Jiang, X. (2020). Visual and lidar-based SLAM by variational bayesian methods. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/139813>

<https://doi.org/10.32657/10356/139813>

This work is licensed under a Creative Commons Attribution-NonCommercial 4.0 International License (CC BY-NC 4.0).

Downloaded on 26 Apr 2023 18:07:11 SGT

Visual and Lidar-based SLAM by Variational Bayesian Methods

Jiang Xiaoyue

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

2020

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

09 Sep. 2019

.....

Date

A rectangular box containing a handwritten signature in black ink. The signature appears to be 'Jiang Xiaoyue' written in a cursive style.

.....

Jiang Xiaoyue

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

25 Aug. 2019

.....

Date

.....

Prof. Justin Dauwels

Authorship Attribution Statement

This thesis contains material from 3 papers accepted at conferences in which I am listed as an author.

Chapter 3 is published as Xiaoyue Jiang, Michael Hoy, Hang Yu, and Justin Dauwels, Linear-complexity Stochastic Variational Bayes Inference for SLAM. In *IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1-6, Oct. 2017.

The contributions of the co-authors are as follows:

- Michael Hoy and Justin Dauwels provided the initial project direction.
- I co-derived the algorithm with Hang Yu.
- I designed the experiments, performed all the laboratory work, and prepared the manuscript.
- Michael Hoy, Hang Yu and Justin Dauwels edited the manuscript draft.

Chapter 4 is published as Xiaoyue Jiang, Hang Yu, Michael Hoy and Justin Dauwels, Robust Linear-complexity Approach to Full SLAM Problems: Stochastic Variational Bayes Inference. In *IEEE 90th Vehicular Technology Conference (VTC)*, Sep. 2019.

The contributions of the co-authors are as follows:

- I derived the algorithm with the help of Hang Yu.
- I designed and conducted the experiments.
- I wrote the draft of the manuscript. The manuscript was revised together with Dr. Yu, Dr. Hoy and Prof. Dauwels.

Chapter 5 is published as Xiaoyue Jiang, Hang Yu, Michael Hoy and Justin Dauwels, Variational Bayesian Point Set Registration. In *IEEE 90th Vehicular Technology Conference (VTC)*, Sep. 2019.

The contributions of the co-authors are as follows:

- I co-designed the algorithm with Hang Yu.
- I conducted the experiments, and prepared the manuscript.
- The manuscript was revised by Hang Yu, Michael Hoy, and Justin Dauwels.

Chapter 6 is an extension of Xiaoyue Jiang, Hang Yu, Michael Hoy and Justin Dauwels, Variational Bayesian Point Set Registration. In *IEEE 90th Vehicular Technology Conference (VTC)*, Sep. 2019.

The contributions of the co-authors are as follows:

- I designed the algorithm with the assistance from Hang Yu.
- I designed and conducted the experiments. and prepared the manuscript.
- I prepared the manuscript draft. The manuscript was revised by Hang Yu, Michael Hoy, and Justin Dauwels.

25 Aug. 2019

.....

Date

A rectangular image showing a handwritten signature in black ink on a light-colored background. The signature appears to be 'Jiang Xiaoyue' written in a cursive style.

.....

Jiang Xiaoyue

Acknowledgements

I wish to express my greatest gratitude to my advisor, Prof. Justin Dauwels, who has supported and encouraged me throughout the four years. His continuous support, immense knowledge, patience, and motivation helped me overcoming numerous obstacles I have been facing through my research. Also, his insightful suggestions and encouragement incited me to widen my research from various perspectives. When I am confused and depressed, he led me to the right path. Without his help, I could not have imagined completing this thesis.

Very special gratitude goes out to NTU and STE-NTU corp lab for helping and providing the funding for my research work.

My sincere thanks also go to Dr Hang Yu, Dr Michael Hoy, for their participation, feedback, cooperation and generous help in my research work. They guided and gave me a push when I am at a low ebb. I enjoyed working with them and learned a lot from them in those stimulating discussions. With their continuous help and hearty advice, I have arrived to the me who I am today.

I am also grateful to my friends, who encouraged me to react positively under difficult conditions.

Last but not least, I would like to thank my family: my parents and to my sister for supporting me spiritually throughout my PhD life. Love you, my family. All the experience in the four years, whatever sadness or joy, would be my treasure I cannot abandon in the rest of my life.

“It always seems impossible until it’s done.”

—Nelson Mandela

To my dear family

Abstract

Autonomous vehicles, such as self-driving cars, unmanned aerial vehicles, planetary rovers, and autonomous underwater vehicles, have found applications in a wide variety of domains in recent years, from transportation to searching, rescue, and military. A major challenge for designing autonomous vehicles lies in understanding the surroundings via their onboard sensors. A technology to bridge over this difficulty is Simultaneous Localization and Mapping (SLAM). SLAM continuously scans and “learns” about the operating environment, allowing vehicles to augment the physical world with digital content depending on their locations within this space.

SLAM has been extensively studied over the past decades resulting in a variety of solutions based on different types of sensors, including cameras, Global Positioning System (GPS), Light Detection and Ranging (Lidar), Inertial Measurement Unit (IMU), Radio Detection and Ranging (Radar), and Sound Navigation and Ranging (Sonar). The most widely applied two types are Visual SLAM (vSLAM) and Lidar-based SLAM. Techniques based on vSLAM and Lidar-based SLAM fundamentally investigate a computational problem of localizing a vehicle while simultaneously constructing or updating the environmental map. This estimation can be addressed via two forms: online and offline. Online updates the most recent pose and map, whereas offline approximates the entire path and map. Both forms are of high importance and wide applications in the real world. Existing online algorithms are referred to as filtering. Popular approaches like Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF), Extended Information Filter (EIF), and Rao-Blackwellized conditional particle filter (FastSLAM) belong to this category. These algorithms are of high computational complexity, can hardly be used in scenarios where the number of features is large. Conversely, smoothing approaches consist in the estimation of the offline problem, and typically produce a maximum a posterior probability (MAP) estimate by least square error minimization. Smoothing outputs more accurate estimates, but requires higher computational

cost if there is a large number of frames. Moreover, both filtering and smoothing are sensitive to noise priors as they are fixed during updates. On the other hand, Lidar-based SLAM registers neighboring frames to predict an initial guess before optimization. Prevalent registration algorithms, Iterative Closest Point (ICP), and Normal Distributions Transform (NDT) are either computational demanding or sensitive to outliers, noise, and a portion of missing points.

In recent years, variational Bayesian methods have shown great potential in numerous complex statistical models. Their fast speed, improved scalability, and accuracy motivate us to apply variational Bayesian methods to the SLAM community. In this thesis, we first develop online and offline SLAM algorithms based on stochastic variational Bayesian inference (SVBI), namely, online SVBI and offline SVBI. In both formulations, we model SLAM as a posterior estimation problem, and then employ a Gaussian distribution to approximate this posterior by maximizing the evidence lower bound (ELBO). We restrict the uncertainty matrix of the Gaussian distribution to be diagonal and tri-diagonal in online SVBI and offline SVBI, respectively. Consequently, both have linear computational complexity. In addition, we jointly update the noise priors together with the posterior. Therefore, online and offline SVBI are robust. The map variable is encoded into a state vector for online SVBI, however, shared across all frames in offline SVBI. We have evaluated online and offline SVBI on synthetic and real-life data, obtaining more consistent estimations. This type of SLAM has achieved a superior performance of a 7% error reduction than the traditional algorithms in both online and offline scenarios. Apart from the fundamental estimation, we propose two types of variational Bayesian point set registration for Lidar-based SLAM: variational Bayesian point registration based on Gaussian Mixture Models (GMMs-VB-PSR) and variational Bayesian point registration based on kernel density (KD-VB-PSR). The former is appropriate for clustering data, and the latter is widely applicable. The registration is described as a state space model (SSM), with the transition being the hidden variable. GMMs-VB-PSR and KD-VB-PSR represent the sets as Gaussian Mixture Models (GMMs) and Gaussian kernel densities, respectively. The L2 distance between two GMMs or densities is then interpreted as the “observation”. We employ SVBI to solve the formulated SSM in both methods, resulting in linear complexity and robust performance. Furthermore, GMMs-VB-PSR and KD-VB-PSR adopt quadrature and randomized quasi-Monte Carlo (RQMC) points, respectively, to ease the computational burden. In addition, KD-VB-PSR relies on stochastic

variance reduced gradient (SVRG) method and variance reduction techniques to improve the convergence speed further. The validity of GMMs-VB-PSR and KD-VB-PSR is demonstrated through extensive experimental results, illustrating the effectiveness, robustness, and efficiency.

To summarize, this thesis demonstrates the superiority of the proposed variational Bayesian approaches for visual and Lidar-based SLAM problems, including online and offline SVBI, GMMs-VB-PSR, and KD-VB-PSR. The proposed method could be further employed in a variety of applications such as real robotic platforms, sensor data fusion, data association, and point shape completion.

Contents

Acknowledgements	ix
Abstract	xiii
List of Figures	xxi
List of Tables	xxiii
Symbols and Acronyms	xxv
1 Introduction	1
1.1 Scope and Overview	1
1.2 Major Contributions	2
1.3 Outline of the Thesis	4
2 Literature Review	7
2.1 The SLAM Problem	8
2.1.1 Problem Definition	9
2.1.2 Sensor Types	11
2.1.2.1 Accoustic Sensors	11
2.1.2.2 Laser Range Finders	12
2.1.2.3 Visual Sensors	13
2.1.3 Map Types	14
2.1.3.1 Metric Map	14
2.1.3.2 Qualitative Map	15
2.1.3.3 Hybrid Map	15
2.2 Methods to Process Sensor Data	17
2.2.1 Types of Visual SLAM Processing Methods	17
2.2.1.1 Feature-based vs. Direct Methods	17
2.2.1.2 Sparse and Dense Methods	18
2.2.2 Types of Lidar-based SLAM Processing Methods	19
2.2.2.1 ICP Algorithm	19
2.2.2.2 NDT Algorithm	21
2.3 The Bayesian Approach to Solve SLAM	22

2.3.1	EKF Algorithm	23
2.3.2	UKF Algorithm	24
2.3.3	FastSLAM Algorithm	26
2.3.4	Graph-based SLAM	27
2.4	Variational Bayesian Inference	30
2.4.1	Bayesian Inference Basics	31
2.4.2	Mean field approximation	32
2.5	The Future of SLAM: Traditional Approaches vs. Deep Learning	35
2.5.1	Deep Learning: Definition	36
2.5.2	Learning-based SLAM	38
2.5.3	The Comparison & Future Trends	40
2.6	Conclusion	41
3	Robust Linear-complexity Variational Bayesian Method for On-line SLAM Problems	43
3.1	Introduction	43
3.2	Related Work	44
3.3	Bayesian Formulation of the Online SLAM Model	47
3.4	Variational Bayesian Inference for SLAM	51
3.4.1	Variational Bayesian Inference	51
3.4.2	Stochastic Gradient Ascent	53
3.4.3	Analysis	55
3.5	Simulation Results	55
3.5.1	Simulation Setup	57
3.5.2	Different Noise Levels	58
3.5.3	Different Input Noise Variances	60
3.5.4	Different Numbers of Observations per Frame	63
3.6	Experimental Results	65
3.6.1	Implementation	65
3.6.2	Stereo Cases	67
3.6.3	RGB-D Cases	68
3.6.4	Monocular Cases	70
3.7	Conclusion	73
4	Robust Linear-complexity Stochastic Variational Bayesian Inference Method for Full SLAM Problems	75
4.1	Introduction	75
4.2	Related Work	77
4.3	Bayesian Full SLAM Problem	79
4.4	Stochastic Variational Bayesian Inference for Full SLAM Problem	82
4.4.1	Stochastic Variational Bayesian Inference	82
4.4.2	Algorithm Overview and Analysis	86
4.5	Simulation Results	87
4.5.1	Simulation Setup	87

4.5.2	Different Noise Levels	88
4.5.3	Different Input Noise Variances	89
4.5.4	Different Number of Frames and Features	90
4.6	Experimental Results	91
4.6.1	System Overview	92
4.6.2	Stereo and RGB-D Vision	93
4.6.3	Monocular Vision	94
4.7	Conclusion	96
5	Variational Bayesian Point Set Registration Based on Gaussian Mixture Models	99
5.1	Introduction	100
5.2	Related Work	101
5.2.1	Distance-Optimization-Based Methods	102
5.2.2	Probability-Based Methods	102
5.2.3	Filtering-Based Methods	103
5.3	Bayesian Formulation of Point Registration	104
5.4	Stochastic Variational Bayesian Inference	107
5.5	Experimental Results	109
5.6	Conclusions	112
6	Variational Bayesian Point Set Registration Based on Kernel Density	115
6.1	Related Work	117
6.2	Problem Formulation	118
6.3	Variational Bayesian Point Set Registration	121
6.3.1	Stochastic Variational Bayesian Inference	121
6.3.2	Improvement of the Computational Efficiency	124
6.4	Experimental Results	127
6.4.1	Experiment Setting	127
6.4.2	Evaluation Results	128
6.5	Conclusion	131
7	Conclusion & Future Work	133
7.1	Conclusion	133
7.2	Future Work	135
A	Proofs for Part I	139
A.1	Derivation of ELBO and its equivalence to KL divergence	139
A.2	Derivation of exact gradients	140
	List of Author's Publications	143
	Bibliography	145

List of Figures

1.1	Structure of the thesis.	2
2.1	SLAM process in ORB-SLAM2 system. Graph from Stefan Hoermann, et al., “Object detection on dynamic occupancy grid maps using deep learning and automatic label generation,” in <i>IEEE Intelligent Vehicles Symposium (IV)</i> , 2018.	9
2.2	Front-end and back-end in a SLAM system.	10
2.3	Ultrasonic sensor, RGB-D camera, camera and Velodyne applied in SLAM.	12
2.4	Examples of feature map, occupancy grid map, topological map, and hybrid map.	16
2.5	A pose-graph representation of a SLAM process.	28
2.6	Deep learning architecture.	36
2.7	Basic calculation in a neuron.	37
2.8	Workflows of traditional and deep learning-based data association.	39
3.1	The general feature-based SLAM problem.	48
3.2	Average run time, CPU usage, and memory footprint of of EKF, UKF, FastSLAM, SGO, BA SLAM and online SVBI for different number of observations per frame.	64
3.3	Tracking and local mapping threads in ORB-SLAM2 systems.	66
3.4	Estimated trajectories of ORB-SLAM2* and ORB-VB+SLAM on KITTI sequence 06, 10 and EuRoC sequence V1_02_medium, MH_05_difficult in the stereo mode.	69
3.5	Estimated trajectories of ORB-SLAM2* and ORB-VB+SLAM on TUM sequence Fr3office and Fr3nst in the RGB-D mode.	70
3.6	Estimated trajectories of ORB-SLAM2* and ORB-VB+SLAM on EuRoC sequence V2_02_medium, MH_04_difficult, and TUM sequence Fr3office and Fr3nst in the monocular mode.	73
4.1	Graphical model of the full SLAM problem.	79
4.2	Average run time, CPU usage, and memory footprint of GBS, K-GBS, and offline SVBI for different number of frames and observations per frame.	91
4.3	Run time vs. number of cores for different values of K	92
4.4	Structure of ORB-SLAM2 system.	93

4.5	Estimated trajectories of ORB-SLAM2 and ORB-VB+SLAM in KITTI 02, 06, EuRoC V1_03_difficult, V2_02_medium, and TUM fr1/room, fr3/nstr_tex_near in stereo and RGB-D cases.	95
4.6	Estimated trajectories of ORB-SLAM2 and ORB-VB+SLAM in KITTI 02, 06, EuRoC V1_03_difficult, V2_02_medium, and TUM fr1/room, fr3/nstr_tex_near in monocular case.	97
5.1	Dynamic Bayesian Network of the point sets generation in a sequential system.	104
5.2	Ground truth in the 2D (a) and the 3D (b) case for one dataset. . .	110
5.3	Logarithm of the RMSE for the translation vector $\mathbf{x}^t$ and the rotation vector $\mathbf{x}^\theta$ as a function of time t resulting from all methods in both 2D and 3D case.	112
5.4	Average memory (a) and CPU (b) usages of all the test methods in 2D and 3D cases without downsampling. CPD and CSCIF run out of memory in the 3D case.	113
5.5	Average run time of ICP, NDT, CPD, CSCIF, and GMMs-VB-PSR for different number of points.	113
6.1	Point cloud collected from Velodyne VLP-16 LiDAR.	116
6.2	Average translation and rotation angles RMSE of ICP, NDT, and CPD in scenarios where exist noise, outliers, and missing partial structures.	129
6.2	Average translation and rotation angles RMSE of GMM-L2, CSCIF, and KD-VB-PSR in scenarios where exist noise, outliers, and missing partial structures.	130

List of Tables

2.1	Advantages & disadvantages of learning-based and traditional SLAM.	42
3.1	Linear-complexity stochastic variational Bayes inference for online SLAM problem.	56
3.2	Average RMSEs of EKF, UKF, FastSLAM, SGO, BA SLAM and online SVBI for different noise levels.	59
3.3	Average run time per frame for EKF, UKF, FastSLAM, SGO, BA SLAM and online SVBI under different noise levels.	59
3.4	Average RMSEs of EKF, UKF, FastSLAM, SGO, BA SLAM and online SVBI for different input noise variances.	61
3.5	Average run time per frame for EKF, UKF, FastSLAM, SGO, BA SLAM and online SVBI under different input noise variances.	61
3.6	Average RMSEs of Adaptive EKF, Adaptive UKF, SGO with Huber's loss, and BA SLAM with Huber loss for different input noise variances.	62
3.7	Average run time per frame for Adaptive EKF, Adaptive UKF, SGO with Huber loss, and BA SLAM with Huber loss under different input noise variances.	63
3.8	The average computational cost per frame of ORB-SLAM2* and ORB_VB+SLAM systems on KITTI, EuRoC and TUM dataset in the stereo, RGBD and monocular mode.	66
3.9	Average accuracy of ORB-SLAM2* and ORB_VB+SLAM systems on KITTI dataset in the stereo mode.	67
3.10	Average accuracy of ORB-SLAM2* and ORB_VB+SLAM systems on EuRoC dataset in the stereo mode.	68
3.11	Average accuracy of ORB-SLAM2* and ORB_VB+SLAM systems on TUM dataset in the RGB-D mode.	69
3.12	Average accuracy of ORB-SLAM2* and ORB_VB+SLAM systems on KITTI dataset in the monocular mode.	71
3.13	Average accuracy of ORB-SLAM2* and ORB_VB+SLAM systems on EuRoC dataset in the monocular mode.	72
3.14	Average accuracy of ORB-SLAM2* and ORB_VB+SLAM systems on TUM dataset in the monocular mode.	72
4.1	Average RMSEs of GBS, K-GBS, Online SVBI, and Offline SVBI for different noise levels.	88

4.2	Average run time of calculating a frame for GBS, k-GBS, Online SVBI, and offline SVBI under varying noise levels.	89
4.3	Average RMSEs of GBS, k-GBS, Online SVBI, and Offline SVBI for varying input noise variances.	89
4.4	Average run time of calculating a frame for GBS, K-GBS, Online SVBI, and Offline SVBI under varying input noise variances.	89
4.5	Average accuracy of ORB-SLAM2 and ORB-VB+SLAM systems on KITTI, EuRoC, and TUM datasets in stereo and RGB-D cases.	94
4.6	Average accuracy of ORB-SLAM2 and ORB-VB+SLAM systems on KITTI, EuRoC, and TUM datasets in monocular case.	96
4.7	Average computational cost of ORB-SLAM2 and ORB-VB+SLAM systems on KITTI, EuRoC and TUM datasets in stereo, RGB-D and monocular cases.	97
5.1	Average runtime per frame for all methods in 2D and 3D cases.	111
6.1	Average runtime per frame for ICP, NDT, CPD, GMM.L2, CSCIF, and KD-VB-PSR in scenarios where exist noise, outliers, and missing partial structures.	132

Symbols and Acronyms

Symbols

$\mathcal{R}^n$	the n -dimensional Euclidean space
$\ \cdot\ $	the L2 norm or Euclidean norm
$p(\cdot)$	the probability distribution
$H[\cdot]$	the entropy of a probability distribution
$\mathbb{E}_p$	the expectation over distribution p
$\mathcal{L}$	the evidence lower bound
$(\cdot)^W$	the quantity expressed in the world coordinate
$(\cdot)^R$	the quantity expressed in the vehicle coordinate
$\mathbf{1}$	a column vector with all ones
$\circ$	the element-wise product
$\oslash$	the element-wise division
∇	the gradient vector
$\mathcal{N}(\cdot)$	the normal distribution
$KL(q p)$	the KL divergence of q from p
$\mathcal{T}$	the rigid transformation
I_P	the identity matrix with size $P \times P$
$[C_t]_{ii}$	the i^{th} diagonal entry of matrix C_t
x^T	the transpose of x
$(\mathbf{x}_k)_i$	the i^{th} element of vector x at time instant k
$\mathbf{x}_k^{(t)}$	the value of $\mathbf{x}$ in the t^{th} iteration at time instant k
$\text{diag}(\mathbf{w})$	the diagonal matrix with diagonal vector $\mathbf{w}$
$\mathcal{G}_Z$	the Gaussian Mixture Models that represent point set Z
$\mathcal{K}_Z$	the kernel densities that represent point set Z
d_{L_2}	the L_2 distance
$\int_{\mathbb{R}^3}$	the infinite integral in a 3D space

Acronyms

SLAM	Simultaneous Localization and Mapping
EKF	Extended Kalman Filter
UKF	Unscented Kalman Filter
SEIF	Sparse Extended Information Filter
VBEM	Variational Bayesian Expectation Maximization
MAP	Maximum a Posterior
BA	Bundle Adjustment
SVBI	Stochastic Variational Bayesian Inference
NLLS	Non-linear Least Squares
NLNG	Non-linear Non-Gaussian
RB-PHD	Rao-Blackwellized Probability Hypothesis Density
RFS	Random Finite Set
KL	Kullback-Leibler
EL	Evidence Lower Bound
ToF	Time of Flight
HYMM	Hybrid Metric Maps
RANSAC	Random Sample Consensus
ATE	Absolute Trajectory Error
RMSE	Root Mean Square Error
ICP	Iterative Closest Point
NDT	Normal Distributions Transform
CSCIF	Cubature Split Covariance Intersection Filter
CPD	Coherent Point Drift
GMM-L2	Point Set Registration based on the L2-based Gaussian Mixture Models
GMMs-VB-PSR	Variational Bayesian Inference Based on Gaussian Mixture Models
SSM	State Space Model
SVRG	Stochastic Variance Reduced Gradient
RQMC	Randomized Quasi-Monte-Carlo
KD-VB-PSR	Variational Bayesian Inference Based on Kernel Density
i.e.	in other words
e.g.	for example
a.k.a.	also known as
w.r.t.	with respect to

Chapter 1

Introduction

1.1 Scope and Overview

With the development of computer vision and artificial intelligence, robotic systems have been applied widely in the military, aerospace, automotive, electronics affairs, i.e., autonomous cars, drones, and planetary rovers. The systems rely on the onboard sensors to operate in a complex environment. The acquisition and explanation of the sensor data involve the simultaneous localization and mapping (SLAM) problem, that is, the concurrent construction of the environmental map and the identification of the vehicle's location. SLAM has been explored since 1988. There is a large variety of schemes based on different sensor modalities, like cameras, Lidar, Radar, Sonar, and IMU. Among them, visual and Lidar-based SLAM are the top two being investigated.

The process of mapping and localizing in visual and Lidar-based SLAM can be addressed via two forms: online and offline. The former seeks to recover the present vehicle pose, and the latter optimizes for the entire path. Although an enormous amount of online and offline algorithms have been proposed, some issues remain to be addressed, i.e., the excessive computational burden and unstable convergence. One primary cause is that the traditional methods are limited in their high computational complexity and the reliance on complete noise information. To handle this issue, we propose stochastic variational Bayesian inference for solving online and offline SLAM: online SVBI and offline SVBI. On the other hand, point set registration has been a challenging task for Lidar-based SLAM due to the massive

FIGURE 1.1: Structure of the thesis.

amount of points. The most commonly adopted algorithms, ICP and NDT, often fail to produce reliable results in scenarios where the presence of misalignment, noise, and local extrema. To overcome these difficulties, we propose two variational Bayesian point set registration algorithms, namely, GMMs-VB-PSR and KD-VB-PSR. GMMs-VB-PSR represents point sets as Gaussian mixture models and is suitable for data structures forming in distinct or meaningful groups. However, KD-VB-PSR is based on kernel densities and applicable to any general registration problem. The structure of this thesis is illustrated in Figure 1.1.

1.2 Major Contributions

The contributions of the proposed variational Bayesian inference methods are stated as follows:

- **Online SVBI:** Online estimation plays a significant role in practical SLAM systems. Beyond the common filtering and online smoothing approaches, this thesis proposes a linear-complexity online stochastic variational Bayesian inference (online SVBI) method for the SLAM problem. The online SVBI overcomes traditional limitations. First, the online SVBI avoids the unreliable linearization and approximates the posterior through a Gaussian distribution. Second, the uncertainty matrix of the Gaussian distribution is diagonal, resulting in linear computational complexity. Third, the employed stochastic gradients overcome the problem of local extrema, and consequently, lead to more reliable results. Last but not least, it is more robust to incomplete noise

priors with a joint update of noise parameters. Extensive experiments show that, when applied in a visual SLAM platform, the online SVBI achieves a higher accuracy yet at a low computational time, enabling real-time implementations.

- Offline SVBI:** Besides the online solution, SLAM problems can also be addressed via offline formulations, where the full trajectory is estimated. The most widely used offline algorithm is graph-based SLAM. However, it still has room to be improved. This thesis presents an offline algorithm based on stochastic variational Bayesian inference (offline SVBI). The offline SVBI employs a Gaussian distribution to approximate the posterior. Since the uncertainty matrix of the Gaussian distribution is tri-diagonal, offline SVBI has a linear complexity w.r.t. the number of frames and features. By contrast, the complexity of the graph-based SLAM is cubic with the number of the optimized frames. In addition, offline SVBI estimates the noise variances adaptively, whereas the graph-based SLAM generally fixes them as constants. Consequently, offline SVBI improves robustness. Furthermore, offline SVBI outputs a full distribution, but graph-based SLAM produces solely point estimates. Last, the stochastic gradients in offline SVBI are beneficial for more consistent estimations. Comprehensive experiments on both synthetic and real-life datasets demonstrate that the offline SVBI has improved accuracy, scalability, and robustness.
- GMMs-VB-PSR:** Robust and efficient point set registration is crucial for the Lidar-based SLAM. Traditional registration methods applied in Lidar-based SLAM include ICP and NDT. Although the probability-based and filtering-based registration algorithms have been proposed in the computer vision community, they are not applicable to the SLAM area due to the huge computational cost. This thesis proposes a variational Bayesian point set registration based on Gaussian mixture models (GMMs-VB-PSR). The proposed GMMs-VB-PSR successfully extends filtering-based and probability-based methods to the SLAM application with retaining a low computational cost. In GMMs-VB-PSR, the point sets are represented by GMMs, and the registration process is described as a state space model. The observation is the L2 distance between two GMMs, and the transition to be estimated is encoded as the hidden state. Through this formulation, GMMs-VB-PSR is robust to various types of noise, outliers, and initial misalignment. The

optimization process is utilized by stochastic variational Bayesian inference, resulting in robustness to local extrema and inaccurate noise. To relieve the heavy computational cost, quadrature points are adopted to lower the computational burden further. Experiments conducted on 2D and 3D point data show that GMMs-VB-PSR is robust and can achieve higher accuracy without increasing extra cost, compared to the state-of-the-art algorithms.

- **KD-VB-PSR:** Since GMMs-VB-PSR is suitable for the data has meaningful structures or groups, this thesis proposes an alternative variational Bayesian point set registration algorithm based on kernel densities (KD-VB-PSR) to widen the extent of application. We stick to model the registration problem as a state space model. However, the point sets are represented by two Gaussian kernel densities, and the observation is set to be the L2 distance between two densities instead. To avoid the time-consuming integration in the L2 distance, we employ Randomized Quasi-Monte Carlo (RQMC) points to approximate. We apply stochastic variational Bayesian inference to solve the state space model. During updates, the convergence speed is accelerated through the SVRG algorithm and variance reduction techniques, i.e., control variate and removal of score functions. Tests conducted on real Velodyne data show that KD-VB-PSR is robust in terms of noise, outliers, and missing partial structures with comparable processing speed.

1.3 Outline of the Thesis

Chapter 1 introduces the scope of this thesis and provides an overview of the visual and Lidar-based SLAM systems. The involved issues and practical concerns intrinsically motivate this research work. The major contributions and outline of the thesis are also described briefly.

Chapter 2 reviews the technical and mathematical background of visual and Lidar-based SLAM systems, including the online and offline Bayesian formulations, sensor and map types. In addition, the methods to process sensor data and Bayesian approaches to solve online and offline SLAM problems are also introduced. Their drawbacks inspire the research work in this thesis. The prerequisite knowledge of this thesis, i.e., variational Bayesian inference, is briefly explained. Finally, the

emerging learning-based SLAM frameworks are reviewed, and the future trend of the SLAM community is discussed.

Chapter 3 focuses on the application of variational Bayesian inference in online SLAM problems. Existing filtering and online smoothing approaches heavily rely on well-designed initialization and accurate noise priors. Moreover, filtering has high computational complexity, and online smoothing requires multiple threads or precisely chosen window size. This chapter proposes a linear-complexity stochastic variational Bayesian inference for online SLAM problems. The proposed algorithm estimates noise adaptively and hence is robust. Meanwhile, owing to the stochastic gradients and simulated smoothing-based optimization, it is more capable of escaping from inadequate solutions. Extensive experiments show that, when applied in real SLAM platforms, the proposed method can run in real-time, while yielding higher accuracy compared to the state-of-the-art.

Chapter 4 addresses the offline SLAM solution based on stochastic variational Bayesian inference. The most conventional graph-based SLAM has cubic complexity with the number of frames and thus has slow speed in scenarios where the number of the frames is large. Moreover, graph-based SLAM is sensitive to local extrema as well as inaccurate noise priors. To overcome these challenges, we propose a linear-complexity stochastic variational Bayesian inference for offline estimation. Unlike the online algorithm, the entire robot path and global map are estimated. The uncertainty matrix of the variational distribution is set to be tri-diagonal, resulting in a linear complexity w.r.t. the number of features and frames. Likewise, we correct the noise variances adaptively to enhance the robustness. Experiments conducted on both synthetic and real-life datasets show that the proposed approach can achieve more reliable results when the noise information is incomplete.

Chapter 5 proposes a variational Bayesian point set registration approach based on Gaussian mixture models. By representing the point sets as GMMs, the algorithm is robust to outliers. The registration problem is modeled as a state space model, integrating various sources of noise. The robust linear-complexity stochastic variational Bayesian inference is then employed to solve the state space model. Furthermore, quadrature points are adopted to reduce the computational cost. Experiments conducted on 2D and 3D data show that the proposed method provides higher accuracy with yet a comparable speed.

Chapter 6 presents an alternative registration method when the original data has no distinct or explicit meaningful groups. Instead of leveraging on GMMs, we model each point set as a Gaussian kernel density, and the means of the kernels are the point centroids. This representation requires no extra parameters or pre-processing steps and thus has broad generality. The registration is still viewed as a state space model, where the observation is the L2 distance between two densities. The complicated integration of L2 distance is approximated via Randomized Quasi-Monte Carlo points. SVRG algorithm and variance reduction techniques are employed to improve the convergence speed. Experiments conducted on Velodyne data from KITTI dataset show that the proposed algorithm enhances the robustness to noise, outliers, and missing partial structures without increasing computational cost.

Chapter 7 summarizes this thesis and proposes ideas for future work.

Chapter 2

Literature Review

With the rapid development of vSLAM and Lidar-based SLAM techniques, there has been considerable excitement about the applications ranging from the navigation system to the indoor robotic vacuum. This Chapter presents a brief review to the technical and mathematical background of vSLAM and Lidar-based SLAM systems, i.e. the SLAM definition, sensor and map types (Section 2.1), methods to process sensor data (Section 2.2), Bayesian approaches to solve the SLAM problem (Section 2.3). Despite the advancements, we also point out the limitations of the existing works and propose variational Bayesian inference (Section 2.4) as a solution for performance improvement. In recent years, deep learning has emerged to formulate some low-level SLAM subproblems, such as visual odometry, semantic fusion, and pose optimization. We review the recent works of deep learning-enhanced SLAM and discuss the future trend (Section 2.5).

The vSLAM employs cameras to map and plot a navigation path. It has been receiving significant interests from robotics, augmented reality, and computer vision communities. Just like the human eyes be the primary source to access external information, vSLAM has a similar characteristic for scene recognition. Typically a vSLAM system consists of two parts, the front end, and the back end. In the front end, feature points are tracked across successive images to triangulate the 3D positions. In the meanwhile, this position information is relaid to the back end for ego-motion estimation. Comparing with other SLAM technology, vSLAM is light, cheap, and applicable to platforms with a limited payload. It is a promising innovation with great potential. However, in non-textured regions and dynamic

environments, vSLAM sometimes fails due to insufficient feature correspondences. Moreover, the approximation methods in the back end still has room for improvement. To address the latter, in Chapter 3 and Chapter 4, we elaborate approximation approaches based on variational Bayesian inference in online and offline forms, respectively.

The Lidar-based SLAM uses Lidar sensors to sense the environment. One of its main advantages versus vSLAM is that it is more robust to light and thus suitable for outdoor situations. Lidar measures the distance to the object by emitting the laser light and receiving reflections. It has high frequency and can offer a 360° field of view. The measurements are in the forms of large-density point clouds, which poses a challenge for processing in real-time. Conventional methods have Iterative Closest Point (ICP) and Normal Distributions Transform (NDT), but both are sensitive to the initial misalignment and local minimum. We designed two variational Bayesian point registration algorithms and will introduce them in Chapter 5 and Chapter 6, respectively.

2.1 The SLAM Problem

Mobile vehicles are expected to perform complicated tasks in actual environments without any human intervention. To navigate autonomously, the vehicle needs to be able to identify its location within the constructed maps, which involves the technology referred to as simultaneous localization and mapping (SLAM). In a SLAM problem, mapping the spatial information of the environment is conducted online with no prior knowledge of the vehicle's location; the vehicle subsequently employs the built map to navigate. In the past decades, there has been considerable excitement about the application of the SLAM technology. Recently, along with the rise of artificial intelligence and autonomous driving cars, it has become the hot research topic once again in the mobile robotics community.

There are a large set of approaches to implement and solve SLAM. In essence, SLAM is more like a concept or system rather than a specific algorithm. For instance, SLAM can be grouped into indoor [82, 97, 155, 185] and outdoor [38, 39, 59, 134, 138]; it can also be classified as 2D [62, 154] and 3D [30, 121, 122] based on the dimension of the environmental map; according to the sensor type, we also

FIGURE 2.1: SLAM process in ORB-SLAM2 system. Graph from Stefan Hoermann, et al., “Object detection on dynamic occupancy grid maps using deep learning and automatic label generation,” in *IEEE Intelligent Vehicles Symposium (IV)*, 2018.

have visual SLAM (cameras) [38, 39, 121, 122], radar SLAM, Lidar-based SLAM [59, 134, 138] and multi-sensor SLAM [2, 109].

The above SLAM approaches are widely applied in self-driving cars, domestic robots, unmanned vehicles, and drones. In practice, vast amounts of data are collected from sensors and processed every moment. With limited onboard computational resources, the SLAM task becomes exceptionally challenging, primarily to fulfill the real-time demand. Additionally, without prior known maps of the environment, data association becomes extremely difficult when the vehicle pose is prone to uncertainty. As a consequence, though a large variety of SLAM systems have been made available, this topic has much room for further improvement.

2.1.1 Problem Definition

The objective of the SLAM problem is to construct or update an environmental map consistently and simultaneously locate the vehicle within the map. SLAM does not require any prior knowledge. The exact robot pose and map are estimated online. Figure 2.1 shows a screenshot of the ORB-SLAM2 [122] work interface as an illustration. The currently received frame is shown in the right pane, where the green rectangles are the tracked feature points. The left displays the global map, in which the red points are the reference map points. The blue and green rectangles are the retained keyframes and current frame, respectively. For every frame, ORB-SLAM2 estimates the vehicle pose (green square in the left pane) as well as the map

FIGURE 2.2: Front-end and back-end in a SLAM system.

points (red points in the left pane) simultaneously and interactively. Observations of the detected features (green rectangles in the right pane) are the projections of map points at the current state.

A SLAM architecture typically includes two main components: the front end and the back end (Figure 2.2). The front end first optimizes the sensor data into abstracted forms that can be understood the vehicle, such as features, points, and lines. Subsequently, it matches these abstractions to the existing models (landmarks, scene sets, or templates). The back end receives the processed data from the front end to perform further estimation.

We can also interpret the SLAM problem from a Bayesian perspective. Given a series of robot controls $\mathbf{u}_{1:K} = \{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_k, \dots, \mathbf{u}_K\}$ and relative observations $\mathbf{z}_{1:K} = \{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_k, \dots, \mathbf{z}_K\}$, the SLAM problem is to estimate the map $\mathbf{m}$ as well as the vehicle poses $\mathbf{x}_{1:K} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k, \dots, \mathbf{x}_K\}$. This estimation can be performed in online and offline fashion. The online estimation involves estimating the posterior $p(\mathbf{x}_k, \mathbf{m} | \mathbf{z}_{1:k}, \mathbf{u}_{1:k})$, that is, the most recent pose and map. By contrast, the offline SLAM optimizes the entire path and map $p(\mathbf{x}_{1:k}, \mathbf{m} | \mathbf{z}_{1:k}, \mathbf{u}_{1:k})$. Both online and offline SLAM are of high importance.

In practice, the online solution is usually decomposed as a recursive integration:

$$\int \int \dots \int p(\mathbf{x}_{1:k}, \mathbf{m} | \mathbf{z}_{1:k}, \mathbf{u}_{1:k}) d\mathbf{x}_1 d\mathbf{x}_2 \dots d\mathbf{x}_{k-1}, \quad (2.1)$$

which is equal to $p(\mathbf{x}_k, \mathbf{m} | \mathbf{z}_k, \mathbf{u}_k, \mathbf{x}_{k-1})$.

Applying Bayes' rule, (2.1) can be factorized as:

$$p(\mathbf{x}_k, \mathbf{m} | \mathbf{z}_k, \mathbf{u}_k, \mathbf{x}_{k-1}) \propto p(\mathbf{z}_k | \mathbf{x}_k, \mathbf{m}) p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{u}_k), \quad (2.2)$$

where $p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{m})$ is typically referred to as the “observation” model, and $p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k)$ is called as the “motion” model.

2.1.2 Sensor Types

In a SLAM system, the sensors receive environmental information and transform them into signals suitable for processing by the agent. Proper sensor selection is pivotal since it directly affects the quantity and quality of collected data provided to the agent and subsequently determines the appropriate SLAM algorithm. Typical sensors include acoustic sensors [188], laser range finders [41, 59, 105, 134, 138], and visual sensors [36, 54, 78, 83, 124].

2.1.2.1 Acoustic Sensors

Acoustic sensors, including sonar [188] and ultrasonic sensors, rely on the modulation of surface acoustic waves to detect the surrounding targets. They first transduce input electrical signal into a mechanical wave and then measure the distance by calculating the Time of Flight (ToF). Acoustic sensors are competitively priced, intrinsically reliable, and inherently rugged. However, the acoustic wave is easily affected by any changes to the characteristics of the propagation path.

In particular, sonar is mostly applied underwater, where cameras and Lidars are not applicable. The low frequency minimizes absorption but retains high resolution at the same time. Nevertheless, it is generally difficult to accurately interpret sonar sensor depth information in a subsea environment due to the high angular uncertainty caused by the monotony of subsea regions. Ultrasonic sensors (Figure 2.3a) are one of the least expensive sensors available for spatial sensing. One of their distinct advantages is the compatibility with most surface types, as long as sufficient acoustic reflectivity is ensured. However, their sensitivity to noise, slow response speed, and low resolution hampers their applicability in the context of robotics.

(c) Velodyne and camera mounted on an autonomous vehicle.

FIGURE 2.3: Ultrasonic sensor, RGB-D camera, camera and Velodyne applied in SLAM.

2.1.2.2 Laser Range Finders

Laser-based systems [59, 134, 138, 154] (Figure 2.3c) rely on laser beams to determine the object position. The sensor sends a laser pulse in a narrow beam towards the surrounding world. It then produces the reflection angle and distance, or the 3D position by measuring the time taken by the pulse to be reflected and returned to the sensor. In this process, ToF and phase-shift techniques are applied. Laser range finders are applicable to both indoor and outdoor, 2D, and 3D environments. Due to the dense data provided, laser-based systems are adept for extracting planar features (like walls). The high speed, robustness, and accuracy ensure the precision

of measurements. However, both acoustic sensors and laser range finders cannot use surface properties to identify surrounding objects. In addition, the price of the Lidar is quite high.

2.1.2.3 Visual Sensors

Visual Sensors include three types: monocular, stereo, and RGB-D. Most of them are at low cost but provide abundant information. The trade-off is a higher computational burden, as well as the requirement for more sophisticated algorithms to implement data processing in the front end. Currently, systems embedded with cameras and IMU are the main trend.

SLAM systems based on monocular cameras [45, 123, 172] (Figure 2.3c) usually have a simpler implementation, lower price, and physically smaller volume. They are accessible even on mobile phones without any auxiliary equipment. A weakness, however, is that monocular SLAM requires sophisticated algorithms since it cannot recover depth information directly from solely a 2D gray image. One solution is to initially track a known structure by integrating measurements in the live stream via multi-view [128] or filtering [30, 35]. After initializing the depth of the pixels, the shape of the map can be jointly recovered. But the scale is still unknown as the depth is not directly observable. The map, therefore, becomes dimensionless without any practical meaning in the map unit. Moreover, monocular SLAM suffers from scale drift and is likely to fail when performing a pure rotation [122]. To overcome this difficulty, occasionally, a pre-specified object [29, 47], or a set with known size [45], is employed to introduce an external scale reference for recovering the real metric scale.

To acquire direct depth, one approach is stereo vision. A stereo optical system [54, 83, 124] consists of two cameras displaced horizontally from one another so that a scene is observed from two different views. Consequently, the depth can be inferred through triangulation. However, the depth range is constrained by the baseline and resolution. In general, the wider the baseline, the more accurate depth can be inferred, though at the cost of a larger space for system setup.

Apart from stereo cameras, an alternative to provide direct depth is through RGB-D cameras [55, 114, 212] (Figure 2.3b). They infer depth information by ToF technology or structured light. ToF cameras [156] measure the ToF of the emitted

light signal to calculate the distance between camera and objects. By contrast, the structured light [55, 114, 212] employs an infrared camera to calculate the depth via the deformation of a known pattern projecting onto objects. Compared to ToF cameras, the structured cameras are more sensitive to illumination, which hampers their applicability in direct sunlight. Moreover, if the surface is semi-transparent or highly reflective, RGB-D cameras usually cannot provide reliable range data. Last but not least, the effective operating range is limited. Therefore, RGB-D cameras are typically only applicable to indoor scenarios [36, 78, 122].

2.1.3 Map Types

The conventional sensors reviewed in Section 2.1.2 transduce environmental information into signals processable for robots. For further guidance, localization, and mapping, a map is accordingly built in the sensed environment. The most commonly used map representations are low-level metric maps, high-level qualitative maps, and hybrid ones. The map quality strongly affects the performance of the robotics system [198].

2.1.3.1 Metric Map

Metric map shows a scaled display of the environment. Feature maps [121, 122], dense maps [38, 39], and occupancy grids [19] fall into this category. Feature maps (Figure 2.4a) represent the surroundings by a limited number of sparse objects, and are commonly used in SLAM systems. The computational cost is low, and map management algorithms are simple. However, feature maps are sensitive to false data association.

Feature maps utilize only a small portion of the sensor information. Conversely, dense maps [131] represent space by compact continuous surfaces. They take advantage of all the available information and track new frames leveraging on whole-image alignment. Dense maps permit a detailed semantic scene understanding and offer the potential for increasing tracking accuracy and robustness. Due to the massive amount of processed data, the computational cost significantly increases. Between feature maps and dense maps, semi-dense depth maps [38, 39] include

depths of regions in the vicinity of a sufficiently large intensity gradient. A semi-dense depth map is continuously propagated and refined from frame to frame. The poses are estimated based on the photometric error from dense image alignment.

Apart from feature maps and dense maps, the occupancy grid (Figure 2.4b) represents the scene as a discrete network of cells. Each cell is either occupied by an obstacle or empty, as tracked by a counter. The counter increases if a value is returned for the corresponding cell by the sensor, otherwise decreases. Occupancy grids are useful in path planning and exploration, where occupancy probability can computationally simplify the problem. They are commonly employed in combination with distance sensors like radar, Lidar, and sonar. The major weakness of occupancy grids is their high memory and computational complexity.

2.1.3.2 Qualitative Map

High-level qualitative maps such as topological graphs compress quantitative information into edges and points that emphasize the characteristics of the environment. They denote important places and their contiguity by a topological diagram of nodes and links. As illustrated in Figure 2.4c, the nodes are the important landmarks; the arcs connecting the nodes indicate a path for the robot to move directly. A typical example of a topological map in the real world is the city subway map. The graph structure simplifies the map management; however, it lacks an accurate geometric description of the environment and explicit representations of the obstacles.

2.1.3.3 Hybrid Map

Hybrid maps are multihierarchies of successively higher-level abstractions. They take advantage of several map representations. For instance, a hybrid metric map (HYMM) representation is introduced for dense SLAM in [131]. They firstly estimated the vehicle pose with a feature map, and then correct the local grids defined relative to the features. As a consequence, the system allows an accurate localization and simultaneously produces a detailed and globally consistent environmental map. We show an example of a hybrid grid and feature map for car localization in urban areas in Figure 2.4d.

(A) Feature map. Graph from Raul Mur-Artal, and Juan D. Tardós, "Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras," in *IEEE Transactions on Robotics*, vol. 33.5, 2017, pp. 1255-1262.

(B) Occupancy grid map. Graph from Stefan Hoermann, et al., "Object detection on dynamic occupancy grid maps using deep learning and automatic label generation," in *IEEE Intelligent Vehicles Symposium (IV)*, 2018.

(C) Topological map.

(D) Hybrid map. Graph from Jose-Luis Blanco, et al., "Toward a Unified Bayesian Approach to Hybrid Metric-Topological SLAM," in *IEEE Transactions on Robotics*, vol. 24.2, 2008, pp. 259-270.

FIGURE 2.4: Examples of feature map, occupancy grid map, topological map, and hybrid map.

2.2 Methods to Process Sensor Data

In autonomous systems, tremendous volumes of data arrive every time instantly from one or multiple diverse sensors. Efficient data processing can contribute to infrastructure improvement, as well as allowing a more efficient and sustainable estimation algorithm. In this section, we will review several different kinds of methods for data processing in visual and Lidar-based SLAM.

2.2.1 Types of Visual SLAM Processing Methods

The processing methods in visual SLAM can be grouped into feature-based vs. direct and sparse vs. dense. The former distinction describes different ways of applying the image data, and the latter describes the quantity of extracted image data.

2.2.1.1 Feature-based vs. Direct Methods

Feature-based methods [30, 121, 122] first extract features from the image data, and then match them to the existing map. The extraction computes abstractions of image information, and defines them as “features”. The resulting features range from points such as corners [22, 24, 194] to elaborate features such as edges [26, 90, 181]. Complex objects such as doorways and windows [70] are also employed in some applications. Once detected, the region around features is encoded into a compact descriptor. It can be the appearance of the feature or the intensity of the pixels in the feature patch. In computer vision, the most representative features include scale-invariant feature transform (SIFT) [103], speeded up robust features (SURF) [9], and Oriented FAST and rotated BRIEF (ORB) [121, 122]. SIFT features [103] are based on scale-space extrema detection. They construct a scale space and employ the Laplacian of Gaussian approximation for finding interesting points. As such, SIFT are not only scale-invariant, but also robust to rotation and illumination changes. The drawback is that SIFT are computationally heavy. SIFT descriptors partly inspire SURF [9] features, detected with a square-shaped filter. This filter has fast computation speed if provided with a precomputed integral image. With this advantage, SURF are detected several order faster than

SIFT features at the cost of accuracy. ORB [121, 122] were developed in 2011, as a viable and efficient alternative to the traditional SIFT and SURF. They build on the FAST keypoint detector and the BRIEF descriptor, both of which have superior performance and low cost. As a result, ORB performs as well as SIFT while being almost as fast as SURF, and hence are widely applied in the state-of-the-art SLAM systems [121, 122]. Feature matching is a process of matching the extracted feature descriptors against each other. The most common matching is utilized with a similarity measure. In an extended sequence, occasionally, changes in the appearance of the features may occur, and feature matching can handle this issue entirely. The matched features are subsequently employed to calculate the pose estimation via the Random Sample Consensus (RANSAC) [42]. In practical implementations, storing all features are incredibly costly. Therefore, people eliminate outdated data for improving the computational speed. Dense maps can be reconstructed from feature-based approaches through estimating the camera positions to find the objects at the given location.

Direct methods [38, 39] keep full images without any abstraction. Semi-dense 3D pixel maps are created in real-time via semi-dense filtering algorithms. More information is involved in reconstructing the environment, and the tracking is performed by minimizing photometric errors over all pixels. Direct approaches provide a more meaningful representation but usually cannot handle outliers quite well. Moreover, their speed is slower than feature-based variants as all the image points are involved in the iterative minimization process.

2.2.1.2 Sparse and Dense Methods

From the perspective of the areas used in acquired images, visual SLAM is classified as sparse and dense. Sparse systems select a small subset of the pixels from the received image, whereas dense process most or all of the pixels. Due to the different quantities of involved pixels and regions, their constructed maps show different patterns. Sparse methods construct point clouds, being a coarse representation of the environment and mainly used for the localization task. By contrast, dense maps provide a more comprehensive understanding of viewed scenes but require more powerful hardware because of the substantial amount of data.

2.2.2 Types of Lidar-based SLAM Processing Methods

In recent years, Lidar-based SLAM has become common. Compared to vSLAM, Lidar systems can provide high-frequency range measurements with relatively small errors in one dimension. Each transceiver in the Lidar emits pulsed light quickly and measures the reflection to calculate position and distance. Because of the fast speed and high precision, Lidar can be a fast and accurate approach to accurately track the exact distance from the vehicle to each object. At each frame, Lidar captures a full 360° 3D scan, and extract key points, belonging to edges and planes, from the acquired 3D geometry. This extraction step generates a point cloud of great size, on average more than hundreds of thousands. Two neighbouring point sets are matched to compute the relative transformation in the registration process.

However, the sensor position or the properties of the surrounding objects will strongly affect the precision of the acquired measurements. If an object is of an inconsistent shape, or one object is occluded by another at the height of the LiDAR, the information will be lost. Moreover, occlusions from scene change can also seriously decrease the registration performance. Last but not least, the large amount of the acquired points requires the registration algorithms to be computationally efficient. Currently, the most common registration approaches are Iterative Closest Point (ICP) and Normal Distributions Transform (NDT).

2.2.2.1 ICP Algorithm

Given two neighboring point sets $\mathbf{U} = \{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_M\}$ and $\mathbf{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_N\}$ in $\mathbb{R}^D$, ICP calculates the relative transformation that $\mathbf{U}$ to $\mathbf{Z}$ through minimizing a sum of the squared error:

$$\operatorname{argmin}_{R, \mathbf{t}} \frac{1}{M} \sum_{i=1}^M \|\mathbf{z}_i - R\mathbf{u}_i - \mathbf{t}\|, \quad (2.3)$$

where R is a rotation matrix, $\mathbf{t}$ denotes the translation vector, and $\mathbf{z}_i, \mathbf{u}_i$ is the i^{th} corresponding point pair.

ICP divides the entire registration into two sub-problems: the establishment of the correspondences and the optimization of (2.3). The two procedures proceed alternatively until the convergence criteria are satisfied. More specifically, for each

transformed point in $\mathbf{U}$, that is, $R\mathbf{U} + \mathbf{t}$, its correspondence is determined as its closest point in $\mathbf{Z}$, and the optimization is solved leveraging on Singular Value Decomposition (SVD).

In each iteration, with known correspondences, we firstly calculate the centers of mass of $\mathbf{U}$ and $\mathbf{Z}$ as follows:

$$\boldsymbol{\mu}_U = \frac{1}{M} \sum_{i=1}^M \mathbf{u}_i, \quad (2.4)$$

$$\boldsymbol{\mu}_Z = \frac{1}{N} \sum_{i=1}^M \mathbf{z}_i. \quad (2.5)$$

Next we subtract $\boldsymbol{\mu}_U$ and $\boldsymbol{\mu}_Z$ from $\mathbf{U}$ and $\mathbf{Z}$ respectively before calculating the transformation. The resulting point sets are:

$$\mathbf{U}^* = \{\mathbf{u}_i - \boldsymbol{\mu}_U\} = \{\mathbf{u}_i^*\}, \quad (2.6)$$

$$\mathbf{Z}^* = \{\mathbf{z}_i - \boldsymbol{\mu}_Z\} = \{\mathbf{z}_i^*\}. \quad (2.7)$$

Let $W = \sum_{i=1}^M \mathbf{z}_i^* (\mathbf{u}_i^*)^T$, then the SVD of W is denoted as:

$$W = P \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix} D^T, \quad (2.8)$$

where $P, D \in \mathbb{R}^3$ are unitary, and $\sigma_1 > \sigma_2 > \sigma_3$ are the singular values of W .

If the rank of W is 3, the optimal solution of (2.3) is unique and is given by:

$$R = PD^T \quad (2.9)$$

$$\mathbf{t} = \boldsymbol{\mu}_Z - R\boldsymbol{\mu}_U, \quad (2.10)$$

and the minimal value of (2.3) is:

$$\frac{1}{M} \sum_{i=1}^M \|(\mathbf{u}_i^*)^2 + (\mathbf{z}_i^*)^2\| - 2(\sigma_1 + \sigma_2 + \sigma_3). \quad (2.11)$$

ICP is computationally efficient. However, the point-wise correspondences are sensitive to outliers and missing partial structures. Moreover, the solution is prone

to local extrema. Furthermore, initial misalignment can significantly decrease the performance, as ICP only is effective when the two sets have sufficient overlap.

2.2.2.2 NDT Algorithm

NDT subdivides the space into cells, and establishes a grid-wise correspondence instead of an assignment between points. As such, the resulting algorithm is more robust compared to ICP.

NDT first rasterizes μ_Z into a set of cells. Each cell requires to contain at least three points. For the i^{th} cell, we collect the points containing in this cell $\{z_{i1}, z_{i2}, \dots, z_{in}\}$, and model the points inside this cell by a Gaussian distribution:

$$\boldsymbol{\mu}_i = \frac{1}{n} \sum_{j=1}^n \mathbf{z}_{ij}, \quad (2.12)$$

$$\Sigma_i = \frac{1}{n} \sum_{j=1}^n (\mathbf{z}_{ij} - \boldsymbol{\mu}_i)(\mathbf{z}_{ij} - \boldsymbol{\mu}_i)^T, \quad (2.13)$$

where $\boldsymbol{\mu}_i$ is the mean vector, and Σ_i is the covariance matrix.

The probability of measuring a random sample point $\mathbf{l}$ contained in this grid is given by:

$$p(\mathbf{l}) \sim \exp \left(-\frac{(\mathbf{l} - \boldsymbol{\mu}_i)\Sigma_i^{-1}(\mathbf{l} - \boldsymbol{\mu}_i)^T}{2} \right). \quad (2.14)$$

Assuming the transformation is described by $R, \mathbf{t}$, we transform $\mathbf{U}$ as $\mathbf{U}' = R\mathbf{U} + \mathbf{t}$. Subsequently, we map $\mathbf{U}'$ into the coordinate frame of $\mathbf{Z}$, and determine the corresponding normal distributions for each point in $\mathbf{U}'$. We define a score of R and $\mathbf{t}$ as a sum of all the probabilities:

$$\text{score}(R, \mathbf{t}) = \sum_{i=1}^M \exp \left(-\frac{(\mathbf{u}'_i - \boldsymbol{\mu}_i)\Sigma_i^{-1}(\mathbf{u}'_i - \boldsymbol{\mu}_i)^T}{2} \right), \quad (2.15)$$

where $\mathbf{u}'_i$ is a point in $\mathbf{U}'$.

The minimization problem in (2.15) is then optimized by Newton's algorithm [127, 192]. The transformation of $\mathbf{U}$ according to $R, \mathbf{t}$, and the optimization of (2.15) are performed in an iterative procedure until the convergence is met.

NDT is susceptible to the cell size. If the size is too large, the precision is decreased; if the size too small, high memory space is required. In practical implementation, the division of cells is according to a fixed size, k-means clustering, or an octree structure. In addition, NDT also requires an overlapping percentage between the two point sets to work smoothly. Last but not least, its computational cost is much higher because of the interpolation.

2.3 The Bayesian Approach to Solve SLAM

In Section 2.2, we reviewed the common data processing methods in the SLAM front-end. The acquired abstracted signals are subsequently relayed to the back end for estimating the vehicle pose and environment map jointly. This process involves a nonlinear and non-Gaussian (NLNG) inference. As stated in Section 2.1.1, the SLAM problem can be converted into a posterior estimation and solved in two forms: online and offline. Among all the proposed methodologies employed, two have been prevalent in the literature:

- **Bayesian Filtering** These methods solve the SLAM problem in the online formulation. They sequentially update posterior distributions representing the map and pose parameters $p(\mathbf{x}_k, \mathbf{m} | \mathbf{z}_k, \mathbf{u}_k, \mathbf{x}_{k-1})$. This category includes the Extended Kalman Filter (EKF; [4, 21, 62, 197]), the Rao-Blackwellized Particle Filter (i.e., FastSLAM; [116, 117]), the Unscented Kalman Filter (UKF; [61, 157]), and the Sparse Extended Information Filter approaches (SEIF; [176, 183]). A new random-finite-set approach was proposed for the feature-based SLAM with unknown number of features and uncertain data association [120].
- **Smoothing Optimization** These methods address the so-called full (offline) SLAM problem and iteratively perform batch optimization based on least-square errors. The state-of-the-art approach is the graph-based SLAM. Graph-based SLAMⁱ exploits the sparse graph structure of the SLAM problem ([31, 51, 84, 87]), in which the nodes represent vehicle poses or landmarks, and edges between two nodes encode observations that constrain the connected

ⁱGraph-based optimization is largely synonymous with “Bundle Adjustment” (BA) and “Graph SLAM”; we use the terms interchangeably.

nodes. When an online version is required, most commonly, this approach involves maintaining a sliding window of key-frames. The size of the sliding window has significant influence over the accuracy and speed of the SLAM system. When adjusting the sliding window size, an accuracy-speed trade-off is made based on the present situation and its actual demand.

In this section, we introduce the most common EKF, UKF, FastSLAM, and graph-based SLAM in detailⁱⁱ.

2.3.1 EKF Algorithm

The EKF accommodates the nonlinearities by linearizing the non-linear motion and observation models. It is the most traditional SLAM algorithm. In the past decades, it has found wide acceptance in the robotics field.

The system is described by the observation model $p(\mathbf{z}_k|\mathbf{x}_k, \mathbf{m})$ and motion model $p(\mathbf{x}_k|\mathbf{x}_{k-1}, \mathbf{u}_k)$:

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_k) + \boldsymbol{\epsilon}_k \quad (2.16a)$$

$$\mathbf{z}_k = h(\mathbf{x}_k) + \boldsymbol{\delta}_k, \quad (2.16b)$$

where f and h are two nonlinear functions, $\boldsymbol{\epsilon}_k$ and $\boldsymbol{\delta}_k$ are the additive noise following Gaussian distributions $\mathcal{N}(\mathbf{0}; Q_k)$ and $\mathcal{N}(\mathbf{0}; R_k)$ respectively, and $\mathbf{u}_k$ is the control vector.

The motion function f can be employed to predict the current state $\mathbf{x}_k$ from the previous estimate $\mathbf{x}_{k-1}$; likewise, h is used to compute the predicted observations from the predicted state. However, both f and h are nonlinear, and any Gaussian input will end up as non-Gaussian after passing through them. To resolve this issue, we resort to linear approximations of the nonlinear f and h . Both functions are linearized at the current estimate according to Taylor series. As such, after applying the approximation, the output is also a Gaussian if fed with a Gaussian.

ⁱⁱNote we exclude SEIF [176, 183], as it tends to have more in common with sliding window optimization approaches.

We denote the estimate of $\mathbf{x}$ at time m provided measurements up to and including time instant n by $\hat{\mathbf{x}}_{m|n}$. The EKF is divided into two separate steps: the prediction and correction.

The Prediction:

$$\text{Predicted mean: } \hat{\mathbf{x}}_{k|k-1} = f(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k), \quad (2.17a)$$

$$\text{Predicted covariance: } P_{k|k-1} = F_{k-1} P_{k-1|k-1} F_{k-1}^T + Q_k. \quad (2.17b)$$

The Correction:

$$\text{Innovation: } \tilde{\mathbf{y}}_k = \mathbf{z}_k - h(\hat{\mathbf{x}}_{k|k-1}), \quad (2.18a)$$

$$\text{Innovation covariance: } S_k = H_k P_{k|k-1} H_k^T + R_k, \quad (2.18b)$$

$$\text{Kalman gain: } K_k = P_{k|k-1} H_k^T S_k^{-1}, \quad (2.18c)$$

$$\text{Updated state estimate: } \hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + K_k \tilde{\mathbf{y}}_k, \quad (2.18d)$$

$$\text{Updated covariance estimate: } P_{k|k} = (I - K_k H_k) P_{k|k-1}, \quad (2.18e)$$

where F_{k-1} and H_k are the corresponding Jacobians, $F_{k-1} = \frac{\partial f}{\partial \mathbf{x}}$ and $H_k = \frac{\partial h}{\partial \mathbf{x}}$.

EKF enjoys easy implementation and can give reasonable performance when the system is mid-nonlinear. However, in general, it is not an optimal estimator since it pursues the mean of the posterior $p(\mathbf{x}_k, \mathbf{m} | \mathbf{z}_k, \mathbf{u}_k, \mathbf{x}_{k-1})$. In addition, it is sensitive to the initial estimate and prone to diverge quickly and easily in a long live stream [62]. Another problem with EKF is that it may become inconsistent easily [4, 21, 197] since it normally tends to underestimate the true covariance matrix. Last but not least, EKF has cubic computational complexity due to the inversion of the uncertainty matrix, and therefore it is inappropriate for dense feature maps.

2.3.2 UKF Algorithm

If f and h are highly nonlinear, EKF produces unsatisfactory performance (see Section 2.3.1). This is because EKF relies on the linearization of the underlying nonlinear system model to propagate the covariance matrix. To fix this issue, UKF is proposed to pick a minimal set of sample points around the mean vector by a deterministic sampling technique. Specifically, this technique is known as the

unscented transformation (UT), and those points are referred to as sigma points. In UKF, the sigma points are subsequently passed through the nonlinear system functions, from which an estimate of the mean and its associated covariance is then formed.

For the k^{th} time step, the sigma points are calculated as:

$$\chi_{k-1|k-1} = \hat{\mathbf{x}}_{k-1|k-1} \pm \sqrt{n_x + \lambda P_{k-1|k-1}}, \quad (2.19)$$

where n_x is the dimension of $\hat{\mathbf{x}}_{k-1|k-1}$, and λ is a scaling parameter.

Next, we pass χ_{k-1} through nonlinear f and h to perform the prediction step:

$$\text{Predicted mean: } \hat{\chi}_{k|k-1} = f(\hat{\chi}_{k-1|k-1}, \mathbf{u}_k), \quad (2.20a)$$

$$\hat{\mathbf{x}}_{k|k-1} = \sum_{i=0}^{2n_x} W_i^a \hat{\chi}_{i,k|k-1}, \quad (2.20b)$$

$$\text{Predicted covariance: } P_{k|k-1} = \sum_{i=0}^{2n_x} W_i^c [\hat{\chi}_{i,k|k-1} - \hat{\mathbf{x}}_{k|k-1}] [\hat{\chi}_{i,k|k-1} - \hat{\mathbf{x}}_{k|k-1}]^T + Q_k, \quad (2.20c)$$

where W_i^a is the first-order weight of the i^{th} sigma point, and W_i^c is the second-order weight.

The update step is then performed as:

$$\text{Predicted measurement: } \mathbf{z}_{k|k-1} = h(\hat{\mathbf{x}}_{k|k-1}), \quad (2.21a)$$

$$\hat{\mathbf{z}}_{k|k-1} = \sum_{i=0}^{2n_x} W_i^a \mathbf{z}_{i,k|k-1}, \quad (2.21b)$$

$$\begin{aligned} \text{Innovation covariance: } S_k &= \sum_{i=0}^{2n_x} W_i^c [\mathbf{z}_{i,k|k-1} - \hat{\mathbf{z}}_{k|k-1}] [\mathbf{z}_{i,k|k-1} - \hat{\mathbf{z}}_{k|k-1}]^T \\ &\quad + R_k, \end{aligned} \quad (2.21c)$$

$$\text{Cross covariance: } C_{\chi z} = \sum_{i=0}^{2n_x} W_i^c [\mathbf{z}_{i,k|k-1} - \hat{\mathbf{z}}_{k|k-1}] [\hat{\chi}_{i,k|k-1} - \hat{\mathbf{x}}_{k|k-1}]^T, \quad (2.21d)$$

$$\text{Kalman gain: } K_k = C_{\chi z} S_k^{-1}, \quad (2.21e)$$

$$\text{Updated state estimate: } \hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + K_k (\mathbf{z}_k - \hat{\mathbf{z}}_{k|k-1}), \quad (2.21f)$$

$$\text{Updated covariance estimate: } P_{k|k} = P_{k|k-1} - K_k S_k K_k^T, \quad (2.21g)$$

where W_i^a and W_i^c are the first-order and second-order weights of the original sigma points separately.

Through the sigma points, UKF avoids direct linearization, and instead depends on calculating the transformed statistics of the UT and the chosen sigma point set. Compared with the EKF, UKF more accurately estimates the posterior. Moreover, the UT technique successfully removes the explicit calculation of Jacobians, which may be difficult and costly. However, the computational cost of UKF also increases accordingly due to the sampling of sigma points. Apart from this, its computational complexity remains to be cubic. As a consequence, despite the proven feasibility, it is not a popular choice in practical applications.

2.3.3 FastSLAM Algorithm

FastSLAM decomposes the online SLAM problem into a vehicle localization problem, and a collection of conditioned landmark estimation problems [116, 117]. We assume that the map $\mathbf{m}$ consists of landmarks $\mathbf{l} := [\mathbf{l}_1, \dots, \mathbf{l}_i, \dots, \mathbf{l}_N]$. The SLAM problem is then expressed as a factored representation:

$$p(x_{1:k}, \mathbf{l} | \mathbf{z}_{1:k}, \mathbf{u}_{1:k}) = p(\mathbf{x}_{1:k} | \mathbf{z}_{1:k}, \mathbf{u}_{1:k}) p(\mathbf{l} | \mathbf{z}_{1:k}, \mathbf{x}_{1:k}). \quad (2.22)$$

For the posterior over robot path $p(\mathbf{x}_{1:k} | \mathbf{z}_{1:k}, \mathbf{u}_{1:k})$, we maintain a set of particles $S_{1:k}$ to approximate its distribution. Each particle $\mathbf{s}_{1:k}^{[m]} \in S_{1:k}$ represents a possible solution of the robot's path,

$$S_{1:k} = \{\mathbf{s}_{1:k}^{[m]}\}_m = \{\mathbf{s}_1^{[m]}, \mathbf{s}_2^{[m]}, \dots, \mathbf{s}_k^{[m]}\}, \quad (2.23)$$

where the superscript is referred to the m^{th} particle.

We calculate the particle set $S_{1:k}$ incrementally, from the set $S_{1:k-1}$. First, each particle $\mathbf{s}_k^{[m]}$ is generated as:

$$\mathbf{s}_k^{[m]} \approx p(\mathbf{x}_k | \mathbf{u}_k, \mathbf{s}_{k-1}^{[m]}). \quad (2.24)$$

Assuming $S_{1:K-1}$ is distributed according to $p(\mathbf{x}_{1:k-1}|\mathbf{z}_{1:k-1}, \mathbf{u}_{1:k-1})$, the new particle follows the distribution $p(\mathbf{x}_{1:k}|\mathbf{z}_{1:k-1}, \mathbf{u}_{1:k})$, which is known as the proposal distribution.

We set the number of particles as M , $s_{1:k}^{[m]}$ is drawn with an associating weight defined as:

$$w_k^{[m]} = \frac{p(s_{1:k}^{[m]}|\mathbf{z}_{1:k}, \mathbf{u}_{1:k})}{p(s_{1:k}^{[m]}|\mathbf{z}_{1:K-1}, \mathbf{u}_{1:k})}, \quad (2.25)$$

which accounts for the differences between the proposal and the target.

On the other hand, the conditional landmark estimate $p(\mathbf{l}|\mathbf{z}_{1:k}, \mathbf{x}_{1:k})$ is represented by Kalman Filters. Firstly, we factorize

$$p(\mathbf{l}|\mathbf{z}_{1:k}, \mathbf{x}_{1:k}) = \prod_{i=1}^N p(\mathbf{l}_i|\mathbf{z}_{1:k}, \mathbf{x}_{1:k}), \quad (2.26)$$

where each $p(\mathbf{l}_i|\mathbf{z}_{1:k}, \mathbf{x}_{1:k})$ is approximated via an EKF.

Explicitly, the weight $w_k^{[m]}$ is calculated as follows:

$$w_k^{[m]} = \frac{1}{|2\pi Q|} \exp\left(-\frac{1}{2}(\mathbf{z}_k - \mathbf{z}_k^{[m]})^T Q^{-1}(\mathbf{z}_k - \mathbf{z}_k^{[m]})\right), \quad (2.27)$$

where Q is the measurement covariance matrix, and $\mathbf{z}_k^{[m]}$ is the predicted measurement at time instant k .

After the above procedures, we resample M particles with replacement, in which each particle is generated using a probability proportional to $w_k^{[m]}$.

The resulting algorithm is a Rao-Blackwellized non-parametric, recursive particle filter, and its complexity is $\mathcal{O}(MN \log N)$, where N is the number of all the landmarks. Despite a linear complexity w.r.t. the number of features, FastSLAM is generally only applicable to low-dimensional spaces due to the large number of particles M .

2.3.4 Graph-based SLAM

In online SLAM, data is processed and then discarded, making it impossible to revisit all data when building a map. By contrast, offline techniques offer improved

accuracy by memorizing all data and postponing the mapping process until the end. Among all the offline algorithms, the state-of-the-art method is graph-based SLAM (Figure 2.5). It addresses the offline SLAM problem via a graph-based formulation, where every node corresponds to a vehicle pose or a landmark, and every edge between two nodes represents a spatial constraint acquired from the observations. The constraints are generally nonlinear, and this graph then naturally transforms the SLAM problem into a sum of nonlinear quadratic constraints minimization. Optimizing these constraints yields a node configuration that is maximally consistent with the measurements. The produced output is a maximum likelihood map and its corresponding set of vehicle states.

FIGURE 2.5: A pose-graph representation of a SLAM process.

Assuming the vehicle is moving in an unknown environment, $(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_i, \dots, \mathbf{x}_P)$ is the vector of unknown parameters (including vehicle trajectory and global map), where $\mathbf{x}_i$ be the i^{th} vertex. Let $\mathbf{z}_{ij}$ and Ω_{ij} respectively be the measurement and information matrix that constrains node $\mathbf{x}_i$ and node $\mathbf{x}_j$. Let $\hat{\mathbf{z}}_{ij}$ be the predicted measurement, and then we can define an error function as follows:

$$\mathbf{e}_{ij}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{z}_{ij} - \hat{\mathbf{z}}_{ij}(\mathbf{x}_i, \mathbf{x}_j), \quad (2.28)$$

which computes the difference between the real and expected measurement.

The log-likelihood l_{ij} of a measurement $\mathbf{z}_{ij}$ is:

$$l_{ij} \propto \mathbf{e}_{ij}^T \Omega_{ij} \mathbf{e}_{ij}. \quad (2.29)$$

We denote the set of pairs of indices for which a constraint exists as $\mathcal{C}$. Essentially, maximizing a likelihood amounts to minimize the negative log-likelihood $F(x)$ of

all the measurements:

$$\begin{aligned} F(\mathbf{x}) &= \sum_{\langle i,j \rangle \in \mathcal{C}} \mathbf{e}_{ij}^T \Omega_{ij} \mathbf{e}_{ij} \\ &= \sum_{\langle i,j \rangle \in \mathcal{C}} F_{ij}. \end{aligned} \quad (2.30)$$

Equivalently, it seeks to find a configuration of the nodes $\hat{\mathbf{x}}^*$ to solve the following equation:

$$\hat{\mathbf{x}}^* = \arg \min_x F(x). \quad (2.31)$$

If a good initial solution $\check{\mathbf{x}}$ is known, we can employ the popular Gauss-Newton or Levenberg-Marquardt methods to solve (2.31). The idea is to approximate $\mathbf{e}_{ij}$ via the first order Taylor expansion around the current guess $\check{\mathbf{x}}$:

$$\mathbf{e}_{ij}(\check{\mathbf{x}}_i + \Delta \mathbf{x}_i, \check{\mathbf{x}}_j + \Delta \mathbf{x}_j) = \mathbf{e}_{ij}(\check{\mathbf{x}} + \Delta \mathbf{x}), \quad (2.32a)$$

$$\simeq \mathbf{e}_{ij} + J_{ij} \Delta \mathbf{x}, \quad (2.32b)$$

where $\mathbf{e}_{ij} = \mathbf{e}_{ij}(\check{\mathbf{x}})$, and J_{ij} is the Jacobian of $\mathbf{e}_{ij}(\mathbf{x})$ at $\check{\mathbf{x}}$.

Substituting (2.32) into F_{ij} in (2.30), we rewrite F_{ij} as:

$$F_{ij}(\check{\mathbf{x}} + \Delta \mathbf{x}) = c_{ij} + 2\mathbf{b}_{ij} \Delta \mathbf{x} + \Delta \mathbf{x}^T H_{ij} \Delta \mathbf{x}, \quad (2.33a)$$

$$\simeq c_{ij} + 2\mathbf{b}_{ij} \Delta \mathbf{x} + \Delta \mathbf{x}^T H_{ij} \Delta \mathbf{x} F(\check{\mathbf{x}} + \Delta \mathbf{x}), \quad (2.33b)$$

where $J_{ij} = \frac{\partial \mathbf{e}_{ij}}{\partial \mathbf{x}}|_{\mathbf{x}=\check{\mathbf{x}}}$, $c_{ij} = \mathbf{e}_{ij}^T \Omega_{ij} \mathbf{e}_{ij}$, $\mathbf{b}_{ij} = \mathbf{e}_{ij}^T \Omega_{ij} J_{ij}$ and $H_{ij} = J_{ij}^T \Omega_{ij} J_{ij}$.

With the local approximation, we then obtain:

$$F(\check{\mathbf{x}} + \Delta \mathbf{x}) = \sum_{\langle i,j \rangle \in \mathcal{C}} F_{ij}(\check{\mathbf{x}} + \Delta \mathbf{x}), \quad (2.34)$$

$$\simeq c + 2\mathbf{b} \Delta \mathbf{x} + \Delta \mathbf{x}^T H \Delta \mathbf{x}, \quad (2.35)$$

where $c = \sum c_{ij}$, $\mathbf{b} = \sum \mathbf{b}_{ij}$, and $H = \sum H_{ij}$.

Finally (2.35) can be minimized via solving the below linear equation:

$$H \Delta \mathbf{x}^* = -\mathbf{b}, \quad (2.36)$$

where H denotes the information matrix of the system.

Since H is sparse, generally we can solve the above equation in (2.36) by sparse Cholesky factorization efficiently and effortlessly. Finally, the linearized solution is

$$\hat{\mathbf{x}}^* = \tilde{\mathbf{x}} + \Delta \mathbf{x}^*. \quad (2.37)$$

The algorithm iterates the linearization in (2.35), the solution in (2.36), and the update in (2.37). In every cycle, the previous $\hat{\mathbf{x}}^*$ is served as the new initial guess and linearization point.

In summary, graph-based SLAM transforms the offline SLAM problem into a graphical network, representing the log-likelihood of the data. The graph structure can be simplified via variable elimination techniques and then solved by least-square error minimization. As a result, graph-based SLAM can handle a large number of features. However, it is also characterized by several limitations. One arises from the fixed variance of the Gaussian noise (fixed Ω_{ij} in the optimization). Clearly, the variance of real-world noise is unknown and, more importantly, the prior may be inaccurate. Another limitation pertains to the painfully slow matrix inversion in (2.36), which has cubic complexity. Moreover, graph-based SLAM is sensitive to the initialization of the state. As time progresses, the accuracy of the odometry-based initial estimate will degrade, leading to increased data association errors. Graph-based SLAM may subsequently diverge.

2.4 Variational Bayesian Inference

We reviewed the common filtering and smoothing methods in Section 2.3, respectively, for solving online and offline SLAM problems. Despite the wide applicability, both approaches still have some drawbacks. Filtering techniques have high computational complexity due to the propagation of uncertainty; for large maps, they become increasingly slow. Graph optimization methods can handle a large number of features, but they are still computationally demanding if the number of frames is vast. To overcome those shortcomings, in this thesis, we propose variational Bayesian inference methods to apply in the SLAM field. Compared with filtering and graph-based SLAM, variational Bayesian inference methods have lower complexity and better scalability. Moreover, it provides simple iterative solutions, with

guaranteed local convergence. It can also avoid singularities and better capture local signal properties.

2.4.1 Bayesian Inference Basics

A central task in the application of modern machine learning and probabilistic models is the evaluation of the posterior distribution of the latent variables given the observed data. Especially, large-scale inference in complex models is in urgent need to be explored [12].

Suppose that we have a complex model at hand, which has observed data denoted by $\mathbf{X}$ and latent variables $\mathbf{Z}$. The posterior distribution $p(\mathbf{Z}|\mathbf{X})$ is unknown and to be estimated.

A potential solution is to resort to Monte Carlo sampling methods, such as Gibbs sampling. However, standard Monte Carlo sampling does not scale computationally well with the number of samples, and the approximation is poor without a large number of samples [12, 44]. As a result, it is limited to small-scale cases. In addition, for complex models, assessing convergence to the stationary posterior distribution is notoriously difficult [12, 161].

As an alternative to MCMC methods, variational Bayesian inference (VBI) [12, 85, 141, 177, 189, 201] are a family of techniques to approximate intractable posterior distributions arising in machine learning and Bayesian inference. Primarily, they can provide an analytical approximation to the posterior probability $p(\mathbf{Z}|\mathbf{X})$. Unlike Monte Carlo sampling, VBI seeks another distribution $q(\mathbf{Z})$ with an easier form, to approximate $p(\mathbf{Z}|\mathbf{X})$.

Firstly we define the log evidence of $q(\mathbf{Z})$ and the KL-divergence of $p(\mathbf{Z}|\mathbf{X})$ from $q(\mathbf{Z})$ as follows:

$$\mathcal{L}(q) = \int q(\mathbf{Z}) \log \left\{ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right\} d\mathbf{Z} \quad (2.38)$$

$$\mathbf{KL}(q||p) = - \int q(\mathbf{Z}) \log \left\{ \frac{p(\mathbf{Z}|\mathbf{X})}{q(\mathbf{Z})} \right\} d\mathbf{Z}, \quad (2.39)$$

where $p(\mathbf{X}, \mathbf{Z})$ is the joint distribution. As $p(\mathbf{X}, \mathbf{Z}) = p(\mathbf{X}|\mathbf{Z})p(\mathbf{Z})$, it follows $\log p(\mathbf{X}|\mathbf{Z}) = \log p(\mathbf{X}, \mathbf{Z}) - \log p(\mathbf{Z})$. As such, the log marginal probability $\log p(\mathbf{X})$

is decomposed as:

$$\log p(\mathbf{X}) = \mathcal{L}(q) + \mathbf{KL}(q||p). \quad (2.40)$$

Our goal is to seek an approximation that satisfies $q(\mathbf{Z}) \approx p(\mathbf{Z}|\mathbf{X})$. This leads to maximize the lower bound $\mathcal{L}(q)$, which essentially amounts to minimizing the KL divergence, as indicated in (2.40). In a sense, the KL divergence usually can be considered as the “closeness” of q from p . Theoretically, the minimization occurs when $q(\mathbf{Z})$ equals to $p(\mathbf{X}|\mathbf{Z})$ with the premise that $q(\mathbf{Z})$ can be any possible choices.

However, generally, we need to limit $q(\mathbf{Z})$ to a family of distributions and seek the most appropriate member to minimize the KL divergence. The restriction is imposed purely to make the $q(\mathbf{Z})$ tractable, while sufficiently flexible and rich to approximate the true posterior distribution well.

One way to impose such restriction is to use the dependence of $q(\mathbf{Z})$ on ω , where ω is subject to a set of parameters. As such, $q(\mathbf{Z})$ is converted into a parameter distribution $q(\mathbf{Z}|\omega)$ and the lower bound $\mathcal{L}(q)$ has become a function of ω . According to the practical application, suitable approaches are employed to determine the optimal ω .

2.4.2 Mean field approximation

To identify a $q(\mathbf{Z})$ that maximizes the lower bound $\mathcal{L}(q)$, is equivalently to minimize the KL divergence $\mathbf{KL}(q||p)$ (see Section 2.4.1). This process is subject to the normalization constraints:

$$\forall i \int q_i(\mathbf{Z}_i) d\mathbf{Z}_i = 1. \quad (2.41)$$

First we elaborate (2.38) as:

$$\begin{aligned} \mathcal{L}(q) &= \int q(\mathbf{Z}) \log \left\{ \frac{p(\mathbf{X}, \mathbf{Z})}{q(\mathbf{Z})} \right\} d\mathbf{Z}, \\ &= \int q(\mathbf{Z}) \log p(\mathbf{X}, \mathbf{Z}) d\mathbf{Z} - \int q(\mathbf{Z}) \log q(\mathbf{Z}) d\mathbf{Z}, \\ &= -E[\mathbf{X}, \mathbf{Z}]_{q(\mathbf{Z})} + H[q(\mathbf{Z})], \end{aligned} \quad (2.42)$$

where the entropy $H(q(\mathbf{Z})) = -\int q(\mathbf{Z}) \log q(\mathbf{Z}) d\mathbf{Z}$ and the energy $E[\mathbf{X}, \mathbf{Z}]_{q(\mathbf{Z})} = -\log p(\mathbf{X}, \mathbf{Z})$.

For exponential models, the function $q(\mathbf{Z})$ is factorized as a product of $q_i(\mathbf{X}_i)$ under the mean field assumption:

$$\mathcal{L}(q) = \int -E[\mathbf{X}, \mathbf{Z}] \left(\prod_i q_i(\mathbf{Z}_i) \right) d\mathbf{Z} - \int \left(\prod_i q_i(\mathbf{X}_i) \right) \sum_i \log(q_i(\mathbf{Z}_i)) d\mathbf{Z}. \quad (2.43)$$

If we bring out the sum, the right term will become as the following:

$$\sum_i \int \left(\prod_i q_i(\mathbf{X}_i) \right) \log(q_i(\mathbf{Z}_i)) d\mathbf{Z}. \quad (2.44)$$

Meanwhile we partition the whole $\mathbf{Z}$ set into two parts $\{\mathbf{Z}_i, \bar{\mathbf{Z}}_i\}$, where $\bar{\mathbf{Z}}_i = \mathbf{Z} \setminus \mathbf{Z}_i$. Correspondingly (2.44) can be rewritten as:

$$\begin{aligned} & \sum_i \int q_i(\mathbf{Z}_i) q_i(\bar{\mathbf{Z}}_i) \log(q_i(\mathbf{Z}_i)) d\mathbf{Z} d\bar{\mathbf{Z}}_i \\ &= \sum_i \langle \log q_i(\mathbf{Z}_i) q_i(\mathbf{Z}_i) d\mathbf{Z}_i \rangle_{q_i(\bar{\mathbf{Z}}_i)} \\ &= \sum_i \int q_i(\mathbf{Z}_i) \ln q_i(\mathbf{Z}_i) d\mathbf{Z}_i. \end{aligned} \quad (2.45)$$

By substituting (2.45) into (2.43), we obtain:

$$\mathcal{L}(q) = \int -E[\mathbf{X}, \mathbf{Z}] \left(\prod_i q_i(\mathbf{Z}_i) \right) d\mathbf{Z} - \sum_i \int q_i(\mathbf{Z}_i) \log q_i(\mathbf{Z}_i) d\mathbf{Z}_i. \quad (2.46)$$

For the left term in (2.46), we partition $\mathbf{Z}$ set into two parts again:

$$\begin{aligned} & \int q_i(\mathbf{Z}_i) d\mathbf{Z}_i \int q_i(\bar{\mathbf{Z}}_i) (-E[\mathbf{X}, \mathbf{Z}]) d\bar{\mathbf{Z}}_i, \\ &= \int q_i(\mathbf{Z}_i) (-E[\mathbf{X}, \mathbf{Z}]_{q_i(\bar{\mathbf{Z}}_i)}) d\bar{\mathbf{Z}}_i \\ &= \int q_i(\mathbf{Z}_i) \log \exp \langle -E[\mathbf{X}, \mathbf{Z}] \rangle_{q_i(\bar{\mathbf{Z}}_i)} d\mathbf{Z}_i, \\ &= \int q_i(\mathbf{Z}_i) \log q_i^*(\mathbf{Z}_i) d\mathbf{Z}_i + \log \mathbf{Z}, \end{aligned} \quad (2.47)$$

where $q_i^*(\mathbf{Z}_i) = \frac{1}{\mathbf{Z}} \exp \langle -E(\mathbf{X}, \mathbf{Z}) \rangle_{q_i(\bar{\mathbf{Z}}_i)}$ and $\mathbf{Z}$ is the normalization factor of $q_i^*(\mathbf{Z}_i)$.

By substituting 2.47 into 2.46, it then yields:

$$\mathcal{L}(q) = \int q_i(\mathbf{Z}_i) \log q_i^*(\mathbf{Z}_i) d\mathbf{Z}_i - \sum_i \int q_i(\mathbf{Z}_i) \log q_i(\mathbf{Z}_i) d\mathbf{Z}_i + \log \mathbf{Z}. \quad (2.48)$$

Finally we marginalize the $H_i = H[q(\mathbf{Z}_i)]$ out from the entropy sum:

$$\mathcal{L}(q) = \left\{ \int q_i(\mathbf{Z}_i) \log q_i^*(\mathbf{Z}_i) d\mathbf{Z}_i - \int q_i(\mathbf{Z}_i) \log q_i(\mathbf{Z}_i) d\mathbf{Z}_i \right\} + H[q(\bar{\mathbf{Z}}_i)] + \log \mathbf{Z}. \quad (2.49)$$

We can rewrite the term in brackets as:

$$\begin{aligned} & \int q_i(\mathbf{Z}_i) \log q_i^*(\mathbf{Z}_i) d\mathbf{Z}_i - \int q_i(\mathbf{Z}_i) \log q_i(\mathbf{Z}_i) d\mathbf{Z}_i \\ &= \int q_i(\mathbf{Z}_i) \log \frac{q_i^*(\mathbf{Z}_i)}{q_i(\mathbf{Z}_i)} d\mathbf{Z}_i, \\ &= -\mathbf{KL}(q_i(\mathbf{Z}_i) || q_i^*(\mathbf{Z}_i)) \end{aligned} \quad (2.50)$$

As a result, minimizing the KL divergence between large joint distributions q and p is now transformed into minimizing the KL divergence between individual 1D distributions:

$$\mathcal{L}(q) = -\mathbf{KL}(q_i(\mathbf{Z}_i) || q_i^*(\mathbf{Z}_i)) + H[q(\bar{\mathbf{Z}}_i)] + \log \mathbf{Z}, \quad (2.51)$$

where the $\mathcal{L}(q)$ depends on each q_i through the KL divergence term.

From (2.51), we can see that the maximization of $\mathcal{L}(q)$ is achieved when $q_i(\mathbf{Z}_i) = q_i^*(\mathbf{Z}_i)$. Since $q_i^*(\mathbf{Z}_i) = \frac{1}{\mathbf{Z}} \exp \langle -E(\mathbf{X}, \mathbf{Z}) \rangle_{q_i(\bar{\mathbf{Z}}_i)}$ in (2.47), therefore the optimal value of $q_i(\mathbf{Z}_i)$ is:

$$q_i(\mathbf{Z}_i) = \frac{1}{\mathbf{Z}} \exp \langle -E(\mathbf{X}, \mathbf{Z}_i, \bar{\mathbf{Z}}_i) \rangle_{q_i(\bar{\mathbf{Z}}_i)}, \quad (2.52)$$

where $E(\mathbf{X}, \mathbf{Z}_i, \bar{\mathbf{Z}}_i) = -\log p(\mathbf{X}, \mathbf{Z}_i, \bar{\mathbf{Z}}_i)$ is the energy.

At last, we can write (2.52) as the following update:

$$q_i(\mathbf{Z}_i) \leftarrow \frac{1}{\mathbf{Z}} \exp \langle -E(\mathbf{X}, \mathbf{Z}_i, \bar{\mathbf{Z}}_i) \rangle_{q_i(\bar{\mathbf{Z}}_i)}. \quad (2.53)$$

This update process optimizes each node $\mathbf{Z}_i$ at a time to make it converge to a local minimum [12, 203].

2.5 The Future of SLAM: Traditional Approaches vs. Deep Learning

In previous sections, we posed SLAM as a purely geometric problem and discussed the conventional approaches for feature engineering and pose estimation. These traditional methods have reached a level of maturity where maps can be constructed at a real-time speed [38, 39, 121, 122]. However, the lack of semantics can cause inefficiency in ill-posed regions and dynamic environments. Unfortunately, in the traditional field, when it comes to semantic information, the SLAM community gets stuck in the bag-of-the-words model [121, 122]. It has no novel strategies for better integration. On the other hand, real-time semantic segmentation [46, 63] arising from the computer vision field have not only achieved great results, but their applications in single/two/multi-view SFM (Structure from Motion) and ego-motion estimation have shown high potential as well. This success has partially shifted the focus of SLAM away from the traditional and towards the learning-based models. At present, end-to-end trainable deep learning frameworks have been slowly creeping up on quite a few advanced SLAM topics (e.g., dynamic SLAM, semantic SLAM). The intersection of SLAM and deep learning has also begun to attract attention.

In this section, we will first introduce the deep learning concept, and then review the learning-based methods applied in the SLAM community. The pros & cons will be compared at the end of this section, and the future trend will be discussed briefly.

2.5.1 Dee Learning: Definition

Deep learning is a stimulating subfield of machine learning. Inspired by the structure and function of biological systems, deep learning processes data through sequential networks and then create patterns for future decision. Given a large volume of data, deep learning is usually considered to be one of the best solutions for discovering complex architectures hidden in the input. Currently, deep learning has achieved improved performance in numerous applications, such as computer vision [102], object detection [99], speech recognition [3], and other data analysis problems [205].

FIGURE 2.6: Deep learning architecture.

Each deep learning network involves multiple sequential layers (Figure 2.6), like the human brains being composed of numerous computing neurons. Each layer is a nonlinear processing unit through which the data is transformed. Typically, these layers can be categorized into three groups: the input layer, the hidden layers, and the output layer. Explicitly, the data is presented visibly at the input layer; then, a series of hidden layers extract features increasingly; the output of the model is finally represented from the output layer. Since numerous layers are involved, this architecture is thus described as “deep”.

FIGURE 2.7: Basic calculation in a neuron.

As shown in Figure 2.6, each layer composes of multiple perceptrons, which are the fundamental building blocks of deep learning. The transformation process in perceptrons is illustrated as Figure 2.7. Concretely, given a finite set of inputs (e.g., $x_1, x_2, \dots, x_n$), we multiply each input by an associating weight (e.g., $w_1, w_2, \dots, w_n$), and then compute the weighted sum $\sum_{i=1}^n w_i x_i$, next add a bias b to the sum and finally pass the combination $\sum_{i=1}^n w_i x_i + b$ through a nonlinear activation function g . This process outputs the prediction $g(\sum_{i=1}^n w_i x_i + b)$, which will input to perceptrons in the subsequent layer. The determination of weights and biases relies on the “learning” phase, which can be further divided into training and testing. The former employs data to estimate the weights and biases through backpropagation, while the latter tests the inferred phase on new unexposed data. According to the level of availability of ground truth in the training data, learning has three types: supervised, semi-supervised, and unsupervised. Supervised learning aims to learn a pattern given a sample of data and desired outputs. Unsupervised learning draws inferences from unlabelled data. Between the two, semi-supervised learning trains the system with the combination of a small set of labeled data and a broad set of unexposed data.

Generally, according to the literature, deep learning shows outperformed performance when the size of data is large. In particular, if there lacks domain knowledge about feature introspection, deep learning is more appropriate as it solves problems on an end-to-end basis with no need to worry about feature engineering. However, the large size of data also brings challenges. First of all, to train deep learning networks in a reasonable time, high-end infrastructure is required. In addition, the

labeling of massive ground truth sometimes is tedious or impractical. As a result, in light of different targets, people designed different networks to address hardware efficiency, data efficiency, or model efficiency accordingly. The deep learning methods for SLAM are mostly based on Convolutional Neural Network (CNN) [206].

2.5.2 Learning-based SLAM

Traditional SLAM methods are mostly based on multi-view geometry. They minimize geometric reprojection errors and work smoothly in scenes where most objects are static. Despite the advancements, much of the success has a heavy reliance on high-quality low-level feature correspondences. In non-textured regions, dynamic surroundings, and abrupt motions, the proneness to outliers is very likely to cause inevitable failure. This issue can be tackled by introducing semantic representations, which can facilitate place recognition by providing conceptual knowledge of the surroundings. However, currently, traditional SLAM has no means of perfectly associating semantic concepts with geometry entities. On the other hand, as a data-driven technique, deep learning has produced excellent results in numerous computer vision tasks, such as classification and matching. With the aid of the big data, learning-based frameworks can successfully capture high-level semantics with no explicit feature engineering. Take data association as an example, as depicted in Figure 2.8, the traditional methodology first extracts hand-crafted features and then build correspondences manually; however, learning architectures take the raw data as input and automatically determine the most representative abstractions. The workflow of learning models has dramatically changed the common knowledge and expertise in the traditional field. The multiple levels of representation at a more abstract level motivate researchers to begin exploring augmenting SLAM with deep learning.

Regarding deep learning techniques for SLAM, one type of learning framework that demands specific mention is convolutional neural networks (CNNs). The term “convolutional” enumerates the concept of “convolution”, which is a specialized linear operation. Unlike traditional networks where layers are fully connected, convolutional layers convolve the input with multiplication or other dot product, that is, apply the same weights to the whole input space. The shared weights

FIGURE 2.8: Workflows of traditional and deep learning-based data association.

make a significant breakthrough as a reduction of the overall number of variables can greatly facilitate the processing speed. Currently, CNNs have become one of the most effective learning techniques for spatial information processing, such as scene recognition, semantic segmentation, and object tracking. Without exception, convolutional layers are also specifically vital for SLAM problems. A large portion of learning-based SLAM schemes are built on the basis of CNNs, such as [48, 77, 110, 191].

Explicitly, existing deep learning enhanced SLAM models can be divided into three groups: fully end-to-end learning approaches for single/two/multiple views; learnable drop-in replacements for components in the traditional framework; the integration of auxiliary modules.

Part of recent studies substitute the traditional framework with a learning-based SLAM solution, estimating depth and ego-motion from single view [96, 108, 210] or visual navigation fully by neural networks [53, 211]. However, these deep models are all heavily dependent on the training data and thus have a large probability of failing in unknown surroundings where there exists a huge difference from the training scenes. Consequently, they can hardly achieve gains against traditional frameworks in practice. To address these limitations, some researchers only applied deep learning to low-level subproblems, keeping other components of the traditional SLAM framework unchanged. Such attempts include single view depth estimation [48], camera relocalization [77, 191], stereo visual odometry [196], and image geolocalization [182]. Despite the promising results demonstrated in certain scenarios, when it comes to the comparison against traditional approaches, those learning algorithms do not always produce superior outputs, especially for

the depth estimation task. This is attributable to the reliance on the priors drawn from the training data, and the replacement approaches are still subject to the same limitation of end-to-end solutions. Instead of replacement, some work integrated auxiliary learning-based modules into the geometric SLAM system so that an improvement can be expected. McCormac [110] has combined CNNs with a geometric SLAM system, where semantic predictions from CNNs are probabilistically fused into a geometric map. This study has achieved a higher accuracy. However, it does not consider the optimization. The direct sparse visual odometry proposed in [37] jointly minimizes the photometric error and camera parameters. This minimization considers a full photometric calibration, accounting for enhanced robustness in dynamic environments. Both [110] and [37] are loose fusion. To incorporate high-level abstractions into general SLAM pipelines much tighter, the SLAM++ model proposed in [152] optimizes the geometric constraints and semantic scene information jointly, extracting the semantics at the level of objects and combining them with the constraints of Bundle Adjustment. However, the inclusion of semantic information leads to an increased number of variables, which poses a challenge for computing power. In addition, no consensus has been reached regarding suitable loss functions for semantic information.

Overall, although promising results have been produced, deep learning enhanced SLAM systems are still in an embryonic stage. More efforts need to be done in this direction for a more mature solution. Nevertheless, deep learning is still one of the most promising approaches to handle the issues of outliers and non-textured regions in the SLAM problems.

2.5.3 The Comparison & Future Trends

The existing issues such as outliers, non-textured regions can hardly be overcome without incorporating high-level semantics (see Section 2.5.2). In many domains, people have begun to embrace learning techniques and enjoy its effectiveness and efficiency. This poses an open question: will deep learning, especially end-to-end learning, dominate in the SLAM community in the near future?

Before answering this question, we firstly draw a table to make a comparison (Table 2.1). Learning-based approaches can well overcome the limitations of traditional SLAM techniques when applying in dynamic environments and ill-posed or

non-textured regions. However, this effectiveness requires a large amount of data. Thus learning-based methods usually are highly dependent on the given data and sometimes cannot well adapt to new environments. In this respect, traditional frameworks are more competitive due to universality. Moreover, deep models require high computing power and training time, which will be costly. The evaluation of learning frameworks is also an unsettled issue at present. Therefore, we believe that deep learning will not entirely replace traditional architectures for some time. Instead, the marriage of deep learning and traditional techniques will be the most probable future trend. We should remind ourselves it does not mean we should use deep learning just because we can.

2.6 Conclusion

The two most representative SLAM platforms, vSLAM, and Lidar-based SLAM, have been addressed in much research work. First, we introduced the basic SLAM definition, the most common sensors, and maps in Section 2.1. A traditional SLAM system is composed of two parts, namely, the front end and the back end. The front end involves data processing (see Section 2.2), while the back end employs Bayesian approaches (see Section 2.3) to optimize the vehicle pose and environmental map. In Section 2.2 and Section 2.3, we listed the achievements as well as the challenges. We introduced variational Bayesian inference as a promising solution for performance improvement in Section 2.4.1. The details of the derived algorithms will be elaborated in the following chapters. As deep learning has become a trend in quite a few domains, we discussed the recent applications for SLAM in Section 2.5.

TABLE 2.1: Advantages & disadvantages of learning-based and traditional SLAM.

Approaches	Advantages	Disadvantages
Traditional	<ul style="list-style-type: none"> • A better solution if the static features are in the majority • No requirement of specific class knowledge • Full transparency, easy to judge and evaluate • Simpler, faster, and superior flexibility • Can be deployed on low cost microcontrollers 	<ul style="list-style-type: none"> • The proneness to outliers and failure in non-textured regions
Learning-based	<ul style="list-style-type: none"> • A better solution in ill-posed regions and dynamic environment 	<ul style="list-style-type: none"> • Require large corpus of ground truth data • Heavy reliance on scene priors, may not fit well into unknown environments • A black box, hard to perform further comparisons and migrate to other SLAM systems • May sacrifice efficiency for accuracy • Require high computing power and training time

Chapter 3

Robust Linear-complexity Variational Bayesian Method for Online SLAM Problems

3.1 Introduction

vSLAM and Lidar-based SLAM are the two most common systems in robot navigation (see Chapter 1). Both of them involve a fundamental computational problem of constructing or updating an environmental map while simultaneously localizing the vehicle within this map.

From a Bayesian perspective, this computational problem can be converted into a posterior estimation, where the observed data are the measurements collected from the sensors, and the hidden variables encode the vehicle's pose as well as the environmental map. In online cases, only the most recent pose and map are updated. Existing techniques to approximate the posterior can then be grouped into two categories: filtering and online smoothing. Filtering recursively updates the mean and covariance matrix of the state density. Instead, online smoothing applies MAP [8] to infer the mode of the posterior distribution in an online form.

Both filtering and online smoothing have wide applications in the real world [30, 82, 121, 122, 135, 197], however, meanwhile, they still have the potential to be improved. First of all, the applications of filtering methods with greater prevalence

are hampered by their high computational complexity. The computational cost of the common EKF, UKF, and SEIF, are all cubic in the number of observed features. By contrast, the complexity of online smoothing is linear in the number of features, and that is why it dominates the current SLAM community [168]. The output of online smoothing is a point estimate with no uncertainty information. Therefore, covariance recovery [64, 65] sometimes is required, imposing an extra computational burden. Moreover, the performance of online smoothing approaches is sensitive to the number of retained frames. Last but not least, both filtering and online smoothing are sensitive to inaccurate noise priors as they regard the noise covariances as constants in the estimation procedure. In real-life scenarios, prior statistics of the noise is usually unknown or not accurate enough [188, 208].

To overcome the above limitations, we propose to solve the online SLAM problem by a robust linear-complexity stochastic variational Bayesian inference method. We construct a variational Bayesian inference framework to approximate the unknown posterior distribution via a tractable Gaussian one. The Gaussian distribution is specified to be diagonal, which favors a linear computational complexity. Though a full covariance matrix might improve accuracy, it will significantly deteriorate the computational speed at the same time. Further experiments in this chapter will show that the accuracy is already great with a diagonal covariance matrix. In the implementation, the diagonal entries are typically initialized with large values, and then decreases progressively. This process can be viewed as a simulated annealing [1, 68] or smoothing-based optimization [202], which contributes to finding a maximum globally [101, 202]. To minimize the effect of inaccurate noise pre-estimates, we jointly estimate the noise variances together with the posterior distribution. The approximation is derived from the stochastic gradient ascent, overcoming local extrema. Once the convergence criterion is satisfied, the output is a full posterior distribution, where the covariance can be applied in the future other tasks, such as feature tracking [25, 143], path-constrained and sampling-based motion [16, 57], active reconstruction [64].

3.2 Related Work

This section gives an overview of the most common filtering and online smoothing methods. In addition, we also discuss the application of variational Bayesian

methods in solving online SLAM problems.

Methods that fall in the filtering category include EKF [4, 21, 62, 197], UKF [61, 157], FastSLAM [52, 116, 117], SEIF [176, 183], RB-PHD filter [120] and their variants [76, 133, 188, 190, 207, 208]. Among all these methods, EKF is the most famous and traditional one. It linearizes the system to capture the mean of the posterior. However, this linearization only works in mildly nonlinear nonGaussian (NLNG) systems. Moreover, a full dense uncertainty matrix is maintained and propagated in EKF. As a result, the computational complexity of EKF is a cubic function of the dimension of the map size, severely hindering its application to large-scale problems. To avoid the unreliable linearization, FastSLAM [52, 116, 117] and UKF [61, 157] have been proposed. FastSLAM factorizes the unknown posterior as a product of conditional map points distributions and a distribution over vehicle poses. The distribution of each map point is estimated by EKF, whereas the distribution over vehicle paths is approximated by particle filtering. A Rao-Blackwellized representation then integrates the two distributions. However, typically, FastSLAM is only applicable to low-dimensional problems. For high-dimensional states, the computational complexity and storage requirements are unwieldy. In addition, particle degradation and sample dilution sometimes lead to inefficiency. UKF approximates the nonlinearity through the unscented transform. A minimal set of sigma points are employed to approximate the input density instead of linearization. Nevertheless, UKF still has cubic computational complexity, precluding real-time operation. Most importantly, empirical evidence suggests that the outputs of UKF are still inconsistent in some specific cases [61]. Sparse Extended Information Filter (SEIF) [176, 183] is based on the well-known information form of the EKF. In comparison to EKF, SEIF propagates an information matrix rather than the standard covariance matrix. In addition, it exploits the sparse structure of this information matrix, thereby defines a sparse network of local relative links between pairs of adjacent map points. This insight has powerful ramifications on the computational properties of the SLAM problems. Due to this insight, offline graph-based optimization [31, 51, 84, 87] has been able to undergo a renaissance and currently belongs to one of the dominant techniques concerning speed and accuracy. It is worth noticing that though SEIF is called filter, it belongs to frequentist methods and has more in common with smoothing approaches. The EKF, FastSLAM, UKF, and SEIF all comprise two separate steps: firstly determine the data association, then estimate the vehicle poses as

well as the map points. The performance of these filters is sensitive to data association uncertainty under high clutter and measurement noise. RB-PHD filter is proposed to fully integrate data association uncertainty into the estimation of the map points and vehicle path. It represents the map as a finite set of features. According to random-finite-set (RFS) theory, SLAM is converted into a Bayesian filtering problem, allowing for a joint estimation of the vehicle poses, the map features, and also the number of features. Subsequently, a Rao-Blackwellized (RB) implementation of the probability hypothesis density (PHD) filter is developed for propagating the posterior PHD of the map, as well as the posterior of the vehicle poses.

On the other hand, smoothing needs to be conducted on a set of selected images, such as a sliding window [119, 166], or particular keyframes that are spatially distributed [91, 98]. Traditional smoothing methods estimate the trajectory of the selected frames as well as all the map points observed in these frames. The state-of-the-art smoothing technique is based on a graph-based formulation of the SLAM problem [31, 51] (Chapter 1). It formulates the SLAM problem graphically and seeks the maximum point of the posterior via least-square error minimization. In online applications, the graph grows through time, resulting in a continuously increasing computational cost. Solutions to reduce the computational cost can be divided into two groups: structure-less SFM [66, 146, 147, 164] and incremental smoothing [74, 75]. The former suggests solving the structure from motion (SFM) problem in terms of multi-view constraints instead of reprojection errors, thereby marginalizing out the 3D structure from the optimization. The latter was developed to perform incremental optimization, identifying the poses and map points that need to be recomputed at each step. Both solutions have numerous applications. The fixed-lag smoother is based on incremental smoothing, where a marginalization operation is conducted every time a new frame is inserted [34, 91, 160, 178]. Some other techniques [82, 121, 122, 135, 168] are inspired by “structure-less” BA [66, 67], in which the camera poses are optimized without including the 3D structure. The online SLAM problem is reduced to merely a pose graph, where solely the vehicle pose is optimized. The environmental map is independently updated in another separate thread. In summary, both techniques, partial or sliding-window structures [34, 91, 160, 178], and “structure-less” BA framework adopted often in visual SLAM systems [66, 67], fundamentally are still least square error minimization techniques [174], but are operated differently.

There is only a handful of studies that apply variational Bayes to solving the SLAM problem [105, 113]. The proposed method in [105] is aimed at radar-based SLAM. It applies variational Bayesian expectation maximization (VBEM) instead of inference. The adoption of VBEM is mainly for solving the unknown data associations in a Gaussian mixture modelled map. To enable the applicability of mean-field VBEM, the parameters of map points are assumed to have conjugate priors. However, in practice, the mean and covariance are usually non-conjugate [105], and further approximation strategies are thus required. On the other hand, the priors of weight and intensity are valid only under the assumption that the modeling of measurements is an inhomogeneous Poisson process. This assumption is not reasonable in vSLAM. As a result, VBEM has limited application scenarios. An amortised deep variational Bayes filter is proposed in [113], augmented with a global latent variable representing the environmental map. Two neural networks represent the motion and observation models. The estimation is then converted to the posterior inference in the resulting generative model and approximated via importance-sampling. In the implementation, the neural network receives the newly arrived observations and outputs the approximated parameters. However, the shared weights and biases across datapoints can hardly ensure faithful approximations to all observations. Moreover, the selection of a suitable transition model is laborious and costly. The training of a long, consecutive time series is also a challenging computational task.

3.3 Bayesian Formulation of the Online SLAM Model

In this section, we elaborate on the Bayesian formulation of the online SLAM problem. Without loss of generality, here we focus on the most common 3D feature-based vSLAM (see Figure 3.1).

Specifically, we assume that the vehicle is moving in an unknown rigid scene, while taking measurements of a set of 3D map points $\mathbf{l} := [\mathbf{l}_m]$ in the real world. The information gained from historic poses and measurements is summarized and propagated through time, therefore, the future vehicle pose $\mathbf{s}_{k+1}$ is independent of the historical poses $\mathbf{s}_1, \dots, \mathbf{s}_{k-1}$ given the current pose $\mathbf{s}_k$. In practice, we typically

stack pose state $\mathbf{s}_k$ and observed map points $\mathbf{l} := [\mathbf{l}_m]$ into one unified vector $\mathbf{x}_k \in \mathbb{R}^D$ to serve as the “state” vector at time k .

FIGURE 3.1: The general feature-based SLAM problem.

A feature-based SLAM system pre-processes the input, such as camera images, in the front-end to extract keypoints [36, 82, 121, 122], and then optimizes the initial guess $\mathbf{x}_k$ with the observations in the back-end (see Figure 2.2). Each keypoint corresponds to a well-localized map point in the real-world space, being the observations of map points $\mathbf{l}$ in the images. In a vSLAM system, the keypoints can be classified as stereo and monocular. We denote stereo keypoints as $[(u_L, v_L, u_R)]$, being (u_L, v_L) the pixel coordinates in the left image and u_R the horizontal coordinate on the right image. The monocular keypoints are denoted as $[(u_L, v_L)]$, the coordinates solely on the left image. For convenient calculation, in practice, we arrange the coordinates of all the keypoints sequentially to form the measurement vector $\mathbf{z}_k \in \mathbb{R}^M$.

We denote the noise in motion and observation processes as $\boldsymbol{\epsilon}_k$ and $\boldsymbol{\delta}_k$ respectively. Each of both is assumed to be additive and independent. As a result, the state

space model of time instant k is described as:

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, \mathbf{u}_k) + \boldsymbol{\epsilon}_k, \quad (3.1a)$$

$$\mathbf{z}_k = h(\mathbf{x}_k) + \boldsymbol{\delta}_k, \quad (3.1b)$$

where $\mathbf{u}_k$ is a vector imposed on $\mathbf{x}_{k-1}$ that controls the transition between current and previous states, while f and h are motion and observation functions, respectively.

The mounted cameras are moving with the vehicle together. We define the base coordinate as W and the varying local “robot” base (the moving camera frame) as R . In the 3D scenarios, the vehicle pose $\mathbf{s}_k$ can be decomposed as 3D position vector $\mathbf{r}_k^W$ and three Euler angles $\mathbf{e}_k^{WR}$, where the superscripts indicate the frames in which the quantities are expressed. In the calculations, we usually convert $\mathbf{e}_k^{WR}$ to a 3×3 rotation matrix T_k^{WR} when required.

At each time step, to obtain the current pose $\mathbf{x}_k$, we have a control vector $\mathbf{u}_k$ that comprises translation velocity $\mathbf{v}_k^R$ and angular velocity $\boldsymbol{\omega}_k^R$. Since time interval between frames is generally assumed to be fixed, the motion function f in (3.1a) then can be written as:

$$f(\mathbf{x}_{k-1}, \mathbf{u}_k) = \begin{pmatrix} \mathbf{r}_k^W \\ \mathbf{e}_k^{WR} \\ \mathbf{l}_k \end{pmatrix} = \begin{pmatrix} \mathbf{r}_{k-1}^W + T_{k-1}^{WR} \mathbf{v}_k^R \Delta t \\ R2e(T_{k-1}^{WR} e2R(\boldsymbol{\omega}_k^R \Delta t)) \\ \mathbf{l}_{k-1} \end{pmatrix}, \quad (3.2)$$

where $R2e$ converts T_k^{WR} to $\mathbf{e}_k^{WR}$, and $e2R$ works in the reverse direction.

As the correspondences of 3D map points, the keypoints are expected to be found in the image through the standard pinhole model. Following the pinhole camera model, the stereo and monocular projections are correspondingly predicted as follows:

$$\begin{bmatrix} u_L \\ v_L \\ u_R \end{bmatrix} = h_s([X, Y, Z]^T) = \begin{bmatrix} u_0 + f_x \frac{X}{Z} \\ v_0 + f_y \frac{Y}{Z} \\ u_0 + f_x \frac{X-b}{Z} \end{bmatrix}, \quad (3.3a)$$

$$\begin{bmatrix} u_L \\ v_L \end{bmatrix} = h_m([X, Y, Z]^T) = \begin{bmatrix} u_0 + f_x \frac{X}{Z} \\ v_0 + f_y \frac{Y}{Z} \end{bmatrix}, \quad (3.3b)$$

where u_0, v_0, f_x, f_y are camera calibration parameters, b is the baseline, h_s and h_m are observation functions for stereo and monocular keypoints, respectively.

In (3.3a) and (3.3b), $[X, Y, Z]^T$ is the position of the keypoint's corresponding map point expressed in the robot frame coordinate. The corresponding map point $\mathbf{l}_m$ has relative position to the image frame:

$$[X, Y, Z]^T = T_k^{RW}([\mathbf{l}_m]_k^W - \mathbf{r}_k^W), \quad (3.4)$$

where T_k^{RW} denotes the rotation matrix of the world frame relative to the vehicle frame.

When the data association information is known, the objective of the online SLAM problem is to estimate $\mathbf{x}_k$ given $\mathbf{x}_{k-1}$ and $\mathbf{z}_k$. From the viewpoint of probability, it involves approximating the posterior $p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{z}_k, \boldsymbol{\delta}_k, \boldsymbol{\epsilon}_k)$. In the SLAM community, $\boldsymbol{\epsilon}_k$ and $\boldsymbol{\delta}_k$ in (3.1) are generally assumed to be white noise. Here we set $\boldsymbol{\epsilon}_k$ to follow a Gaussian distribution $\mathcal{N}(\mathbf{0}; \Lambda_k)$, whereas $\boldsymbol{\delta}_k \sim \mathcal{N}(\mathbf{0}; \Psi_k)$. Both Λ_k and Ψ_k are assumed to be diagonal. In reality, the statistical characteristics of both Λ_k and Ψ_k change easily due to time-varying internal and external uncertainties. In previous studies, their values are often set by consulting datasheets or from experience [30, 51, 197], which is often suboptimal. Even if Λ_k and Ψ_k have already been determined through pre-testing on a training set, we cannot guarantee that the obtained values are applicable to all cases. To obtain accurate noise information, the motion noise covariance is estimated on-the-fly in [197] through the time-consuming bootstrap approach. On the other hand, the proposed adaptive UKF in [188, 208] estimates noise on-the-fly but cannot handle complex scenarios effectively [72].

In this work, we update Λ_k and Ψ_k along with the posterior distribution p to handle incorrect noise pre-estimates. In contrast to previous studies [4, 21, 61, 62, 197], the proposed SVBI approach does not introduce heavy additional computational burden, nor does it diverge. We further rewrite Λ_k and Ψ_k as $(\boldsymbol{\alpha}_k \circ I_D)^{-1}$ and $(\boldsymbol{\beta}_k \circ I_M)^{-1}$ respectively, where both I_D and I_M are identity matrices with dimension D and M respectively, and $\circ$ represents the element-wise product. As such, the process noise $\boldsymbol{\epsilon}_k$ and $\boldsymbol{\delta}_k$ are eventually encoded by two vectors $\boldsymbol{\alpha}_k$ and $\boldsymbol{\beta}_k$, respectively.

The state space model (3.1) can, therefore, be described in a probabilistic manner as:

$$p(\mathbf{x}_k | \mathbf{x}_{k-1}, \boldsymbol{\alpha}_k) \propto \sqrt{\prod_{i=1}^D (\boldsymbol{\alpha}_k)_i} \exp \left\{ -\frac{1}{2} (\mathbf{x}_k - f(\mathbf{x}_{k-1}))^T \text{diag}(\boldsymbol{\alpha}_k) (\mathbf{x}_k - f(\mathbf{x}_{k-1})) \right\}, \quad (3.5a)$$

$$p(\mathbf{z}_k | \mathbf{x}_k, \boldsymbol{\beta}_k) \propto \sqrt{\prod_{j=1}^M (\boldsymbol{\beta}_k)_j} \exp \left\{ -\frac{1}{2} (\mathbf{z}_k - h(\mathbf{x}_k))^T \text{diag}(\boldsymbol{\beta}_k) (\mathbf{z}_k - h(\mathbf{x}_k)) \right\}, \quad (3.5b)$$

where $(\boldsymbol{\alpha}_k)_i$ and $(\boldsymbol{\beta}_k)_j$ are the i^{th} and j^{th} elements of vectors $\boldsymbol{\alpha}_k$ and $\boldsymbol{\beta}_k$, respectively. We drop the dependence of $\mathbf{u}_k$ in the notation of f for brevity.

Furthermore, the above posterior distribution of $\mathbf{x}_k$ is rewritten as $p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{z}_k, \boldsymbol{\alpha}_k, \boldsymbol{\beta}_k)$. Under Bayes's laws, it can be factorized as:

$$p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{z}_k, \boldsymbol{\alpha}_k, \boldsymbol{\beta}_k) \propto p(\mathbf{z}_k | \mathbf{x}_k, \boldsymbol{\alpha}_k) p(\mathbf{x}_k | \mathbf{x}_{k-1}, \boldsymbol{\beta}_k). \quad (3.6)$$

3.4 Variational Bayesian Inference for SLAM

In this section, we derive a variational Bayesian inference algorithm for online SLAM problem proposed in Section 3.3. We approximate the exact posterior distribution $p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{z}_k, \boldsymbol{\alpha}_k, \boldsymbol{\beta}_k)$ in (3.6) by a tractable variational distribution q under the framework of variational Bayes. We first construct the evidence lower bound (ELBO), and then maximize this ELBO via stochastic gradient ascent.

3.4.1 Variational Bayesian Inference

In the SLAM problem, given the previous vehicle state $\mathbf{x}_{k-1}$ and the current measurements $\mathbf{z}_k$, we aim to obtain the noise estimates of $\boldsymbol{\alpha}_k$ and $\boldsymbol{\beta}_k$ and further infer the full posterior distribution $p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{z}_k, \boldsymbol{\alpha}_k, \boldsymbol{\beta}_k)$. To this end, one straightforward approach is maximizing the likelihood. Hence we seek $\boldsymbol{\alpha}_k$ and $\boldsymbol{\beta}_k$ that

maximize the loglikelihood:

$$\begin{aligned}\log p(\mathbf{z}_k | \mathbf{x}_{k-1}, \boldsymbol{\alpha}_k, \boldsymbol{\beta}_k) &= \log \int p(\mathbf{x}_k, \mathbf{z}_k | \mathbf{x}_{k-1}, \boldsymbol{\alpha}_k, \boldsymbol{\beta}_k) d\mathbf{x}_k, \\ &= \log \int p(\mathbf{x}_k | \mathbf{x}_{k-1}, \boldsymbol{\alpha}_k) p(\mathbf{z}_k | \mathbf{x}_k, \boldsymbol{\beta}_k) d\mathbf{x}_k.\end{aligned}\quad (3.7)$$

Unfortunately, the above integration in (3.7) is intractable due to complicated nonlinear functional forms f and h in (3.5).

To resolve the issue, as elaborated in Appendix A.1, we maximize the evidence lower bound (ELBO) $\mathcal{L}$ [13] of the $\log p(\mathbf{z}_k, \mathbf{x}_{k-1} | \boldsymbol{\alpha}_k, \boldsymbol{\beta}_k)$ instead, that is,

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_q \left[\log \frac{p(\mathbf{x}_k, \mathbf{z}_k | \mathbf{x}_{k-1}, \boldsymbol{\alpha}_k, \boldsymbol{\beta}_k)}{q(\mathbf{x}_k)} \right] \\ &= \mathbb{E}_q [\log p(\mathbf{x}_k, \mathbf{z}_k | \mathbf{x}_{k-1}, \boldsymbol{\alpha}_k, \boldsymbol{\beta}_k)] + H[q(\mathbf{x}_k)], \\ &\leq \log p(\mathbf{z}_k | \mathbf{x}_{k-1}, \boldsymbol{\alpha}_k, \boldsymbol{\beta}_k),\end{aligned}\quad (3.8)$$

where $\mathbb{E}_q$ denotes the expectation over distribution q and $H[q(\mathbf{x}_k)]$ is the entropy of the variational distribution q . When $q(\mathbf{x}_k)$ equals $p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{z}_k, \boldsymbol{\alpha}_k, \boldsymbol{\beta}_k)$, $\mathcal{L}$ reaches a maximum and equality thus holds.

However, due to the complex models in our problem, the closed-form expression of $p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{z}_k, \boldsymbol{\alpha}_k, \boldsymbol{\beta}_k)$ is not available. To circumvent this, we therefore resort to the variational inference method and approximate $p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{z}_k, \boldsymbol{\alpha}_k, \boldsymbol{\beta}_k)$ by a tractable variational distribution $q(\mathbf{x}_k)$. Interestingly, as discussed in Appendix A.1, under variational Bayesian theory, maximizing $\mathcal{L}$ is mathematically equivalent to minimizing the Kullback-Leibler (KL) divergence [137] of q to p .

Generally, in the implementation, $q(\mathbf{x}_k)$ should be restricted to be within a family of tractable densities [12]. In this work, we set $q(\mathbf{x})$ to follow a Gaussian distribution. The covariance matrix of this Gaussian distribution is diagonal, giving rise to a variational inference algorithm with linear computational complexity [177]. Although off-diagonal entries in the covariance may improve the approximation, they would result in extra computational and storage burden. In addition, our experimental results in the following sections demonstrate that the proposed $q(\mathbf{x}_k)$ with a diagonal covariance matrix has led to reasonable approximations.

Additionally, we borrow the idea of reparameterization from [81, 85, 150, 177], and reparameterize $\mathbf{x}_k$ as:

$$\mathbf{x}_k = U_k^{-1} \mathbf{e} + \boldsymbol{\mu}_k, \quad (3.9)$$

where $\mathbf{e} \in \mathbb{R}^D$ is a random vector following the standard normal distribution $\phi(\mathbf{e})$, $\boldsymbol{\mu}_k$ denotes the real mean vector and the scale matrix $U_k \in \mathbb{R}^{D \times D}$ is a diagonal matrix with positive entries. In general, we initialize $\boldsymbol{\mu}_k$ as $f(\mathbf{x}_{k-1}, \mathbf{u}_k)$ through the prediction function (3.2) and the diagonal elements of U_k as reasonable large values.

After the reparameterization, $\mathbf{x}_k$ thus follows the Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_k, U_k U_k^T)$. Correspondingly, the ELBO $\mathcal{L}$ in (3.8) can be rewritten as:

$$\mathcal{L} = \mathbb{E}_{\phi(\mathbf{e})} [\log p(U_k^{-1} \mathbf{e} + \boldsymbol{\mu}_k | \mathbf{x}_{k-1}, \boldsymbol{\alpha}_k) + \log p(\mathbf{z}_k | U_k^{-1} \mathbf{e} + \boldsymbol{\mu}_k, \boldsymbol{\beta}_k)] - \log |U_k| + H[\mathbf{e}], \quad (3.10)$$

where $H[\mathbf{e}]$ denotes the entropy of $\phi(\mathbf{e})$ and is a constant.

3.4.2 Stochastic Gradient Ascent

In order to find out the optimal q that maximizes $\mathcal{L}$, we resort to the gradient ascent method to seek for the corresponding $\boldsymbol{\mu}_k$, U_k , $\boldsymbol{\alpha}_k$, and $\boldsymbol{\beta}_k$. Based on Appendix A.2, the exact gradients of $\mathcal{L}$ with respect to the above variables are as follows:

$$\nabla_{\boldsymbol{\mu}_k} \mathcal{L} = \mathbb{E}_{\phi(\mathbf{e})} \left[(\boldsymbol{\beta}_k \circ \nabla_{\mathbf{x}_k} h(\mathbf{x}_k))^T (\mathbf{z}_k - h(\mathbf{x}_k)) - \boldsymbol{\alpha}_k \circ (\mathbf{x}_k - f(\mathbf{x}_{k-1})) \right], \quad (3.11a)$$

$$\begin{aligned} \nabla_{U_k} \mathcal{L} = & - (U_k^T)^{-1} \text{diag} \left(\mathbb{E}_{\phi(\mathbf{e})} \left[\left\{ (\boldsymbol{\beta}_k \circ \nabla_{\mathbf{x}_k} h(\mathbf{x}_k))^T (\mathbf{z}_k - h(\mathbf{x}_k)) \right. \right. \right. \\ & \left. \left. \left. - \boldsymbol{\alpha}_k \circ (\mathbf{x}_k - f(\mathbf{x}_{k-1})) \right\} \cdot \mathbf{e}^T \right] \right) (U_k^T)^{-1} - \Delta U_k, \end{aligned} \quad (3.11b)$$

$$\nabla_{\boldsymbol{\alpha}_k} \mathcal{L} = -\frac{1}{2} (\mathbf{x}_k - f(\mathbf{x}_{k-1})) \circ (\mathbf{x}_k - f(\mathbf{x}_{k-1})) + \frac{1}{2} \mathbf{1} \oslash \boldsymbol{\alpha}_k, \quad (3.11c)$$

$$\nabla_{\boldsymbol{\beta}_k} \mathcal{L} = -\frac{1}{2} \mathbf{z}_k \circ \mathbf{z}_k + \mathbb{E}_{\phi(\mathbf{e})} [h(\mathbf{x}_k)] \circ \mathbf{z}_k - \frac{1}{2} \mathbb{E}_{\phi(\mathbf{e})} [h(\mathbf{x}_k) \circ h(\mathbf{x}_k)] + \frac{1}{2} \mathbf{1} \oslash \boldsymbol{\beta}_k, \quad (3.11d)$$

where $\Delta U_k := [1/U_{ii}]$ is a diagonal matrix and U_{ii} denotes the i^{th} entry of U_k on the diagonal, $\mathbf{1}$ is a vector of all ones, and $\oslash$ represents element-wise division.

The expectations $\mathbb{E}_{\phi(\mathbf{e})}$ in the exact gradients (3.11), however, are still intractable. Since we have introduced the randomness through the reparameterization trick in (3.9), as a remedy, we further exploit stochastic gradients instead to maximize the ELBO $\mathcal{L}$. Stochastic gradients [14, 18, 145, 151, 163] are unbiased estimators of the exact gradients [5, 15, 129], but can be computed efficiently when the exact gradients cannot be calculated directly. In our case, we first draw a sample $\mathbf{e}$ from $\phi(\mathbf{e})$. Then as an example of $\mathbf{x}_k$, the $\bar{\mathbf{x}}_k$ is obtained by (3.9) with the initialized U_k and $\boldsymbol{\mu}_k$. We replace the expectations $\mathbb{E}_{\phi(\mathbf{e})}[g(\mathbf{x}_k)]$ in (3.11) with $g(\bar{\mathbf{x}}_k)$ to obtain the stochastic gradients, where g denotes an arbitrary function of $\mathbf{x}_k$.

As such, the exact intractable gradients in (3.11), except (3.11c), are approximated by the following stochastic gradients:

$$\nabla_{\boldsymbol{\mu}_k} \mathcal{L} \approx (\boldsymbol{\beta}_k \circ \nabla_{\mathbf{x}_k} h(\mathbf{x}_k))^T (\mathbf{z}_k - h(\mathbf{x}_k)) - \boldsymbol{\alpha}_k \circ (\mathbf{x}_k - f(\mathbf{x}_{k-1})), \quad (3.12a)$$

$$\begin{aligned} \nabla_{U_k} \mathcal{L} \approx & - (U_k^T)^{-1} \text{diag} \left[\left\{ (\boldsymbol{\beta}_k \circ \nabla_{\mathbf{x}_k} h(\mathbf{x}_k))^T (\mathbf{z}_k - h(\mathbf{x}_k)) \right. \right. \\ & \left. \left. - \boldsymbol{\alpha}_k \circ (\mathbf{x}_k - f(\mathbf{x}_{k-1})) \right\} \cdot \mathbf{e}^T \right] (U_k^T)^{-1} - \Delta U_k, \end{aligned} \quad (3.12b)$$

$$\nabla_{\boldsymbol{\beta}_k} \mathcal{L} \approx -\frac{1}{2} \mathbf{z}_k \circ \mathbf{z}_k + h(\mathbf{x}_k) \circ \mathbf{z}_k - \frac{1}{2} h(\mathbf{x}_k) \circ h(\mathbf{x}_k) + \frac{1}{2} \mathbf{1} \oslash \boldsymbol{\beta}_k. \quad (3.12c)$$

Finally, we update $\boldsymbol{\mu}_k$, U_k , $\boldsymbol{\alpha}_k$, and $\boldsymbol{\beta}_k$ in a recursive manner:

$$\boldsymbol{\mu}_k^{(t)} = \boldsymbol{\mu}_k^{(t-1)} + \rho_t \nabla_{\boldsymbol{\mu}_k} \mathcal{L}^{(t)}, \quad (3.13)$$

$$U_k^{(t)} = U_k^{(t-1)} + \rho_t \nabla_{U_k} \mathcal{L}^{(t)}, \quad (3.14)$$

$$\boldsymbol{\alpha}_k^{(t)} = \boldsymbol{\alpha}_k^{(t-1)} + \rho_t \nabla_{\boldsymbol{\alpha}_k} \mathcal{L}^{(t)}, \quad (3.15)$$

$$\boldsymbol{\beta}_k^{(t)} = \boldsymbol{\beta}_k^{(t-1)} + \rho_t \nabla_{\boldsymbol{\beta}_k} \mathcal{L}^{(t)}, \quad (3.16)$$

where t in the bracket denotes the t^{th} iteration and $\rho_t > 0$ is the step size (a.k.a. learning rate) in the t^{th} iteration.

According to the Robbins-Monro theorem [144], the stochastic gradient ascent method is guaranteed to converge to a local maximum if ρ_t satisfies the following conditions:

$$\sum_1^\infty \rho_t^2 < \infty \ \& \ \sum_1^\infty \rho_t = \infty. \quad (3.17)$$

In practice, we adopt the Adam method proposed in [80] to determine ρ_t adaptively and automatically.

We summarize the overall algorithm in Table 3.1.

3.4.3 Analysis

Computational efficiency is essential for online SLAM problems. Increasing the number of features is generally more productive than the number of frames for performance improvement [168]. Under this condition, filtering is not applicable anymore due to the cubical complexity. For instance, the computational complexity of both EKF [4, 197] and UKF [61, 157] are $\mathcal{O}(M^3)$. By contrast, the complexity of online smoothing is $\mathcal{O}(K^3 + K^2G)$, where K is the number of frames to be optimized, G is the dimension of the local map that is proportional to M . The first term K^3 represents the cost of solving the reduced linear system, while the K^2G term reflects the Schur complement [40, 168]. Since online smoothing has a linear complexity to the number of features, it is more prevalent in the current SLAM community. When it comes to the proposed online SVBI in Table 3.1, since the covariance matrix U_k has been set to be diagonal, the resulting computational complexity is thus only $\mathcal{O}(DM)$. It is linear with the dimension of the state and measurement vector.

Moreover, EKF has to store a full covariance matrix at each time instant, whereas the proposed SVBI algorithm only saves a diagonal matrix U_k . Hence, SVBI has the merit of memory efficiency as well. Simulation results in the following sections will demonstrate this point.

3.5 Simulation Results

In this section, we evaluate the proposed online SVBI in Table 3.1 through numerical simulations to assess its performance under different conditions. We first compare online SVBI with other existing algorithms regarding different noise levels and noise variance inputs. In addition, we investigate how the computational cost scales with the number of feature points.

TABLE 3.1: Linear-complexity stochastic variational Bayes inference for online SLAM problem.

Input: $\mathbf{x}_{k-1}, \mathbf{z}_k$.

All operations on vectors here are element-wise. $\left(g_k^{(t)}\right)^2$ denotes the element-wise square; κ_1^t, κ_2^t indicates κ_1, κ_2 to power t .

1. Exponential decay rates $\kappa_1 = 0.9, \kappa_2 = 0.999$ and $\varepsilon = 10^{-8}$, λ is the step size.
2. Initialize $\boldsymbol{\mu}_k^{(0)}, U_k^{(0)}, \boldsymbol{\alpha}_k^{(0)}$ and $\beta_k^{(0)}$.
3. Set $t = 0$, both the 1st and 2nd moment vectors are initialized as: $m_\Xi^{(0)} = 0, v_\Xi^{(0)} = 0$ for $\Xi \in \{\boldsymbol{\mu}_k, U_k, \boldsymbol{\alpha}_k, \beta_k\}$
4. **Repeat** the following procedure until $\boldsymbol{\mu}_k, U_k, \boldsymbol{\alpha}_k$ and β_k all converge.
 - (a) $t = t + 1$.
 - (b) Draw a sample $\mathbf{e} \sim \phi(\mathbf{e})$.
 - (c) Compute gradients $g_k^{(t)}: \nabla_{\boldsymbol{\mu}_k} \mathcal{L}, \nabla_{U_k} \mathcal{L}, \nabla_{\boldsymbol{\alpha}_k} \mathcal{L}$ and $\nabla_{\beta_k} \mathcal{L}$ following (3.11).
 - (d) Update the biased 1st and 2nd moment vectors for $\Xi \in \{\boldsymbol{\mu}_k, U_k, \boldsymbol{\alpha}_k, \beta_k\}$ separately:

$$\begin{aligned} m_\Xi^{(t)} &= \kappa_1 \cdot m_\Xi^{(t-1)} + (1 - \kappa_1) \cdot g_k^{(t)} \\ v_\Xi^{(t)} &= \kappa_2 \cdot v_\Xi^{(t-1)} + (1 - \kappa_2) \cdot \left(g_k^{(t)}\right)^2. \end{aligned}$$

- (e) Update the bias-corrected 1st and 2nd moment vector for $\Xi \in \{\boldsymbol{\mu}_k, U_k, \boldsymbol{\alpha}_k, \beta_k\}$:

$$\hat{m}_\Xi^{(t)} = \frac{m_\Xi^{(t)}}{1 - \kappa_1^t}, \quad \hat{v}_\Xi^{(t)} = \frac{v_\Xi^{(t)}}{1 - \kappa_2^t}.$$

- (f) Update final estimated quantity for $\Xi \in \{\boldsymbol{\mu}_k, U_k, \boldsymbol{\alpha}_k, \beta_k\}$:

$$\Xi^{(t)} = \Xi^{(t-1)} + \frac{\lambda \cdot \hat{m}_\Xi^{(t)}}{\sqrt{\hat{v}_\Xi^{(t)}} + \varepsilon}.$$

Output: $\hat{\mathbf{x}}_k = \boldsymbol{\mu}_k$.

3.5.1 Simulation Setup

First we generate the true motion paths according to the motion function (3.2). Assume the vehicle is moving in a 3D environment. We set the true translational velocity in the X, Y, Z axis as $[0.5, 0.5, 0.5]$ m per frame, and the true angular velocity as $[0.005, 0.005, 0.005]$ rad per frame.

Based on the ground truth, the input translational and angular velocities for the tested SLAM algorithms are corrupted by time-varying zero-mean normal distributed process noise. The noise on each axis is assumed to be independent and the standard deviations are randomly selected from $\mathcal{U}(0, 0.6a)$ and $\mathcal{U}(0, 0.002c)$ at every time instant, respectively. Here, a and c are two positive scalars that will be set later.

Following [168], we consider a local motion where the number of frames k is set to be 16. In particular, two general scenes are simulated: 1) all map points are visible in all frames; 2) the map points are only partially overlapped between neighbouring frames. The number of observations per frame is chosen as 240. We initialize the map points by the properly selected observations. The calibration parameters follow the setting of the widely-used KITTI dataset [49]. At every moment, based on the simulated vehicle pose, the true observations are given by the observation function in (3.3). We take typical measurement error characteristic into account, and add time-varying zero-mean white noise to the observations, whose standard deviation is randomly drawn from the uniform distribution $\mathcal{U}(0, g)$. Here g is a positive scalar to be set later.

We benchmark the previously described approach in Table 3.1 against the following aforementioned SLAM algorithms:

1. Extended Kalman Filter (EKF) [4]
2. Unscented Kalman Filter (UKF) [61]
3. Rao-Blackwellized Conditional Particle Filter (FastSLAM) [116]
4. Sliding-window Graph-based Optimization (SGO) [91]
5. Bundle Adjustment SLAM (BA SLAM) [168].

We consider three scenarios:

1. **Different noise levels:** The setting of (a, c, g) is listed as below, where a and c are the amplitudes of the translation and rotation noise, g is the amplitude of the measurement noise.

$(0.1, 0.1, 0.25)$	$(0.1, 0.1, 0.5)$	$(0.1, 0.1, 1)$	$(0.05, 0.05, 1)$	$(0.2, 0.2, 1)$
--------------------	-------------------	-----------------	-------------------	-----------------

2. **Different input noise variances:** We denote the true variances of translation, rotation, and observation noise as γ_t , γ_r , and β . To simulate the real-life scenario where the true noise priors are unknown and their estimates are possibly inaccurate, we test the algorithms under different input variances:

$(\gamma_t, \gamma_r, \beta)$	$(\hat{\gamma}_t, \gamma_r, \beta)$	$(\gamma_t, \hat{\gamma}_r, \beta)$	$(\gamma_t, \gamma_r, \hat{\beta})$	$(\hat{\gamma}_t, \hat{\gamma}_r, \hat{\beta})$
-------------------------------	-------------------------------------	-------------------------------------	-------------------------------------	---

Here, for $\hat{\boldsymbol{\eta}} \in \{\hat{\gamma}_t, \hat{\gamma}_r, \hat{\beta}\}$, $\hat{\boldsymbol{\eta}} = d\boldsymbol{\eta}$, where d is a random number drawn from the uniform distribution $\mathcal{U}(0, 2)$.

3. **Different number of observations per frame:** We measure the computational cost (run time, CPU utilization, memory footprint) of all the algorithms with respect to different numbers of observations per frame:

60	120	240	360	480	600	1200	2400	3600	4800	6000	7200
----	-----	-----	-----	-----	-----	------	------	------	------	------	------

We average all results over 100 runs. The accuracy is evaluated by the root mean square error (RMSE) of the position and rotation angles. All of our experiments were performed on a PC with Intel(R) Xeon(R) CPU E5-2630 v3@ 2.40GHz and DIMM 2133 MHz RAM.

3.5.2 Different Noise Levels

Here, the initial uncertainty matrix (normally denoted as P) in EKF and UKF is specified as a diagonal matrix with small positive diagonal entries. The size of the sliding window in SGO is chosen to be 3. The number of particles in FastSLAM is 1000, and the threshold of effective particles is set to be 800. Note that we stick to the same settings in the following experiments. Additionally, in this scenario, the input noise variances are set to be their true values.

TABLE 3.2: Average RMSEs of EKF, UKF, FastSLAM, SGO, BA SLAM and online SVBI for different noise levels.

(a, c, g)	EKF		UKF		FastSLAM	
	Position [m]	Angle [rad]	Position [m]	Angle [rad]	Position [m]	Angle [rad]
(0.1, 0.1, 0.25)	0.0661	0.1594e-3	0.0618	0.1539e-3	0.0659	0.4179e-3
(0.1, 0.1, 0.5)	0.0678	0.1634e-3	0.0645	0.1596e-3	0.0693	0.3530e-3
(0.1, 0.1, 1)	0.0679	0.2077e-3	0.0659	0.2073e-3	0.0665	0.3365e-3
(0.05, 0.05, 1)	0.0383	0.1473e-3	0.0371	0.1475e-3	0.0410	0.4335e-3
(0.2, 0.2, 1)	0.1434	0.3868e-3	0.1381	0.3759e-3	0.1206	0.4482e-3

(a, c, g)	SGO		BA SLAM		Online SVBI	
	Position [m]	Angle [rad]	Position [m]	Angle [rad]	Position [m]	Angle [rad]
(0.1, 0.1, 0.25)	0.0295	0.0730e-3	0.0277	0.0697e-3	0.0281	0.0711e-3
(0.1, 0.1, 0.5)	0.0594	0.1409e-3	0.0563	0.1367e-3	0.0391	0.1370e-3
(0.1, 0.1, 1)	0.1184	0.2934e-3	0.1113	0.2801e-3	0.0509	0.2022e-3
(0.05, 0.05, 1)	0.1110	0.2852e-3	0.1066	0.2769e-3	0.0301	0.1399e-3
(0.2, 0.2, 1)	0.1105	0.2710e-3	0.1041	0.2583e-3	0.0788	0.2522e-3

TABLE 3.3: Average run time per frame for EKF, UKF, FastSLAM, SGO, BA SLAM and online SVBI under different noise levels.

(a, c, g)	EKF	UKF	FastSLAM	SGO	BA SLAM	Online SVBI
	[s]	[s]	[s]	[s]	[s]	[s]
(0.1, 0.1, 0.25)	0.0597	13.4494	31.3664	8.8286	27.5748	2.8831
(0.1, 0.1, 0.5)	0.0579	13.2438	32.6330	10.8297	29.4993	2.6838
(0.1, 0.1, 1)	0.0596	12.7743	33.7875	12.2054	32.3353	2.7532
(0.05, 0.05, 1)	0.0610	13.6048	33.5410	12.1369	31.9539	2.7280
(0.2, 0.2, 1)	0.0616	13.7050	31.7896	12.2787	32.4769	2.7047

We present the averaged RMSEs of all the testing algorithms across all the frames in Table 3.2. In addition, their average run time per frame is presented in Table 3.3.

From Table 3.2 and Table 3.3, we observe that:

1. The estimation errors of all the algorithms increase along with the observation noise.
2. The larger the motion noise, the higher the estimation errors of EKF, UKF, FastSLAM, and online SVBI. However, the accuracy of SGO and BA SLAM do not vary in pace with the motion noise.

One possible explanation could be that the motion constraints between frames are a key factor to determine accurate trajectories [75], but they are not included in the cost function of SGO and BA SLAM. When the motion noise level is relatively high (e.g., 0.2), the coefficients associated with the motion constraints become small. As such, the influencing degree of dismissing the motion errors decrease and the accuracy rise instead.

3. EKF achieves the fastest speed overall. In comparison to BA SLAM, SGO distinctly increases the processing speed at the cost of accuracy.
4. (a) Online SVBI has yields the best overall accuracy.
 - (b) In the case of large motion noise but small observation noise, smoothing methods (SGO and BA SLAM) outperform filtering algorithms (EKF, UKF, and FastSLAM). Conversely, filtering achieves better performance when the motion noise is small whereas the observation noise is large.
 - (c) UKF and FastSLAM have slower deliver superior accuracy over EKF with higher processing speed.

3.5.3 Different Input Noise Variances

An accurate estimation of the process states is vital for the safe operation and efficient management of intelligent vehicles. Since EKF, UKF, FastSLAM and BA employ the statistical properties of the random disturbances and measurements to approximate the posterior distribution of the state estimates, their accuracy is heavily reliant on the input noise variances. In Section 3.5.2, we tested all the algorithms under true noise statistics. However, in practical applications, the noise statistics are usually unknown and are set by experience [30] or methods, such as bootstrapping [4] and multiscale image pyramids [121, 122], without assurance of correctness. Therefore, to ensure the system performance under uncertain noise statistics, the robustness to noise estimation errors is crucial for the testing algorithms.

In this test, we compare the robustness of the aforementioned algorithms under inaccurate noise priors. The (a, c, g) values are chosen to be $(0.2, 0.2, 1)$. As indicated in section 3.5.1, we provide the testing algorithms with 5 different input settings. We first consider the case with the true noise variances, and then randomly set

the variance of the translation noise, the rotation noise, the observation noise, and all types of noise in the remaining four cases. The average estimation errors are presented in Table 3.4. Correspondingly, Table 3.5 presents the average processing speed per frame. It is worth noticing that the performance of SGO, BA SLAM, and Online SVBI do not use motion noise directly. As a consequence, all of them will not be affected by the variation of translation and rotation noise variances.

TABLE 3.4: Average RMSEs of EKF, UKF, FastSLAM, SGO, BA SLAM and online SVBI for different input noise variances.

Input Noise Variances	EKF		UKF		FastSLAM	
	Position [m]	Angle [rad]	Position [m]	Angle [rad]	Position [m]	Angle [rad]
$(\gamma_t, \gamma_r, \beta)$	0.1434	0.3868e-3	0.1381	0.3759e-3	0.1206	0.4482e-3
$(\hat{\gamma}_t, \gamma_r, \beta)$	0.1462	0.3889e-3	0.1394	0.3776e-3	0.1268	0.4758e-3
$(\gamma_t, \hat{\gamma}_r, \beta)$	0.1434	0.3868e-3	0.1381	0.3759e-3	0.1357	0.6010e-3
$(\gamma_t, \gamma_r, \hat{\beta})$	0.2161	0.6077e-3	0.1897	0.5777e-3	0.1985	0.7981e-3
$(\hat{\gamma}_t, \hat{\gamma}_r, \hat{\beta})$	0.2088	0.6241e-3	0.2038	0.5792e-3	0.2013	0.8896e-3

Input Noise Variances	SGO		BA SLAM		Online SVBI	
	Position [m]	Angle [rad]	Position [m]	Angle [rad]	Position [m]	Angle [rad]
$(\gamma_t, \gamma_r, \beta)$	0.1105	0.0003	0.1041	0.2583e-3	0.0788	0.2522e-3
$(\hat{\gamma}_t, \gamma_r, \beta)$	/	/	/	/	/	/
$(\gamma_t, \hat{\gamma}_r, \beta)$	/	/	/	/	/	/
$(\gamma_t, \gamma_r, \hat{\beta})$	0.2356	0.0017	0.2655	0.7641e-3	0.0933	0.3371e-3
$(\hat{\gamma}_t, \hat{\gamma}_r, \hat{\beta})$	/	/	/	/	/	/

TABLE 3.5: Average run time per frame for EKF, UKF, FastSLAM, SGO, BA SLAM and online SVBI under different input noise variances.

Input Noise Variances	EKF [s]	UKF [s]	FastSLAM [s]	SGO [s]	BA SLAM [s]	Online SVBI [s]
$(\gamma_t, \gamma_r, \beta)$	0.0616	13.7050	31.7896	12.2787	32.4769	2.7047
$(\hat{\gamma}_t, \gamma_r, \beta)$	0.0621	13.3944	31.4744	/	/	/
$(\gamma_t, \hat{\gamma}_r, \beta)$	0.0620	13.7340	33.1284	/	/	/
$(\gamma_t, \gamma_r, \hat{\beta})$	0.0644	11.8531	31.8089	17.1833	18.1860	2.7926
$(\hat{\gamma}_t, \hat{\gamma}_r, \hat{\beta})$	0.0684	12.4663	33.6443	/	/	/

From Table 3.4 and Table 3.5, we draw the following conclusions:

1. EKF, UKF, and FastSLAM are dramatically affected by the input noise variances, especially the observation noise variances. These filters are all based on the framework of the Kalman filter (KF); the covariance matrices of noise are

usually fixed during updates. Therefore, the navigation accuracy can only be ensured under the complete and exact information of the noise distributions.

2. For SGO and BA SLAM, the observation noise variances are involved as the coefficients of constraints, and they are normally regarded as constants in the optimization process. As a result, inaccurate noise variances will lead to false cost function, and subsequently diverged estimations.
3. The noise priors are estimated adaptively in online SVBI, and thus it is generally robust to different input variances.

Furthermore, we test Adaptive EKF [199], Adaptive UKF [188], SGO with Huber loss function [86], and BA SLAM with Huber loss function [86]. The first two algorithms are both based on Sage-Husa noise statistics estimator. However, different from the recurrent estimation of noise variances in [188, 199], at every time instant, we introduce the input noise priors to replace the previous noise variances as more exact information during updates. In the latter two, we adopt the Huber loss function, which is robust and so the cost for large residuals can be reduced. The resulting estimation errors and computation time are presented in Table 3.6 and Table 3.7.

TABLE 3.6: Average RMSEs of Adaptive EKF, Adaptive UKF, SGO with Huber's loss, and BA SLAM with Huber loss for different input noise variances.

Input Noise Variances	Adaptive EKF		Adaptive UKF	
	Position [m]	Angle [rad]	Position [m]	Angle [rad]
$(\gamma_t, \gamma_r, \beta)$	0.1439	0.4101e-3	0.1390	0.3925e-3
$(\hat{\gamma}_t, \gamma_r, \beta)$	0.1442	0.4402e-3	0.1390	0.3934e-3
$(\gamma_t, \hat{\gamma}_r, \beta)$	0.1439	0.4102e-3	0.1392	0.3930e-3
$(\gamma_t, \gamma_r, \hat{\beta})$	0.1911	0.7231e-3	0.1797	0.7301e-3
$(\hat{\gamma}_t, \hat{\gamma}_r, \hat{\beta})$	0.2073	0.7756e-3	0.1924	0.8023e-3
Input Noise Variances	SGO with Huber's loss		BA SLAM with Huber's loss	
	Position [m]	Angle [rad]	Position [m]	Angle [rad]
$(\gamma_t, \gamma_r, \beta)$	0.2107	0.0007	0.1086	0.2689e-3
$(\hat{\gamma}_t, \gamma_r, \beta)$	/	/	/	/
$(\gamma_t, \hat{\gamma}_r, \beta)$	/	/	/	/
$(\gamma_t, \gamma_r, \hat{\beta})$	0.4863	0.0017	0.3011	0.6122e-3
$(\hat{\gamma}_t, \hat{\gamma}_r, \hat{\beta})$	/	/	/	/

TABLE 3.7: Average run time per frame for Adaptive EKF, Adaptive UKF, SGO with Huber loss, and BA SLAM with Huber loss under different input noise variances.

Input Noise Variances	Adaptive EKF [s]	Adaptive UKF [s]	SGO with Huber's loss [s]	BA SLAM with Huber's loss [s]
$(\gamma_t, \gamma_r, \beta)$	0.1864	15.7207	40.0183	76.3381
$(\hat{\gamma}_t, \gamma_r, \beta)$	0.2850	16.1336	/	/
$(\gamma_t, \hat{\gamma}_r, \beta)$	0.1861	15.6523	/	/
$(\gamma_t, \gamma_r, \hat{\beta})$	0.1878	16.0227	52.6501	159.0747
$(\hat{\gamma}_t, \hat{\gamma}_r, \hat{\beta})$	0.1882	16.5099	/	/

From Table 3.6 and Table 3.7, we can observe that although adaptive correction and robust kernels are introduced, the performance of Adaptive EKF, Adaptive UKF, SGO with Huber loss, and BA SLAM with Huber's loss are not improved. That is possibly because the adaptive estimation schemes have no guaranteed convergence; Huber loss is less sensitive to large errors but can not detect and recover true noise variances effectively.

3.5.4 Different Numbers of Observations per Frame

The analysis in [168] concluded that increasing the number of features is usually more beneficial than the number of frames to improve the performance of vSLAM. As a result, algorithms equipped with a low complexity regarding the number of features typically gain popularity in this field. For example, BA SLAM, whose computational complexity is linear with the feature number, is used more frequently than the filtering approaches (i.e., EKF and UKF), whose complexity is cubic. In this section, in order to check how the computational cost changes with respect to the feature number, we change the number of observations per frame and then monitor the run time, CPU usage, memory footprint. The value of (a, c, g) is set to be $(0.1, 0.1, 1)$, and all the algorithms are provided with the true noise variances during tests. Figure 3.2 displays how the run time, CPU utilization, and memory footprint changes as function of the number of observations per frame.

We can tell from Figure 3.2 that:

(A) Log-log scale plot of average run time per frame. (B) Log-log scale plot of average CPU usage. (C) Log-log scale plot of average memory footprint.

FIGURE 3.2: Average run time, CPU usage, and memory footprint of of EKF, UKF, FastSLAM, SGO, BA SLAM and online SVBI for different number of observations per frame.

1. The time complexities calculated from Figure 3.2a are listed as below, where N is the number of observations per frame. FastSLAM has the lowest com-

Algorithm	EKF	UKF	FastSLAM	SGO	BA SLAM	Online SVBI
Time Complexity	$\Theta(N^{2.37})$	$\Theta(N^{2.29})$	$\Theta(N^{0.98})$	$\Theta(N^{2.01})$	$\Theta(N^{2.08})$	$\Theta(N^{1.50})$

plexity due to its linear computational complexity and non-iterative scheme. Although SGO, BA SLAM and online SVBI have linear computational complexity with the number of observations per frame, their time complexity is higher since all of them are iterative approaches. The complexity of EKF and UKF are the highest because of their cubic computational complexity.

2. EKF and UKF require much CPU usage, and reach the system limits even for relatively few observations per frame. Below 600 observations per frame, SGO and BA SLAM have low CPU time, however, the CPU time increases rapidly for increasing number of observations beyond 600. Online SVBI and FastSLAM have low CPU time even for more than 6000 observations per frame.
3. We calculate the space complexities of all the algorithms and express them as functions of the number of observations. The results are presented as follows. Online SVBI possesses the lowest memory usage, while FastSLAM requires

Algorithm	EKF	UKF	FastSLAM	SGO	BA SLAM	Online SVBI
Space Complexity	$\Omega(N^{1.73})$	$\Omega(N^{1.74})$	$\Omega(N^{0.46})$	$\Omega(N^{0.80})$	$\Omega(N^{1.12})$	$\Omega(N^{1.15})$

the most because of its large amount of particles. However, FastSLAM has the lowest empirical space complexity, whereas EKF and UKF have a square one. The complexities of BA SLAM, SGO, and Online SVBI are all close to linear, which facilitates their applications in limited resources.

3.6 Experimental Results

To evaluate the proposed online SVBI, we develop our visual SLAM framework based on ORB-SLAM2 and benchmark its performance on various real-life datasets. This section starts with a brief introduction to the system structure and then presents the experimental results tested in several cases.

3.6.1 Implementation

ORB-SLAM2 is a complete 3D feature-based SLAM system for stereo, RGB-D, and monocular cameras. It is the state-of-the-art visual SLAM system and has achieved extraordinary results on KITTI [49], TUM [169], and EuRoc [17] datasets. In ORB-SLAM2, three parallel threads are utilized to process the input images: tracking, local mapping, and loop closing.

Specifically, the tracking thread is performed with every input image to optimize the current pose. The local mapping thread accordingly updates the set of covisible keyframes and local map points when the insertion of a new keyframe is detected. While in the loop closing thread, a pose-graph is optimized to correct the trajectory drift. A full Bundle Adjustment is followed to achieve the optimal solution further.

All the optimizations performed in the above three threads rely on BA, which has a Levenberg-Marquard implementation in g2o [86]. The tracking thread employs “Motion-only BA”, whereas the local mapping and loop closing uses the “Local BA” and “Full BA”, respectively. “Motion-only BA” solves the online estimation of the current frame, while the other two aim at a batch optimization of multiple frames and a local map. Therefore, we apply the proposed SVBI to substitute for “Motion-only BA” in the tracking thread. We call the resulting modified SLAM framework as ORB_VB+SLAM.

FIGURE 3.3: Tracking and local mapping threads in ORB-SLAM2 systems.

Our objective is to benchmark the performance of SVBI through evaluating the modified system ORB_VB+SLAM. Since the accumulated drift can be corrected by loop closure (see [121]), we disable the loop closing thread in both ORB-SLAM2 and ORB_VB+SLAM to eliminate the effect from loop closure. Here by “ORB-SLAM2*” we refer to the system with only two threads. The main components of tracking and local mapping are shown in Figure 3.3, where ORB-SLAM2* implements “pose optimization” with motion-only BA, whereas ORB_VB+SLAM applies online SVBI.

We test both systems in monocular, stereo, and RGB-D modes. Following [121, 122], we measure the accuracy of each trial by the mean and RMSE of absolute translation error (ATE). For all the datasets, each sequence is repeated 10 times, and we report the average. We evaluate the algorithms on three datasets: KITTI [49], TUM [169], and EuRoC [17]. The frequencies are 10Hz, 20 Hz, and 30 Hz, respectively. Both ORB-SLAM2* and ORB_VB+SLAM achieve real-time speed. The average computational cost is shown in Table 3.8:

TABLE 3.8: The average computational cost per frame of ORB-SLAM2* and ORB_VB+SLAM systems on KITTI, EuRoC and TUM dataset in the stereo, RGBD and monocular mode.

Case	ORB-SLAM2*	ORB_VB+SLAM
Stereo KITTI	0.08 s	0.10 s
Stereo EuRoC	0.057 s	0.049 s
RGBD TUM	0.026 s	0.034 s
Monocular KITTI	0.04 s	0.04 s
Monocular EuRoC	0.04 s	0.04 s
Monocular TUM	0.02 s	0.03 s

3.6.2 Stereo Cases

In stereo vision [54, 83, 124], two horizontally displaced cameras are employed to obtain two differing views on a scene. By comparing the two images, scale becomes directly observable in the form of a disparity map, which encodes the horizontal separation of corresponding points. Therefore, stereo vision does not suffer from “scale drift” [165, 167].

In ORB-SLAM2, textured ORB features [149] are extracted from both images. After matching features seen in both frames with sufficient disparity, we generate stereo and monocular keypoints. In this work, stereo cases are evaluated on two popular datasets, i.e., KITTI and EuRoC. The results of both systems are presented in Table 3.9 and Table 3.10. Besides the ATE.Mean and ATE.RMSE, we also present the relative error of ORB_VB+SLAM in percentage compared to ORB+SLAM2* in both tables (bold numbers in brackets). They are calculated as $\text{relative error} = \frac{\text{the error of ORB+SLAM2}^*}{\text{the error of ORB_VB+SLAM}}$. Furthermore, the average relative error is attached at the bottom line of the table.

TABLE 3.9: Average accuracy of ORB-SLAM2* and ORB_VB+SLAM systems on KITTI dataset in the stereo mode.

Sequence	ORB-SLAM2*		ORB_VB+SLAM	
	ATE.Mean[m]	ATE.RMSE[m]	ATE.Mean[m]	ATE.RMSE[m]
00	3.57	3.87	3.35 (94%)	3.67 (95%)
01	4.63	5.10	3.47 (75%)	3.82 (75%)
02	7.64	8.84	7.41 (97%)	8.57 (97%)
03	0.28	0.30	0.25 (90%)	0.28 (91%)
04	0.18	0.20	0.19 (106%)	0.21 (106%)
05	1.83	1.99	1.70 (93%)	1.85 (93%)
06	1.81	2.03	1.50 (83%)	1.66 (82%)
07	1.06	1.27	1.01 (96%)	1.22 (96%)
08	2.63	3.12	2.63 (100%)	3.18 (102%)
09	2.28	2.85	2.21 (97%)	2.74 (96%)
10	0.89	1.00	0.85 (96%)	0.93 (93%)
Average	-	-	(93%)	(93%)

From Table 3.9 and Table 3.10, we can see that the modified ORB_VB+SLAM system has an improved accuracy in contrast to ORB+SLAM2*, which demonstrates the advantages of online SVBI. Both systems get lost in EuRoC V2_03_difficult

TABLE 3.10: Average accuracy of ORB-SLAM2* and ORB_VB+SLAM systems on EuRoC dataset in the stereo mode.

Sequence	ORB-SLAM2*		ORB_VB+SLAM	
	ATE.Mean[m]	ATE.RMSE[m]	ATE.Mean[m]	ATE.RMSE[m]
V1_01_easy	0.081	0.087	0.081 (100%)	0.087 (100%)
V1_02_medium	0.089	0.094	0.076 (85%)	0.079 (84%)
V1_03_difficult	0.273	0.297	0.295 (108%)	0.315 (106%)
V2_01_easy	0.065	0.067	0.063 (97%)	0.064 (96%)
V2_02_medium	0.098	0.108	0.093 (95%)	0.094 (96%)
V2_03_difficult	Lost		Lost	
MH_01_easy	0.030	0.036	0.029 (96%)	0.033 (91%)
MH_02_easy	0.030	0.037	0.029 (95%)	0.035 (94%)
MH_03_medium	0.032	0.036	0.031 (97%)	0.035 (97%)
MH_04_difficult	0.084	0.091	0.076 (91%)	0.084 (92%)
MH_05_difficult	0.086	0.096	0.071 (82%)	0.081 (84%)
Average	-	-	(95%)	(94%)

due to severe motion blur. It is worth mentioning that ORB-SLAM2* system occasionally gets lost on sequence V1_02_medium, V2_01_easy, and V2_02_medium in EuRoC dataset. ORB_VB+SLAM seems to be more robust and misses either none or fewer frames on those three sequences.

We show several examples of the estimated trajectories by the two systems on KITTI and EuRoC datasets in Figure 3.4. To display the endpoints of the predicted paths clearly, we add the zoomed up subplots. As indicated in Figure 3.4, ORB_VB+SLAM produces more accurate estimations in the selected examples.

3.6.3 RGB-D Cases

In the past years, the emergence of RGB-D sensors open novel ways in the field of SLAM [97, 155, 185]. Instead of leveraging two gray images, RGB-D determines the scale information directly from the input depth data. Therefore, RGB-D can handle surfaces with little or no textures in a reliable manner, for which stereo SLAM is generally not applicable [114, 212]. However, RGB-D vision is typically limited to small indoor workspaces due to the measurement depth range [55].

We evaluate ORB-SLAM2* and ORB_VB+SLAM on the TUM dataset in the RGB-D working mode. Sequences in this dataset are divided into several groups

FIGURE 3.4: Estimated trajectories of ORB-SLAM2* and ORB_VB+SLAM on KITTI sequence 06, 10 and EuRoC sequence V1_02_medium, MH_05_difficult in the stereo mode.

TABLE 3.11: Average accuracy of ORB-SLAM2* and ORB_VB+SLAM systems on TUM dataset in the RGB-D mode.

Sequence	ORB-SLAM2*		ORB_VB+SLAM	
	ATE.Mean[m]	ATE.RMSE[m]	ATE.Mean[m]	ATE.RMSE[m]
Fr1desk	Lost	Lost	Lost	Lost
Fr1desk2	Lost	Lost	Lost	Lost
Fr1room	0.099	0.106	0.011 (110%)	0.115 (109%)
Fr2desk	0.010	0.011	0.009 (94%)	0.010 (93%)
Fr2xyz	0.003	0.003	0.003 (100%)	0.003 (100%)
Fr3office	0.011	0.012	0.010 (94%)	0.011 (94%)
Fr3nst	0.019	0.021	0.015 (78%)	0.016 (78%)
Average	-	-	(95%)	(95%)

FIGURE 3.5: Estimated trajectories of ORB-SLAM2* and ORB.VB+SLAM on TUM sequence Fr3office and Fr3nst in the RGB-D mode.

according to texture, illumination, and structure conditions. For coherence, we choose the same seven videos in [122] for testing and present the accuracy results in Table 3.11. Both systems achieved high accuracy on most sequences, except Fr1desk and Fr1desk2, where both systems got lost. The estimated trajectories of ORB.VB+SLAM are overall more closer to the ground truth than the ones estimated by ORB-SLAM2*. Figure 3.5 shows the estimated trajectories of sequence Fr3office and Fr3nst in one trial. Both systems generated close estimations to the ground truth. However, from the inset subplots of the end parts, we can still observe that the estimations provided by ORB.VB+SLAM are more consistent.

3.6.4 Monocular Cases

Unlike stereo and RGB-D vision, monocular SLAM relies only on a single camera. As a result, the absolute scale of the map cannot be determined directly, and additional procedures are required [45, 123, 172]. Most representative monocular SLAM works are keyframe-based [38, 121, 122]. In ORB-SLAM2, the system arbitrarily define scale as the median scene depth at the initialization step. This initialization relies on a heuristic model selection, i.e., a homograph for a planar scene, or a fundamental matrix for a nonplanar scene. With the initialized map and scene scale, the first relative pose is calculated, and the system then begins to operate on a single camera frame by frame smoothly.

TABLE 3.12: Average accuracy of ORB-SLAM2* and ORB_VB+SLAM systems on KITTI dataset in the monocular mode.

Sequence	ORB-SLAM2*		ORB_VB+SLAM	
	ATE.Mean[m]	ATE.RMSE[m]	ATE.Mean[m]	ATE.RMSE[m]
00	69.12	77.74	60.82 (88%)	69.96 (90%)
01	Lost		Lost	
02	25.69	30.06	29.80 (116%)	34.26 (118%)
03	0.81	1.05	0.89 (110%)	1.20 (114%)
04	1.05	1.18	0.97 (92%)	1.11 (94%)
05	36.25	42.86	36.61 (101%)	43.29 (101%)
06	45.45	51.54	43.18 (95%)	48.96 (95%)
07	14.79	18.06	15.38 (104%)	18.78 (104%)
08	41.63	53.15	39.13 (94%)	50.49 (95%)
09	Lost		Lost	
10	6.12	7.46	5.88 (96%)	7.16 (96%)
Average	-	-	(100%)	(100%)

We test ORB-SLAM2* and ORB_VB+SLAM in the monocular mode on KITTI, TUM, and EuRoC, and present their results in Table 3.12, Table 3.13 and Table 3.14, respectively. We can observe that both systems have a satisfactory performance on the indoor EuRoC and TUM, but behave poorly on KITTI sequence 00, 02, 05, 06, 08. As mentioned in [121], KITTI is an extremely challenging dataset for the monocular task because of the high speed, fast rotations, and areas with lots of foliage. We disable the loop closing thread in both systems aforementioned in Section 3.6.1, which might deteriorate the performance further. In all, ORB_VB+SLAM does not achieve any obvious accuracy improvements regarding monocular KITTI.

By contrast, ORB_VB+SLAM shows quite obvious advantages on both EuRoC and TUM in the monocular working mode. The accuracy is improved clearly on all the trackable test sequences (see Table 3.13 and Table 3.14). It is worth noticing that ORB-SLAM2* generated unusual solutions on sequence EuRoC sequence V1_02_medium, MH_05_difficult and TUM Fr3office possibly due to a poor initialization. For a fair comparison, the average accuracy of those sequences is calculated after having already removed the failed trials and indicated with an asterisk in Table 3.13 and Table 3.14.

The estimated trajectories of ORB-SLAM2* and ORB_VB+SLAM are shown in

TABLE 3.13: Average accuracy of ORB-SLAM2* and ORB_VB+SLAM systems on EuRoC dataset in the monocular mode.

Sequence	ORB-SLAM2*		ORB_VB+SLAM	
	ATE.Mean[m]	ATE.RMSE[m]	ATE.Mean[m]	ATE.RMSE[m]
V1_01_easy	0.093	0.095	0.092 (99%)	0.094 (99%)
V1_02_medium	0.091*	0.103*	0.074 (81%)	0.080 (78%)
V1_03_difficult		Lost		Lost
V2_01_easy		Lost		Lost
V2_02_medium	0.070	0.074	0.067 (97%)	0.072 (97%)
V2_03_difficult		Lost		Lost
MH_01_easy	0.040	0.045	0.038 (95%)	0.036 (97%)
MH_02_easy	0.031	0.037	0.029 (95%)	0.035 (95%)
MH_03_medium	0.035	0.038	0.034 (97%)	0.037 (98%)
MH_04_difficult	0.061	0.068	0.054 (88%)	0.059 (87%)
MH_05_difficult	0.053*	0.060*	0.041 (77%)	0.046 (76%)
Average	-	-	(91%)	(91%)

TABLE 3.14: Average accuracy of ORB-SLAM2* and ORB_VB+SLAM systems on TUM dataset in the monocular mode.

Sequence	ORB-SLAM2*		ORB_VB+SLAM	
	ATE.Mean[m]	ATE.RMSE[m]	ATE.Mean[m]	ATE.RMSE[m]
Fr1desk	0.016	0.019	0.015 (99%)	0.018 (97%)
Fr1desk2		Lost		Lost
Fr1room		Lost		Lost
Fr2desk	0.012	0.014	0.011 (96%)	0.012 (85%)
Fr2xyz	0.002	0.003	0.002^a (91%)	0.003^a (91%)
Fr3office	0.027*	0.033*	0.015 (57%)	0.019 (58%)
Fr3nst	0.022	0.025	0.019 (87%)	0.022 (87%)
Average	-	-	(86%)	(84%)

^a Here the average ATE.Mean and ATE.RMSE of ORB_VB+SLAM on Fr2xyz are less than but are rounded up to 0.002 and 0.003, respectively.

Figure 3.6. The plots provided for the selected sequences demonstrate the superiority of ORB_VB+SLAM distinctly.

FIGURE 3.6: Estimated trajectories of ORB-SLAM2* and ORB_VB+SLAM on EuRoC sequence V2.02_medium, MH.04_difficult, and TUM sequence Fr3office and Fr3nst in the monocular mode.

3.7 Conclusion

In this chapter, we proposed a linear-complexity SVBI for solving online SLAM problems. To enhance the robustness, we updated the hyperparameters rather than fix them as constant values; to improve the computational efficiency, we have set the covariance matrix to be diagonal to vectorize the practical implementation. We first validated the online SVBI method on synthetic data and compared it against other prevalent methods under three scenarios: different noise levels, different noise variance inputs, and different numbers of features. The simulation results have

shown that online SVBI has improved accuracy, a linear complexity, and enhanced robustness.

Furthermore, by applying SVBI to the state-of-the-art ORB-SLAM2, we have created the modified ORB_VB+SLAM system. We have then tested ORB_VB+SLAM on KITTI, EuRoC, and TUM datasets with stereo, RGB-D, and monocular interfaces. Experimental results showed that ORB_VB+SLAM system has an average accuracy improvement of 7% when compared to ORB-SLAM2*.

Note that though this inference algorithm was built upon the vSLAM model, it is extensible to other point-based SLAM estimation problems. We believe that this approach will significantly help to improve the online estimation of single or multi-sensor SLAM systems, especially in challenging scenarios where the number of features is large. In the future, we will deploy it on real robotic platforms to validate its performance under specific tasks.

Chapter 4

Robust Linear-complexity Stochastic Variational Bayesian Inference Method for Full SLAM Problems

We proposed an online solution for the SLAM problem in Chapter 3. Apart from the online formulation, the solution can also be addressed via an offline form, which involves estimating the full trajectory. Both online and offline are of equal importance in practical SLAM systems. Since offline algorithms jointly analyze all frames, the accuracy is usually higher than online solutions. In this chapter, we move onto the full SLAM problem and derive an offline solution: offline SVBI. Compared with the state-of-the-art graph-based SLAM, offline SVBI is robust to inaccurate noise priors. Furthermore, its computational complexity is linear with the number of frames, while that of graph-based SLAM is cubical. As a consequence, offline SVBI has enhanced robustness and scalability.

4.1 Introduction

In a SLAM system, the vehicle tours around the environment, and maps the surroundings by collecting and analyzing sensor data. This process involves the computational problem of estimating the vehicle pose and constructing or updating the

environmental map. The literature distinguishes two solutions: online and offline. Online SLAM attempts to recover only the most recent robot location and map points. Algorithms that fall into this category include EKF [4, 21, 62, 197], UKF [61, 157], SEIF [176, 183], FastSLAM [52, 116, 117], RB-PHD Filter [120], and their variants [76, 133, 190, 207, 209] (see Section 3.1). Conversely, the full SLAM addresses the estimation of the full trajectory and the global map. It requires extensive batch processing to cope with perceptual ambiguities [158, 175]. Compared to the online description, the estimation is generally more accurate but needs more computational cost. Most smoothing algorithms [33, 104, 173] rely on nonlinear least square error (NLLS) minimization [174]. Among them, the state-of-the-art is graph-based SLAM [31, 51, 84, 87]. It involves constructing a graphical network, where landmarks and robot poses are denoted as nodes; the measurements are represented as edges that constrain the connected nodes. The approximation is then acquired via finding a configuration of the nodes that is maximally consistent with the constraints. This graphical formulation greatly facilitates the calculation and avoids complex constrained optimization.

Despite the wide applicability, graph-based SLAM also has limitations. For example, it is at vast expense for solving large SLAM problems due to the cubical complexity w.r.t. the number of frames. In addition, the noise priors are all fixed during the update. However, there is no guarantee that the priors are the ground truth, and the deviation will further deteriorate the performance [188, 208]. We proposed the online SVBI in Chapter 3. It has achieved improved performance on both synthetic and real data. In this chapter, we focus on offering its offline version (offline SVBI) for full SLAM problems.

Similarly, we formulate the full SLAM problem into a posterior estimation and employ a variational Bayesian inference framework to approximate. However, the map points here are global and shared across time. We set the variational distribution to be Gaussian. The uncertainty matrix is tri-diagonal, where the antidiagonals describe the internal links between neighboring frames. As a result, the computational complexity is linear w.r.t. the number of features and frames. Moreover, the estimated uncertainty might be of great use in feature tracking [25, 143], motion planning [16], path-constrained motion [57], and maintaining reasonable data size [64]. Furthermore, the noise parameters of the system are estimated jointly

together with the posterior. As a consequence, offline SVBI is robust to inaccurate noise statistics.

4.2 Related Work

In this section, we review the existing algorithms for solving the full SLAM problems. Approaches in this category [31, 33, 51, 84, 87, 104, 173] are mostly based on least square error minimization techniques. Among them, the graph-based SLAM [31, 51, 84, 87] is applied most widely. It draws the intuition from the graphical representation, which was first mentioned in [104]. This representation poses the SLAM problem as a graph, where nodes represent the vehicle poses and map point locations. Sensor measurements are encoded into edges that constrain the connected nodes. These constraints might be contradictory since they are affected by noise. Therefore, graph-based SLAM also associates the noise priors with the corresponding arcs. These priors are fixed during the optimization. Graph-based SLAM avoids the sophisticated constrained optimization and determines the solutions via relaxing the constraints, that is, solving a large error minimization problem. This minimization is unwieldy and computationally demanding if we solve it directly. To overcome this difficulty, sparse structures were introduced and significantly reduced the computational burden. [176, 183]. Currently, people have explored a number of algorithms to solve the formulated minimization problem efficiently, such as Gauss-Newton [127, 192], Levenberg–Marquardt (LM) [28, 162], Powell’s ‘dogleg’ (DL) [88, 204].

The complexity of graph-based SLAM is $\mathcal{O}(K^G + K^3M^3)$, where K is the total number of frames, M is the dimension of the vehicle pose, and the global map vector is G dimensional. Since it is linear with the number of features, graph-based SLAM is scalable to scenarios where the number of features is large. However, the complexity is a cubic function of the number of frames K . Therefore, graph-based SLAM is still cumbersome for scenarios where the number of frames is large. To relieve the computational burden, one performs batch optimization over selected images from the live stream, such as particular spatially distributed keyframes [82, 91, 121, 122, 139, 140], or frames in a sliding window [91, 119, 132, 139, 140, 166].

By taking a Bayesian probabilistic perspective, graph-based SLAM is a maximum a posteriori probability (MAP) problem [8]. The MAP seeks for the mode of the posterior and outputs a point estimate. Since there is no associated uncertainty with the obtained estimations, uncertainty recovery sometimes is required in tasks such as [16, 25, 57, 64, 143].

The statistical characteristics of noise variances are usually time-varying and change easily due to internal and external uncertainties [188, 208]. However, graph-based SLAM systems [38, 39, 121, 122] fix the noise variances during updates, causing potential performance degradation. To address this issue, we propose a linear-complexity stochastic variational Bayesian inference for full SLAM problems (offline SVBI). This method approximates the posterior distribution of the full SLAM problem by stochastic variational Bayesian inference. In addition, the noise covariances are jointly updated. The complexity of offline SVBI is $\mathcal{O}(KN)$, where K is the number of frames, and N is the dimension of each observation vector. Therefore, offline SVBI is linear w.r.t. the number of features and frames, and more scalable than graph-based optimization. To the best of our knowledge, only two variational approaches were proposed for full SLAM problems. The VBEM approach proposed in [41] is limited to radar-based SLAM and considers the data association task. The measurements in VBEM are assumed to be generated by an inhomogeneous Poisson process. Their spatial distribution is described as a Gaussian mixture model (GMM), where each component corresponds to a landmark. The priors of the GMM parameters are assumed to be Gamma or Inverse-Wishart distributions to be conjugate to a Gaussian likelihood. Consequently, VBEM is only applicable to certain situations. An amortized deep variational Bayes filter was proposed in [113] to solve full SLAM problems. This filter is augmented with a global latent variable representing the environmental map. Two neural networks are employed to represent the motion and observation processes. The estimation of vehicle poses and map points is then formulated as posterior inference in the resulting generative model. After training the networks through a heuristic, particle-based scheme, the posterior can be approximated via importance-sampling. Although the computational cost has been amortized, the training process is still costly. Moreover, the observation network needs to be carefully chosen so that the resulting posterior inference can be related to the SLAM problem. What's more, the shared weights and biases across data limit the capacity of the variational family.

FIGURE 4.1: Graphical model of the full SLAM problem.

4.3 Bayesian Full SLAM Problem

This section presents a Bayesian formulation of the full SLAM problem. Without loss of generality, we formulate this framework based on feature-based vSLAM, but the derived model can be easily extended to other point-based SLAM systems.

We depict the dynamic Bayesian network of the full SLAM problem in Figure 4.1. The blue nodes are the observed data, and green ones represent the hidden variables. $\mathbf{x}_0$ is the starting position, which typically is the origin of the world coordinate. Assuming the total number of tracked frames is K , the vehicle's trajectory from time 1 to K is then referred to as $\mathbf{x}_{1:K} = \{\mathbf{x}_1, \dots, \mathbf{x}_k, \dots, \mathbf{x}_K\}$. Accordingly, the corresponding measurements are expressed as $\mathbf{z}_{1:K} = \{\mathbf{z}_1, \dots, \mathbf{z}_k, \dots, \mathbf{z}_K\}$. We denote the environmental map by a global latent variable $\mathbf{m}$. The motion and observation process are subject to uncertainty. We refer to the noise as ϵ_k and δ_k , respectively. In SLAM, ϵ_k and δ_k are usually assumed to be white noise. If the dimensions of $\mathbf{x}_k$ and $\mathbf{z}_k$ are M and N , respectively, we set ϵ_k to follow the distribution $\mathcal{N}(0; (\alpha_k \circ \mathbf{I}_{MM})^{-1})$ and $\delta_k \sim \mathcal{N}(0; (\beta_k \circ \mathbf{I}_{NN})^{-1})$. Here α_k and β_k are two vectors, $\mathbf{I}_{MM}$ and $\mathbf{I}_{NN}$ are square identity matrices, the dimensions of which are M and N , respectively. As such, the uncertainties are encoded into series of vectors $\alpha_{1:K}$ and $\beta_{1:K}$, respectively (see Figure 4.1). The Bayesian formulation of

the full SLAM problem is given by:

$$p(\mathbf{x}_{1:K}, \mathbf{z}_{1:K} | \mathbf{x}_0, \mathbf{m}, \boldsymbol{\alpha}_{1:K}, \boldsymbol{\beta}_{1:K}) \propto p(\mathbf{x}_0) \prod_{k=1}^K p(\mathbf{x}_k | \mathbf{x}_{k-1}, \boldsymbol{\alpha}_k) p(\mathbf{z}_k | \mathbf{m}, \mathbf{x}_k, \boldsymbol{\beta}_k). \quad (4.1)$$

In (4.1), the evolution process between consecutive frames $p(\mathbf{x}_k | \mathbf{x}_{k-1}, \boldsymbol{\alpha}_k)$ is known as the “motion” model, and $p(\mathbf{z}_k | \mathbf{x}_k, \boldsymbol{\beta}_k)$ is typically referred to as the “observation” model. We assume that the control vectors (i.e., translation speed, angular speed) and data association have already been determined, and skip their notations throughout the rest. Consequently, the motion and observation models can be described as follows:

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}) + \boldsymbol{\epsilon}_k, \quad (4.2a)$$

$$\mathbf{z}_k = h(\mathbf{x}_k, \mathbf{m}) + \boldsymbol{\delta}_k, \quad (4.2b)$$

where f and h are two nonlinear functions representing the motion and observation processes, respectively.

Following Chapter 3, we indicate the fixed world coordinate by “ W ” and denote the moving camera frames as “ R ”. The vehicle pose $\mathbf{x}_k$ at time k is expressed as a 6D vector in the fixed world frame “ W ”. The former three elements of $\mathbf{x}_k$ are the 3D metric position $\mathbf{r}_k^W$, and the latter are the Euler angles (i.e., roll, pitch, yaw) $\mathbf{e}_k^{WR}$. The superscripts denote the coordinates that the quantities are expressed relative to. During the updates, we often convert $\mathbf{e}_k^{WR}$ into a 3×3 rotation matrix Rot_k^{WR} or reversely. Assuming the time interval between neighboring frames is constant and defined as Δt , f can be further expanded as:

$$f(\mathbf{x}_{k-1}) = \begin{pmatrix} \mathbf{r}_k^W \\ \mathbf{e}_k^{WR} \end{pmatrix} = \begin{pmatrix} \mathbf{r}_{k-1}^W + \mathbf{v}_k^W \Delta t \\ R2e(Rot_{k-1}^{WR} e2R(\boldsymbol{\omega}_k^R \Delta t)) \end{pmatrix}, \quad (4.3)$$

where $\mathbf{v}^W$ is a 3D velocity vector, and $\boldsymbol{\omega}^R$ is the 3D angular velocity.

The global map $\mathbf{m}$ is initialized at system start-up, evolves continuously and dynamically, and persists until the operation ends. It consists of a series of features extracted from camera images at salient keypoint locations. Each feature corresponds to a 3D position vector $[\mathbf{l}_i^W]$ expressed in the world frame. The observations $\mathbf{z}_{1:K}$ are the image coordinates of features. We can further classify the features as

monocular and stereo [121, 122]. Take the i^{th} feature $[\mathbf{l}_i^W]$ as an example, it is defined by $[u_{iL}, v_{iL}, u_{iR}]$ if it is stereo, and $[u_{iL}, v_{iL}]$ when monocular. $[u_{iL}, v_{iL}]$ are the coordinates of the left image, and u_{iR} is the horizontal coordinate on the right image. In stereo and RGB-D vision, monocular keypoints exist when a stereo match could not be found, or an invalid depth value occurs. In particular, if we denote the baseline between two lenses as b , and f_x as the horizontal focal length, the right coordinate u_{iR} is a virtually defined as:

$$u_{iR} = u_{iL} - \frac{f_x b}{d}, \quad (4.4)$$

where d is the depth value of the keypoint.

With a perspective camera, the coordinates of $[\mathbf{l}_i^W]$ in frame $\mathbf{x}_k$ can be calculated by the standard pinhole model [56]:

$$[h(\mathbf{x}_k, \mathbf{m})]_i = h(\mathbf{x}_k, \mathbf{l}_i^W) = \begin{bmatrix} u_{iL} \\ v_{iL} \\ u_{iR} \end{bmatrix} = \begin{bmatrix} u_0 + f_x \frac{X_{ik}^R}{Z_{ik}^R} \\ v_0 + f_y \frac{Y_{ik}^R}{Z_{ik}^R} \\ u_0 + f_x \frac{X_{ik}^R}{Z_{ik}^R} - b \end{bmatrix}, \quad (4.5)$$

where (u_0, v_0) is the principal point, f_y is the vertical focal length, together with f_x defined previously, all known from the camera calibration.

The monocular measuring process is expressed as:

$$[h(\mathbf{x}_k, \mathbf{m})]_i = h(\mathbf{x}_k, \mathbf{l}_i^W) = \begin{bmatrix} u_{iL} \\ v_{iL} \end{bmatrix} = \begin{bmatrix} u_0 + f_x \frac{X_{ik}^R}{Z_{ik}^R} \\ v_0 + f_y \frac{Y_{ik}^R}{Z_{ik}^R} \end{bmatrix}. \quad (4.6)$$

In (4.5) and (4.6), $[X_{ik}^R, Y_{ik}^R, Z_{ik}^R]$ is the relative position of $\mathbf{l}_i^W$ in the k^{th} image:

$$[X_i^R, Y_i^R, Z_i^R]^T = Rot_k^{RW}(\mathbf{l}_i^W - \mathbf{r}_k^W). \quad (4.7)$$

With available motion and observation functions defined in (4.3), (4.5) and (4.6), the distributions of the motion and observation models in (4.1) can be further

expressed as:

$$p(\mathbf{x}_k | \mathbf{x}_{k-1}, \boldsymbol{\alpha}_k) \propto \sqrt{\prod_{i=1}^M (\boldsymbol{\alpha}_k)_i} \exp \left\{ -\frac{1}{2} (\mathbf{x}_k - f(\mathbf{x}_{k-1}))^T \text{diag}(\boldsymbol{\alpha}_k) (\mathbf{x}_k - f(\mathbf{x}_{k-1})) \right\}, \quad (4.8a)$$

$$p(\mathbf{z}_k | \mathbf{m}, \mathbf{x}_k, \boldsymbol{\beta}_k) \propto \sqrt{\prod_{j=1}^N (\boldsymbol{\beta}_k)_j} \exp \left\{ -\frac{1}{2} (\mathbf{z}_k - h(\mathbf{x}_k, \mathbf{m}))^T \text{diag}(\boldsymbol{\beta}_k) (\mathbf{z}_k - h(\mathbf{x}_k, \mathbf{m})) \right\}, \quad (4.8b)$$

where $(\boldsymbol{\alpha}_k)_i$ and $(\boldsymbol{\beta}_k)_j$ represent the i^{th} and j^{th} element of $\boldsymbol{\alpha}_k$ and $\boldsymbol{\beta}_k$, respectively.

Existing offline smoothing algorithms [31, 33, 51, 84, 87, 104, 173] fix $\boldsymbol{\alpha}_{1:K}, \boldsymbol{\beta}_{1:K}$ as constants during updates. However, there is no guarantee that the priors are reliable. For example, the priors are determined by the scale factor of ORB features in [121, 122]. In addition, the least-squares minimization is sensitive to the coefficients, that is, the noise parameters. Therefore, the system may suffer from performance degradation or even divergence when the noise priors are not accurate enough. To overcome this challenge, we update $\boldsymbol{\alpha}_{1:K}$ and $\boldsymbol{\beta}_{1:K}$ along with the posterior distribution. The details will be discussed in the next section. As such, the resulting algorithm enhances the robustness.

4.4 Stochastic Variational Bayesian Inference for Full SLAM Problem

In this section, we derive a robust linear-complexity stochastic variational Bayesian inference algorithm for the full SLAM problem presented in Section 4.3. In addition, the computational complexity of this proposed method is discussed.

4.4.1 Stochastic Variational Bayesian Inference

Variational methods [12, 60, 85, 141, 177, 189, 201] have recently gained popularity in the context of graphical models. They are a family of techniques for approximating complex joint distributions, for example, the distribution in (4.1), by a simpler variational distribution.

In the Bayesian paradigm, the SLAM solutions comprise two separate steps:

- Maximize the likelihood $\log p(\mathbf{z}_{1:K}, \mathbf{x}_0 | \mathbf{m}, \boldsymbol{\alpha}_{1:K}, \boldsymbol{\beta}_{1:K})$ to determine the point estimates of the global map $\mathbf{m}$ and the noise prior parameters $\alpha_{1:K}, \beta_{1:K}$.
- Given the estimated $\mathbf{m}, \alpha_{1:K}, \beta_{1:K}$, estimate the posterior distribution of the vehicle trajectory $p(\mathbf{x}_{1:K} | \mathbf{z}_{1:K}, \mathbf{x}_0, \mathbf{m}, \boldsymbol{\alpha}_{1:K}, \boldsymbol{\beta}_{1:K})$.

Since the analytical expression of $p(\mathbf{x}_{1:K} | \mathbf{z}_{1:K}, \mathbf{x}_0, \mathbf{m}, \boldsymbol{\alpha}_{1:K}, \boldsymbol{\beta}_{1:K})$ is intractable, we resort to variational Bayesian methods to approximate this unwieldy posterior distribution. Starting from the marginal probability of $\mathbf{z}_{1:K}$, we have

$$\begin{aligned}
 & \log p(\mathbf{z}_{1:K}, \mathbf{x}_0 | \mathbf{m}, \boldsymbol{\alpha}_{1:K}, \boldsymbol{\beta}_{1:K}) \\
 &= \log \int p(\mathbf{z}_{1:K}, \mathbf{x}_0, \mathbf{x}_{1:K} | \mathbf{m}, \boldsymbol{\alpha}_{1:K}, \boldsymbol{\beta}_{1:K}) d\mathbf{x}_{1:K}, \\
 &= \log \int p(\mathbf{z}_{1:K}, \mathbf{x}_0, \mathbf{x}_{1:K} | \mathbf{m}, \boldsymbol{\alpha}_{1:K}, \boldsymbol{\beta}_{1:K}) \frac{q(\mathbf{x}_{1:K})}{q(\mathbf{x}_{1:K})} d\mathbf{x}_{1:K}, \\
 &\geq \int q(\mathbf{x}_{1:K}) \log \frac{p(\mathbf{z}_{1:K}, \mathbf{x}_0, \mathbf{x}_{1:K} | \mathbf{m}, \boldsymbol{\alpha}_{1:K}, \boldsymbol{\beta}_{1:K})}{q(\mathbf{x}_{1:K})} d\mathbf{x}_{1:K}, \\
 &\triangleq \mathcal{L}.
 \end{aligned} \tag{4.9}$$

Through applying the Jensen's inequality [69], $\mathcal{L}$ in (4.9) is the evidence lower bound (ELBO) of $\log p(\mathbf{z}_{1:K}, \mathbf{x}_0 | \mathbf{m}, \boldsymbol{\alpha}_{1:K}, \boldsymbol{\beta}_{1:K})$. The optimal $q(\mathbf{x}_{1:K})$, along with the point estimates of $\mathbf{m}$ and $\boldsymbol{\alpha}_{1:K}, \boldsymbol{\beta}_{1:K}$, can then be obtained via the tractable maximization of $\mathcal{L}$. Moreover, ELBO equals to the likelihood *iff* $q(\mathbf{x}_{1:K})$ is identical to the true posterior $p(\mathbf{x}_{1:K} | \mathbf{z}_{1:K}, \mathbf{x}_0, \mathbf{m}, \boldsymbol{\alpha}_{1:K}, \boldsymbol{\beta}_{1:K})$.

According to the definition of expectation, $\mathcal{L}$ can also be written as:

$$\begin{aligned}
 \mathcal{L} &= \mathbb{E}_q \left[\log \frac{p(\mathbf{z}_{1:K}, \mathbf{x}_0, \mathbf{x}_{1:K} | \mathbf{m}, \boldsymbol{\alpha}_{1:K}, \boldsymbol{\beta}_{1:K})}{q(\mathbf{x}_{1:K})} \right], \\
 &= \mathbb{E}_q [\log p(\mathbf{z}_{1:K}, \mathbf{x}_0, \mathbf{x}_{1:K} | \mathbf{m}, \boldsymbol{\alpha}_{1:K}, \boldsymbol{\beta}_{1:K})] + \mathcal{H}_q,
 \end{aligned} \tag{4.10}$$

where $\mathcal{H}_q = -\mathbb{E}_q[\log q(\mathbf{x}_{1:K})]$ is the so-called Shannon entropy of $q(\mathbf{x}_{1:K})$ [20].

To find an appropriate approximation q efficiently, in practice, we generally restrict $q(\mathbf{x}_{1:K})$ to be within a certain family [12]. In mean field variational inference [12, 60], $q(\mathbf{x}_{1:K})$ is usually factorized as $\prod_{i=1}^M q(\mathbf{x}_{1:K}^i)$, where the superscript i is the entry index in each $\mathbf{x}_k$ ($k = 1 : K$). In this work, we set $q(\mathbf{x}_{1:K}^i)$ to follow the

Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}^i, J^{i-1})$, where $\boldsymbol{\mu}^i$ is the real mean vector and J^i is its inverse covariance matrix. Obviously, the diagonal entries of J^i are all positive. The off-diagonal entries, for example, entries (k_1, k_2) and (k_2, k_1) in J^i are non-zero iff $x_{k_1}^i$ and $x_{k_2}^i$ are conditionally dependent. For the model presented in Section 4.3, $\mathbf{x}_{1:K}$ is a series of successive images, and we assume x_k^i is only related to x_{k-1}^i and x_{k+1}^i . As a consequence, for every i (i.e., $i = 1 : 6$ here), the precision matrix J^i can be defined as a tri-diagonal matrix, in which the two secondary diagonals are used to describe the links between neighboring vehicle poses.

We employ the gradient ascent method to maximize the ELBO $\mathcal{L}$ in (4.9). Provided both $p(\mathbf{z}_{1:K}, \mathbf{x}_0, \mathbf{x}_{1:K} | \mathbf{m}, \boldsymbol{\alpha}_{1:K}, \boldsymbol{\beta}_{1:K})$ and $q(\mathbf{x}_{1:K})$ are differentiable, we can easily compute all the required gradients: $\nabla_{\nu} \mathbb{E}_q[\log p(\mathbf{z}_{1:K}, \mathbf{x}_0, \mathbf{x}_{1:K} | \mathbf{m}, \boldsymbol{\alpha}_{1:K}, \boldsymbol{\beta}_{1:K})]$ and $\nabla_{\nu} \mathbb{E}_q[\log q(\mathbf{x}_{1:K})]$, where $\nu \in \{\mathbf{x}_{1:K}, \mathbf{m}, \boldsymbol{\alpha}_{1:K}, \boldsymbol{\beta}_{1:K}\}$. However, since $q(\mathbf{x}_{1:K})$ is also parameterized by $\mathbf{x}_{1:K}$, we apply a reparameterization trick [81, 85, 150] to $q(\mathbf{x}_{1:K})$ to calculate the gradients in a more amenable way:

$$\mathbf{x}_{1:K}^i = U^{i-1} \mathbf{e} + \boldsymbol{\mu}_{1:K}^i, \quad (4.11)$$

where $\mathbf{e}$ is a random vector that follows the standard normal distribution $\phi(\mathbf{e}) = \mathcal{N}(\mathbf{0}, I_K)$, and U^i is the Cholesky decomposition of J^i so that $J^i = U^{iT} U^i$. U^i is thus an upper triangular bi-diagonal matrix. We represent its main diagonal as D_0^i , the entries of which are all positive. The counter diagonal is denoted as D_1^i , containing only negative entries. To this end, we assume that $D_0^i = \exp(\mathbf{a})$ and $D_1^i = -\exp(\mathbf{b})$, where $\mathbf{a}, \mathbf{b} \in \mathcal{R}^K$.

After the reparameterization, $\mathcal{L}$ in (4.9) can be elaborated as:

$$\mathcal{L} = \mathbb{E}_{\phi(\mathbf{e})} \left[\sum_{k=1}^K \log p(\mathbf{x}_k | \mathbf{x}_{k-1}, \boldsymbol{\alpha}_k) p(\mathbf{z}_k | \mathbf{x}_k, \boldsymbol{\beta}_k, \mathbf{m}) \right] - \sum_{i=1}^M \log |U^i| + c_0, \quad (4.12)$$

where $\log |U^i| = \sum_{k=1}^K \log U_{kk}^i$, and c_0 is a constant.

Since currently $\phi(\mathbf{e})$ is independent of any unknown parameter, the gradient of its expectation equals the expectation of its gradient. As such, the gradients of $\mathcal{L}$ w.r.t. all the parameters are given as:

$$\nabla_{\boldsymbol{\mu}_k^i} \mathcal{L} = \mathbb{E}_{\phi(\mathbf{e})} [\Delta_k^i], \quad (4.13a)$$

$$\nabla_{D_{0k}^i} \mathcal{L} = -\mathbb{E}_{\phi(\mathbf{e})} \left[\left(U^{i-T} \Delta_{1:K}^i \right)_k \left(U^{i-1} \mathbf{e} \right)_k \right] - \frac{1}{D_{0k}^i}, \quad (4.13b)$$

$$\nabla_{D_{1k}^i} \mathcal{L} = -\mathbb{E}_{\phi(\mathbf{e})} \left[\left(U^{i-T} \Delta_{1:K}^i \right)_k \left(U^{i-1} \mathbf{e} \right)_{k+1} \right], \quad (4.13c)$$

$$\nabla_{\mathbf{m}} \mathcal{L} = \sum_{k=1}^K \mathbb{E}_{\phi(\mathbf{e})} \left[(\beta_k \circ \nabla_{\mathbf{m}} h(\mathbf{x}_k, \mathbf{m}))^T \Psi_k \right], \quad (4.13d)$$

$$\nabla_{\alpha_k} \mathcal{L} = \frac{1}{2} \mathbf{1} \oslash \alpha_k - \frac{1}{2} \mathbb{E}_{\phi(\mathbf{e})} [\Phi_k \circ \Phi_k], \quad (4.13e)$$

$$\nabla_{\beta_k} \mathcal{L} = \frac{1}{2} \mathbf{1} \oslash \beta_k - \frac{1}{2} \mathbb{E}_{\phi(\mathbf{e})} [\Psi_k \circ \Psi_k], \quad (4.13f)$$

where $(\mathbf{a})_k$ is the k^{th} of $\mathbf{a}$, $\circ$, $\oslash$ represent the elementwise product and division, respectively, and $\mathbf{1}$ is a vector with all ones.

In (4.13), for brevity, we use Φ_k and Ψ_k to represent $\mathbf{x}_k - f(\mathbf{x}_{k-1})$ and $\mathbf{z}_k - h(\mathbf{x}_k, \mathbf{m})$, respectively. Δ_k is calculated as:

$$\Delta_k = -\alpha_k \circ \Phi_k + \alpha_{k+1} \circ \nabla_{\mathbf{x}_k} f(\mathbf{x}_k) \circ \Phi_{k+1} + (\beta_k \circ \nabla_{\mathbf{x}_k} h(\mathbf{x}_k, \mathbf{m}))^T \Psi_k. \quad (4.14)$$

The expectations $\mathbb{E}_{\phi(\mathbf{e})}$ in (4.13) are intractable. Provided $\phi(\mathbf{e})$ permits straightforward simulation of independent samples, we estimate gradients in (4.13) by stochastic approximations. For $i = [1 : K]$, we proceed by firstly drawing a sample $\mathbf{e}^{(s)}$ from $\phi(\mathbf{e})$, and then deterministically obtain $(\mathbf{x}_{1:K}^i)^{(s)} = U^{i-1} \mathbf{e}^{(s)} + \boldsymbol{\mu}_{1:K}^i$. After substituting the obtained $(\mathbf{x}_{1:K}^i)^{(s)}$ into Φ_k, Φ_{k+1} , and Ψ_k contained in (4.14), $(\Delta_k^i)^{(s)}$ serves as a stochastic direction for updating $\boldsymbol{\mu}_k^i$. All the remaining quantities in (4.13) are estimated in a similar manner and listed below:

$$\nabla_{\boldsymbol{\mu}_k^i} \mathcal{L} \approx \Delta_k, \quad (4.15a)$$

$$\nabla_{D_{0k}^i} \mathcal{L} \approx -\left[U^{i-T} \Delta_{1:K}^i \right]_k \left[U^{i-1} \mathbf{e} \right]_k - \frac{1}{D_{0k}^i}, \quad (4.15b)$$

$$\nabla_{D_{1k}^i} \mathcal{L} \approx -\left[U^{i-T} \Delta_{1:K}^i \right]_k \left[U^{i-1} \mathbf{e} \right]_{k+1}, \quad (4.15c)$$

$$\nabla_{\mathbf{m}} \mathcal{L} \approx \sum_{k=1}^K (\beta_k \circ \nabla_{\mathbf{m}} h(\mathbf{x}_k, \mathbf{m}))^T \Psi_k, \quad (4.15d)$$

$$\nabla_{\alpha_k} \mathcal{L} \approx \frac{1}{2} \mathbf{1} \oslash \alpha_k - \frac{1}{2} \Phi_k \circ \Phi_k, \quad (4.15e)$$

$$\nabla_{\beta_k} \mathcal{L} \approx \frac{1}{2} \mathbf{1} \oslash \beta_k - \frac{1}{2} \Psi_k \circ \Psi_k. \quad (4.15f)$$

Algorithm 1: Stochastic variational Bayesian inference for full SLAM problems.

Data: $\mathbf{z}_{1:K}, \mathbf{x}_0$

Result: $\mathbf{x}_{1:K}, \mathbf{m}$

Initialize $\boldsymbol{\mu}_{1:K}^{(0)}, U^{(0)}, \boldsymbol{\alpha}_{1:K}^{(0)}, \boldsymbol{\beta}_{1:K}^{(0)}, t = 0$;

while *convergence criterion is not satisfied* **do**

$t = t + 1$;

$\mathbf{e}^{(s)} \sim \phi(\mathbf{e})$;

$(\mathbf{x}_{1:K}^{(t-1)})^{(s)} = (U^{-1})^{(t-1)} \mathbf{e}^{(s)} + \mathbf{u}_{1:K}^{(t-1)}$;

 For a variable $\nu \sim \{\boldsymbol{\mu}_{1:K}, U, \boldsymbol{\alpha}_{1:K}, \boldsymbol{\beta}_{1:K}\}$,

 Calculate the gradient $\nabla_{\nu} \mathcal{L}^{(t)}$ through (A.6) with $(\mathbf{x}_{1:K}^{(t-1)})^{(s)}$;

$\nu^{(t)} = \nu^{(t-1)} + \rho_t \nabla_{\nu} \mathcal{L}^{(t)}$;

The gradients in (4.15) are referred to as stochastic gradients, and they are the unbiased expectations of the exact quantities in (4.13) [5, 15, 129]. With the stochastic gradients, we optimize the ELBO $\mathcal{L}$ in (4.12) iteratively.

4.4.2 Algorithm Overview and Analysis

The offline SVBI is summarized in Algorithm 1. ρ_t denotes the learning rate. We employ the ADAM method [80] to adaptively determine a schedule of $\{\rho_t\}$ in practice. According to [144], when the learning rates satisfy $\sum \rho_t = \infty$ and $\sum \rho_t^2 < \infty$, the iteration procedure will converge to a local maxima of the ELBO $\mathcal{L}$ in (4.12) or the global maxima if $\mathcal{L}$ is concave.

Compared to the point estimate produced by graph-based SLAM, offline SVBI outputs a complete distribution. The covariance information is of great use in practical applications [16, 25, 57, 64, 143]. Stochastic gradients in Algorithm 1 can help to escape from poor solutions [5, 15, 129]. In addition, the uncertainty matrix simulates an annealing process [1, 68] or smoothing-based optimization [202], contributing to a superior optimum [101, 202]. Since we estimate $\boldsymbol{\alpha}_{1:K}, \boldsymbol{\beta}_{1:K}$ in an adaptive manner, the resulting algorithm is robust to inaccurate noise priors. The complexity of offline SVBI is $\mathcal{O}(KN)$ due to the bi-diagonal U^i , linear with both the number of frames and features. However, graph-based SLAM has a complexity of $\mathcal{O}(K^2G + K^3M^3)$ [40, 168], though linear with the dimension of the global map G , cubical with the number of frames K . Therefore, offline SVBI is more appropriate for large SLAM problems than graph-based SLAM. The computation of derivatives

in Algorithm 1 is the most time-consuming part. But since the derivatives consist of K terms, one for each time step, we can exploit parallel architectures such as GPUs to the full extent. Furthermore, we can also construct a control variate [112] and remove the score function term [148] to mitigate the computational burden.

4.5 Simulation Results

We conduct a series of experiments on synthetic data in this section. These experiments aimed to compare the proposed offline SVBI to several state-of-the-art variants of graph-based SLAM. All the experiments were performed on a desktop computer with an Intel(R) Xeon(R) CPU E5-2620 v4 with 2.10 GHz. Based on the simulation results, we conclude that offline SVBI has higher accuracy, enhanced robustness, and improved scalability.

4.5.1 Simulation Setup

We simulated a 3D environment, where a camera is moving while making observations from the surroundings. The velocity on each axis is 0.5m, and the absolute value of the angular velocity is 0.005rad. The noise is time-changing and follows zero-mean Gaussian distributions. We set the standard deviations of transnational and angular noise on each axis as random numbers selected from uniform distributions $\mathcal{U}(0, 0.6a)$ and $\mathcal{U}(0, 0.002c)$ respectively at every time instant. Similarly, the standard deviation of observation noise is randomly selected from $\mathcal{U}(0, g)$ at each time step. Here, a, c, g are all real positive numbers that will be set in the text below. The camera calibration parameters follow the conventions of the KITTI dataset [49].

The test algorithms are listed as below, where we chose the keyframes every two frames in K-GBS.

Graph-Based SLAM (GBS) [51, 86]	Keyframe-based Graph-Based SLAM (K-GBS) [121, 122]	Online SVBI	Offline SVBI
---------------------------------	--	-------------	--------------

We consider three scenarios:

1. **Different noise levels:** The setting of (a, c, g) is listed as below, where a and c are the amplitudes of the translation and rotation noise, g is the amplitude of the measurement noise.

(0.1, 0.1, 0.5)	(0.1, 0.1, 1)	(0.1, 0.1, 2)	(0.05, 0.05, 1)	(0.2, 0.2, 1)
-----------------	---------------	---------------	-----------------	---------------

2. **Different input noise variances:** We denote the true variance of observation noise as β . To simulate the real-life scenario where the true noise priors are unknown, and the estimates are possibly inaccurate, we corrupt β by multiplying a random vector drawn from the uniform distribution $\mathcal{U}(0, \mathbf{2})$. The corrupted observation noise is denoted as $\hat{\beta}$. We test all the algorithms under true value β as well as corrupted $\hat{\beta}$.

3. **Different number of frames and features:**

Evaluate the computational efficiency of all the algorithms under a different number of frames and a different number of observations per frame.

For each scenario, we test 100 runs and compute the average root mean square error (RMSE) of the estimated translation and rotation angle vectors.

4.5.2 Different Noise Levels

Here, the number of frames K is set to be 50, and the number of observations per frame is set to be 200. The input noise variances are all the true values. We present the averaged RMSEs and run time of all the testing algorithms in Table 4.1 and Table 4.2. From Table 4.1 and Table 4.2, we draw the following conclusions:

TABLE 4.1: Average RMSEs of GBS, K-GBS, Online SVBI, and Offline SVBI for different noise levels.

(a, c, g)	GBS		K-GBS		Online SVBI		Offline SVBI	
	Position [m]	Angle [rad]	Position [m]	Angle [rad]	Position [m]	Angle [rad]	Position [m]	Angle [rad]
(0.1, 0.1, 0.5)	0.0136	0.0373e-3	0.0169	0.0478e-3	0.0213	0.1452e-3	0.0142	0.0637e-3
(0.1, 0.1, 1)	0.0277	0.0715e-3	0.0339	0.0893e-3	0.0223	0.1939e-3	0.0189	0.0681e-3
(0.1, 0.1, 2)	0.0562	0.1480e-3	0.0707	0.1842e-3	0.0399	0.2494e-3	0.0350	0.1262e-3
(0.05, 0.05, 1)	0.0284	0.0747e-3	0.0345	0.0920e-3	0.0268	0.1811e-3	0.0114	0.0527e-3
(0.2, 0.2, 1)	0.0275	0.0741e-3	0.0338	0.0882e-3	0.0316	0.2025e-3	0.0271	0.1062e-3

- Offline SVBI achieves the overall highest accuracy with a close processing speed to GBS.
- Since offline SVBI considers the internal relations between frames, its accuracy is superior to that of online SVBI.

TABLE 4.2: Average run time of calculating a frame for GBS, k-GBS, Online SVBI, and offline SVBI under varying noise levels.

(a, b, c)	GBS [s]	K-GBS [s]	Online SVBI [s]	Offline SVBI [s]
(0.1, 0.1, 0.5)	40.62	20.51	0.87	45.88
(0.1, 0.1, 1)	41.52	19.66	0.78	46.92
(0.1, 0.1, 2)	43.47	20.25	0.81	41.42
(0.05, 0.05, 1)	45.44	21.13	0.56	46.80
(0.2, 0.2, 1)	42.91	24.71	0.72	47.99

- GBS outperforms others when the motion noise is large, but the observation noise is relatively small.
- Compared to GBS, K-GBS performs much faster, while at the cost of accuracy.

4.5.3 Different Input Noise Variances

In real-world applications, the noise covariances are generally unknown, and the initializations are often inaccurate. To simulate the real scenarios, we assume the true noise covariances are all unknown and test all the algorithms with true and corrupted noise variances. The (a, c, g) values are chosen to be (0.1, 0.1, 1). The number of frames K is 50, and the number of observations per frame is 200. We present the average accuracy and computational cost in Table 4.3 and Table 4.4 respectively. From Table 4.3, we can observe that inaccurate noise variances have

TABLE 4.3: Average RMSEs of GBS, k-GBS, Online SVBI, and Offline SVBI for varying input noise variances.

Input Noise Variance	GBS		K-GBS		Online SVBI		Offline SVBI	
	Position [m]	Angle [rad]	Position [m]	Angle [rad]	Position [m]	Angle [rad]	Position [m]	Angle [rad]
β	0.0277	0.0715e-3	0.0339	0.0893e-3	0.0223	0.1939e-3	0.0189	0.0681e-3
$\hat{\beta}$	0.3753	8.1257e-4	0.3092	7.4757e-4	0.0252	1.7824e-4	0.0203	9.0116e-5

TABLE 4.4: Average run time of calculating a frame for GBS, K-GBS, Online SVBI, and Offline SVBI under varying input noise variances.

Input Noise Variance	GBS [s]	K-GBS [s]	Online SVBI [s]	Offline SVBI [s]
β	41.52	19.66	0.78	46.92
$\hat{\beta}$	59.68	22.75	0.61	55.50

a substantial impact on the performance of GBS and K-GBS. That is because they rely on the least square error minimization. The noise variances can be viewed as the coefficients of the error terms, and the inaccuracy will lead to performance degeneration. By contrast, both online and offline SVBI estimates the noise variances along with the posterior distribution and thus are robust to inaccurate priors. Moreover, Huber Loss cannot detect and correct inaccurate noise variances timely (see Chapter 3); therefore, here we dismiss the evaluation of GBS and K-GBS with Huber Loss for brevity.

4.5.4 Different Number of Frames and Features

Computational efficiency is essential in practical SLAM applications, where computing resources are limited. When there are a large number of frames and features, scalability is a crucial evaluation metrics for optimization algorithms. We test GBS, K-GBS, and offline SVBI with different numbers of frames and features, and then present their log-log scale plots regarding run time, CPU usage, and memory footprint in Figure 4.2. Furthermore, we calculate the slopes of the curves so that we can express the time complexity as functions of the number of frames and features. The empirical time and space complexities w.r.t. the number of frames are listed as below: In addition, we perform the same operations for the number of observations

Algorithm	GBS	K-GBS	Offline SVBI
Time Complexity	$\Theta(K^{2.42})$	$\Theta(K^{2.47})$	$\Theta(K^{1.38})$
Space Complexity	$\Omega(K^{0.84})$	$\Omega(K^{0.87})$	$\Omega(K^{0.91})$

per frame and then present the results in the following table:

Algorithm	GBS	K-GBS	Offline SVBI
Time Complexity	$\Theta(N^{2.36})$	$\Theta(N^{2.66})$	$\Theta(N^{1.38})$
Space Complexity	$\Omega(N^{1.20})$	$\Omega(N^{1.21})$	$\Omega(N^{0.84})$

As discussed in Section 4.2, GBS has a $\mathcal{O}(K^2G + K^3M^3)$ complexity, whereas offline SVBI has linear $\mathcal{O}(KN)$. As a result, GO is only linear with the number of features, while offline SVBI is linear w.r.t. the number of features as well as the number

FIGURE 4.2: Average run time, CPU usage, and memory footprint of GBS, K-GBS, and offline SVBI for different number of frames and observations per frame.

of frames. Moreover, since the computation of the components $\nabla_{\mathbf{x}_k} h(\mathbf{x}_k, \mathbf{m})$ and $\nabla_{\mathbf{m}} h(\mathbf{x}_k, \mathbf{m})$ in the gradients (4.14) and (4.15d) can be easily parallelized, we verify the run time as a function of the number of cores and frames for a multi-core server. The curves shown in Figure 4.3 suggest that parallelization can reduce the computational complexity, especially for large K . As the number of cores increases, the effect weakens.

4.6 Experimental Results

To further evaluate the proposed offline SVBI method in real-world cases, we integrate the proposed algorithm into the ORB-SLAM2 system [122] (refer to it as ORB-VB-SLAM system), and consider a range of stereo, monocular and RGB-D

FIGURE 4.3: Run time vs. number of cores for different values of K .

datasets. The tests are conducted on an Intel(R) Xeon(R) CPU E5-1630 v3 with 3.70 GHz.

4.6.1 System Overview

ORB-SLAM2 [122] is a complete visual SLAM library for stereo, RGB-D, and monocular cameras. It computes the camera trajectory and performs 3D reconstruction in real-time. ORB-SLAM2 is applicable to a wide range of applications, from small hand-held indoor videos to drones hovering in industrial halls and cars driving around a mid-size city. ORB-SLAM2 is composed of three parallel threads: tracking, local mapping, and loop closing. When a loop is detected during the operation, a fourth thread is created after the pose-graph optimization, allowing the system to continue creating map points and detecting possible loops. The system diagram is shown in Figure 4.4.

The optimizations in ORB-SLAM2 rely on graph-based SLAM [51], and are implemented by the g2o library [86]. Specifically, the tracking thread performs motion-only BA to optimize the current camera pose. The local mapping thread optimizes a local window of keyframes and map points seen in those frames. The full BA estimates all keyframes except the initial one, together with the entire map. We re-implement the full BA module by offline SVBI and obtain a modified ORB-VB+SLAM system. Three reasons support our choices. First of all, the full BA module can contribute to the system performance [121]. Secondly, the modification of the core local mapping thread may easily degenerate or destabilize the system. Last, provided that the other three threads being fixed, ORB-SLAM2 and the

FIGURE 4.4: Structure of ORB-SLAM2 system.

modified ORB-VB+SLAM will have almost the same input, which is crucial for a fair comparison.

4.6.2 Stereo and RGB-D Vision

The depth is observable in stereo and RGB-D scenarios. When images are received, ORB-SLAM2 first extracts features by detecting salient keypoint locations. For rectified stereo images, ORB-SLAM2 extracts ORB features in both frames, and for every left ORB, it searches for the correspondence in the right image. As regarding RGB-D cameras, it only extracts features from the RGB frame and then transforms the depth value of each found feature into a virtual right coordinate through (4.4). Once extraction completed, the system discards the input images and performs the following operations directly on the extracted features, regardless of the sensor type being stereo or RGB-D.

The full BA module is called when loops are detected; consequently, we select live streams that contain loops from KITTI, EuRoC, and TUM datasets to test. Among the three datasets, KITTI is the only outdoor dataset captured by a driving car around a city, from rural areas to highways. Its frequency is 10 Hz. The EuRoC and TUM datasets are indoor, recorded at 20 Hz, and 30 Hz, respectively. EuRoC was collected onboard a Micro Aerial Vehicle in industrial environments. TUM contains RGB-D data recorded from a Microsoft Kinect. In the experiments, the

output trajectory is evaluated by the provided “evaluate_ate_scale” script. This script automatically aligns the scale of the estimated path with the ground truth and computes the Absolute Trajectory mean and RMSE errors [122, 169]. We test each video sequence for 10 times and report the average result. The quantitative results of ORB-SLAM2 and ORB-VB+SLAM are presented in Table 4.5. The numbers in brackets are the relative ratios of the errors of ORB-VB+SLAM to those of ORB-SLAM2.

TABLE 4.5: Average accuracy of ORB-SLAM2 and ORB-VB+SLAM systems on KITTI, EuRoC, and TUM datasets in stereo and RGB-D cases.

Test Cases		ORB-SLAM2		ORB-VB+SLAM	
		ATE.Mean[m]	ATE.RMSE[m]	ATE.Mean[m]	ATE.RMSE[m]
Stereo KITTI	00	0.84	0.90	0.85(101%)	0.92(102%)
	02	05.13	6.22	4.76 (92%)	5.79 (91%)
	05	0.34	0.39	0.38(112%)	0.43(110%)
	06	0.62	0.66	0.54 (89%)	0.59 (89%)
	07	0.44	0.48	0.42 (95%)	0.45 (95%)
	09	1.37	1.45	1.32 (96%)	1.40 (97%)
Stereo EuRoC	V1_03_difficult	0.086	0.099	0.074 (87%)	0.081 (82%)
	V2_02_medium	0.056	0.067	0.053 (95%)	0.058 (87%)
	fr1/room	0.063	0.070	0.055 (88%)	0.061 (88%)
RGB-D TUM	fr2/desk	0.007	0.007	0.007^a(98%)	0.007^a(99%)
	fr3/office	0.010	0.011	0.008 (79%)	0.009 (79%)
	fr3/nstr_tex_near	0.019	0.022	0.018 (88%)	0.020 (88%)
Average		-	-	(93%)	(92%)

^a Here the average ATE.Mean and ATE.RMSE of ORB-VB+SLAM on fr2/desk are less than but are rounded up to 0.007.

From Table 4.5, we can observe that ORB-VB-SLAM outperforms ORB-SLAM2, due to the application of offline SVBI. In sequence 09, a loop occurs in very few frames at the end of the video. Therefore, for each system, we selected 10 successful trials to evaluate. The examples of the estimated trajectories are presented in Figure 4.5. The inset subplots display the endpoints of the calculated trajectories. We can see that the paths estimated by ORB-VB+SLAM are closer to the ground truth than the ones by ORB-SLAM2.

4.6.3 Monocular Vision

The input is a sequence of single images in monocular SLAM. As a result, the depth cannot be observed directly. To tackle this problem, an initial map is required to be created in advance. Previous initialization methods either recover the

FIGURE 4.5: Estimated trajectories of ORB-SLAM2 and ORB-VB+SLAM in KITTI 02, 06, EuRoC V1_03_difficult, V2_02_medium, and TUM fr1/room, fr3/nstr_tex_near in stereo and RGB-D cases.

relative camera pose from a homography or exploit an essential matrix to model planar scenes. Both suffer from ambiguity if points are all close to the camera center. Moreover, these methods are not well constrained under low parallax and fail to handle non-planar scenes. ORB-SLAM2 overcomes this challenge by shifting between homography (for planar scenes) and fundamental matrix (for non-planar scenes) automatically. The map is initialized repeatedly until a unique solution with significant parallax is produced. However, since the true depths are unavailable, the estimated trajectories in monocular cases are subject to high uncertainty, and not at the real scale.

We present the accuracy results of ORB-SLAM2 and ORB-VB+SLAM in Table 4.6. ORB-VB+SLAM outperforms ORB-SLAM2 on most sequences, except

TUM fr1/room where both systems suffer from detection failure. The most probable cause is that fr1/room covers the whole room with little keyframes overlap. The movement is relatively faster than other test sequences. Explicitly, its average translational and angular speed are 0.33m/s and 29.88deg/s respectively, whereas those of fr2/desk are just 0.19m/s and 6.34deg/s. The limitation of monocular SLAM is represented by pose estimation under fast movement, especially pure rotational camera motion. As a consequence, tracking failures occur because stereo estimation cannot be applied in this case. The examples of estimated trajectories with zoomed-up insets are shown in Figure 4.6, demonstrating the superiority of offline SVBI further.

TABLE 4.6: Average accuracy of ORB-SLAM2 and ORB-VB+SLAM systems on KITTI, EuRoC, and TUM datasets in monocular case.

Test Cases		ORB-SLAM2		ORB-VB+SLAM	
		ATE.Mean[m]	ATE.RMSE[m]	ATE.Mean[m]	ATE.RMSE[m]
Monocular KITTI	00	7.25	8.43	7.18(99%)	8.14(97%)
	02	22.96	28.69	20.68 (90%)	27.32 (95%)
	05	5.34	5.69	5.81(109%)	6.18(109%)
	06	13.54	15.59	11.88 (88%)	13.54 (87%)
	07	2.42	2.70	2.58 (107%)	2.87 (106%)
	09	3.52	3.98	3.04 (86%)	3.44 (86%)
Stereo EuRoC	V1.03_difficult	0.146	0.187	0.122 (84%)	0.154 (82%)
	V2.02_medium	0.054	0.057	0.051 (94%)	0.054 (95%)
RGB-D TUM	fr1/room	Failed to detect			
	fr2/desk	0.009	0.009	0.007 (85%)	0.008 (86%)
	fr3/office	0.015	0.017	0.013 (87%)	0.015 (85%)
	fr3/nstr_tex_near	0.012	0.014	0.011 (89%)	0.013 (91%)
Average		-	-	(93%)	(93%)

All the sequences are played at the real frame-rate at which they were recorded. We present the average mean tracking time per frame of both systems in Table 4.7.

4.7 Conclusion

We proposed an algorithm for full SLAM in this chapter, namely, offline SVBI. The map is a global variable, and constraints exist between consecutive frames. We modelled the full SLAM problem in terms of estimating a posterior distribution. Since evaluating this posterior directly is difficult, we employed a Gaussian

FIGURE 4.6: Estimated trajectories of ORB-SLAM2 and ORB-VB+SLAM in KITTI 02, 06, EuRoC V1.03_difficult, V2.02_medium, and TUM fr1/room, fr3/nstr_tex_near in monocular case.

TABLE 4.7: Average computational cost of ORB-SLAM2 and ORB-VB+SLAM systems on KITTI, EuRoC and TUM datasets in stereo, RGB-D and monocular cases.

Case	ORB-SLAM2	ORB-VB+SLAM
Stereo KITTI	0.096 s	0.091 s
Stereo EuRoC	0.053 s	0.051 s
RGBD TUM	0.030 s	0.029 s
Monocular KITTI	0.052 s	0.049 s
Monocular EuRoC	0.038 s	0.038 s
Monocular TUM	0.023 s	0.022 s

distribution for approximation instead. The uncertainty matrix of the Gaussian distribution was set to be tri-diagonal, resulting in a linear computational complexity. We applied stochastic gradients to conduct the approximation, avoiding

local extrema. During the update, we estimated the noise covariances adaptively. Therefore, the proposed offline SVBI is robust to inaccurate noise priors. In the evaluation, the proposed algorithm was shown to produce more accurate estimates than the-state-of-the-art graph-based SLAM on both synthetic and real data. We believe that this offline algorithm is favorable to cases where there are a large number of features and frames, and expect its future applications in real robotic platforms.

Chapter 5

Variational Bayesian Point Set Registration Based on Gaussian Mixture Models

Light Detection and Ranging (Lidar) is a remote sensing sensor based on a form of pulsed lasers. It is equipped with multiple transceivers. Each transceiver quickly emits the laser light to illuminate the target and then measures the reflected pulses to calculate the relative position and distance. The collected information is typically presented as dense point clouds. Compared to vSLAM, Lidar has fast travelling speed and can provide precise locations. Notably, it runs steadily in the outdoor environment. However, the registration between neighbouring point sets poses a challenge. The collected points not only have large data size, but usually suffer from noise, outliers, and missing partial structures [92, 93, 93, 95]. As a consequence, though numerous registration algorithms have been proposed, most of them experience sizeable computational cost or low accuracy under challenging conditions. Moreover, some registration methods are subject to the initial misalignment.

To address these limitations, we describe the point sets as Gaussian mixture models (GMMs), changing accordingly with the transformation. The registration problem is then formulated into a Bayesian state space model. We define the latent states and observations by the transformation variables and L2 distance between two GMMs, respectively. We infer the transformation by a stochastic variational

Bayesian inference and reduce the computational cost by approximating the integration of the L2 distance with a set of quadrature points. This variational Bayesian point set registration based on Gaussian mixture models (GMMs-VB-PSR) is thus efficient and robust. Experimental results show that the proposed GMMs-VB-PSR method achieves higher accuracy with comparable or less time and resources when compared to the state-of-the-art registration methods.

5.1 Introduction

For autonomous driving, a vehicle needs to locate itself while simultaneously building the environmental map through sensor data. This simultaneous localization and mapping (SLAM) process generally consists of two parts: the front end and the back end. The front end performs frame-by-frame pose estimation, while the back end refines this initial guess by correlating observations. Accurate odometry obtained from the front end is crucial for the vehicle to work smoothly and efficiently. Compared with visual odometry (VO), Lidar odometry (LO) has improved accuracy and enhanced stability, especially in outdoor environments. The acquisition of the relative pose between every two consecutive Lidar point clouds is referred to as point set registration.

Point set registration can be divided into rigid [43, 79, 142, 193] and nonrigid [106, 107, 126, 187] according to the transformation type. The rigid registration consists of only translation and rotation, whereas the nonrigid involves nonlinear transformations. Here we consider SLAM applications, hence the registration is assumed to be rigid. The substantial literature on rigid point registration can be roughly grouped into three categories: distance-optimization-based [6, 7, 10, 130, 159, 179, 180, 186], probability-based, and filtering-based methods [93, 94, 115, 153]. Among all the distance-optimization-based algorithms, the iterative closest point (ICP) algorithm [10] stands out due to its simplicity and efficiency. It is widely applied in Lidar-based SLAM. Given an initialized transformation, ICP first maps the moving point set onto the fixed one and then establishes the point-wise correspondences according to the closest distance. The transformation is then updated by minimizing the distance error terms resulting from the matched pairs. The two steps are iterated until convergence is reached.

The computational complexity of ICP is $\mathcal{O}(MD \log(MD))$, where M is the number of points in the template point set, and D is the dimension of the point space. Despite the wide applicability, ICP is susceptible to the initial misalignment, noise, outliers, and missing structures, which often co-exist in the real world. Apart from ICP, another popular approach in Lidar-based SLAM is the normal distributions transform (NDT) algorithm [11]. NDT rasterizes the space of the fixed point set into cells. Normal distributions are employed to describe the points in each cell separately. NDT estimates the transformation by maximizing the likelihood of the mapped points w.r.t. the distributions of the fixed point set. Its computational complexity is $\mathcal{O}(KD \log(KD))$, where K is the number of cells. NDT is robust to outliers and missing structures as it is probability-based. However, it is still subject to initial misalignment. Moreover, it is sensitive to the cell size. To mitigate this issue, probability-based algorithms, such as Coherent Point Drift (CPD) [125] and GMM_L2 [71, 92, 95], and filtering-based methods like cubature Kalman filters (CKF) [93] and split covariance intersection filter (CSCIF) [94] were proposed. However, they cannot be applied in SLAM problems directly due to the high computational cost.

In this work, we model the point set registration problem by a state space model. The transformation variables are the latent states. To handle the noise, outliers, and missing partial structures, we describe the point sets by GMMs as in [71, 92, 95]. The L2 distance of the densities of the GMMs is viewed as the observation. The motion and observation noise are assumed to be additive white noise. We resort to a stochastic variational Bayesian inference to infer the latent states given observations. In order to improve the computational efficiency, we approximate the integration of the L2 distance by a set of quadrature points. The complexity of the proposed method is $\mathcal{O}(LDP)$, where L is the number of quadrature points. Typically, L is a small number in 2D and 3D spaces ($L < 100$).

5.2 Related Work

In this section, we discuss the three common types of registration approaches: distance-optimization-based, probability-based and filtering-based methods.

5.2.1 Distance-Optimization-Based Methods

Most distance-optimization-based point set registration methods comprise two steps. Firstly they build the correspondence relations between two point sets and calculate the distance. Secondly, they minimize the distance parameterized by the relative transformation. The most widely used distance-optimization-based method is ICP [10]. Given two point sets, ICP first associates each other point by point. Next, it calculates the L2 distance and optimize the transformation by least-squares minimization. The two steps are repeated iteratively until reaching the convergence. Since the correspondences are point-wise, ICP is sensitive to the initial misalignment, outliers, noise and missing partial structures. Moreover, ICP gets stuck in local extrema easily. To overcome these disadvantages, various ICP variants [23, 50, 195] were proposed. The globally optimal GO-ICP [195] relies on a branch-and-bound (BnB) scheme to search the entire space and utilizes a trimming strategy to handle outliers. EM-ICP in [50] formulates the registration problem as Maximum-Likelihood (ML) estimation and infers the transformation based on expectation-maximization (EM) principles [118]. The noise variances are modelled as a “scale” parameter and exploited in an annealing scheme. Although GO-ICP and EM-ICP are more robust than ICP, they have higher computational cost. Apart from the L2 distance, the correlation can also be measured by kernel-based methods [130, 179, 186]. Since the kernel correlation mathematically equals the entropy of two point sets, these methods usually measure the entropy instead and employ gradient-based methods to minimize. Specifically, the method in [180] associates the correspondences via the Havrda-Charvat-Tsallis entropy and minimizes the resulting divergence via a deterministic annealing approach. KL divergence [6, 7] and Kernelized Renyi distance [159] were also proposed for measuring the distance. Though distance-optimization-based methods have a large variety of applications in practice, they achieve reliable estimations only for proper initialization. In addition, most of them degrade in the presence of noise, outliers, missing partial structures, and local extrema.

5.2.2 Probability-Based Methods

Probability-based algorithms represent the point sets as probability densities. The similarity between two point sets [71, 92, 95] or the probability of one point set

represented by the other [125] is minimized iteratively. The most widely-used probability-based algorithms include CPD [125], GMM.L2 [71] and NDT [11]. CPD [125] fits the GMM centroids representing the moving point set to the fixed point set via maximizing the likelihood. The maximization relies on an expectation-maximization (EM) algorithm [118]. The GMM centroids thus always preserve a topological structure through moving coherently as a group. They impose a coherence constraint in rigid cases. The GMM.L2 method was first proposed in [71]. It represents the fixed and moving sets as two GMMs, and minimizes the L2 distance between the densities of GMMs to find the optimal transformation. GMM.L2 and CPD are robust to outliers and missing structures, but both of them are computationally burdensome. NDT [11] transforms the discrete points from the fixed point set into a probability density consisting of mixtures of Gaussian distributions. Then NDT maps the transformed points from the moving set to the fixed density and maximizes the “score” defined by density values. NDT is widely applied in SLAM tasks. However, it is inappropriate if the rasterize space cannot reliably characterize the shape of the point sets. In addition, the performance of NDT is sensitive to the grid size as well as the initial misalignment. Some other probability-based methods are presented in [27, 92, 95, 170, 171].

5.2.3 Filtering-Based Methods

Filtering-based methods model the registration problem as a state space model. The process model is represented by $\mathbf{x}_k = \mathbf{x}_{k-1} + \boldsymbol{\eta}_k$. $\mathbf{x}_k$ and $\mathbf{x}_{k-1}$ are the states of iteration k and $k - 1$ respectively, and $\boldsymbol{\eta}_k$ is the additive noise that follows the Gaussian distribution $\mathcal{N}(0; Q_{k-1})$. The observation model measures the distance between two point sets after associating them point by point. A particle filter was proposed in [153] to predict the state by a proposed density. The particles are updated to minimize the observation error after running a preset number of ICP steps. The UKF-based algorithm proposed in [115] assumes the correspondences between two point sets to be known, and the noise as constants. Starting from the first point, after one iteration, one more point is added, and then a new iteration is performed. The nonlinearity is approximated by quadratic items. By contrast, CKF [93] and CSCIF [94] handle the nonlinearity through third-order statistics where the correspondences are unknown. The system noise is determined dynamically through the continuous simulated annealing (CSA) approach. Furthermore,

FIGURE 5.1: Dynamic Bayesian Network of the point sets generation in a sequential system.

CSCIF introduced a split covariance intersection filter (SCIF) to capture the correlation between the transition and observation fully. Filtering-based methods characterize various types of errors successfully. However, UKF, CKF, and CSCIF have a cubic computational complexity due to the inversion of the covariance matrix. Particle filter requires a large number of particles typically. Therefore, filtering-based methods typically cannot be applied to dense points directly because of the high computational cost.

5.3 Bayesian Formulation of Point Registration

Given the model set $\mathbf{U} = \{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_M\}$ and the scene set $\mathbf{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_N\}$ in $\mathbb{R}^D$, the objective of point registration is to find the relative transformation between the two sets, which is defined as $\mathbf{Z} = \mathcal{T}(\mathbf{U}, \mathbf{x})$. Specifically, $\mathbf{x}$ is the transition vector and defined as $[\mathbf{x}^t; \mathbf{x}^\theta]$, where $\mathbf{x}^t$ is the translation vector while $\mathbf{x}^\theta$ encodes the rotation information. For every point $\mathbf{u}_i$ in $\mathbf{U}$, its correspondence in the space of $\mathbf{Z}$ given the alignment $\mathbf{x}$ can be computed as $R(\mathbf{x}^\theta)\mathbf{u}_i + \mathbf{x}^t$, where R is the rotation matrix that is a nonlinear function of $\mathbf{x}^\theta$.

To enhance the robustness to outliers, we describe each finite point set as a GMM as in [71, 92, 95]. As a result, the distribution of the point cloud $\mathbf{U}$ and $\mathbf{Z}$ can be respectively expressed as $\mathcal{G}_U(\mathbf{u}) = \sum_{i=1}^m \omega_i \mathcal{N}(\mathbf{u}; \boldsymbol{\mu}_i, \Sigma_i)$ and $\mathcal{G}_Z(\mathbf{z}) = \sum_{j=1}^n \lambda_j \mathcal{N}(\mathbf{z}; \boldsymbol{\nu}_j, \Xi_j)$, where $\mathcal{N}(\mathbf{u}; \boldsymbol{\mu}_i, \Sigma_i)$ is a Gaussian distribution with mean $\boldsymbol{\mu}_i$ and covariance Σ_i , and ω_i and λ_j denote the weight of component i . The GMMs are obtained by clustering techniques [111]. When the point sets are dense (i.e.,

M and N are large), characterizing them using GMMs can be viewed as an efficient way for downsampling. The registration between $\mathbf{Z}$ and $\mathbf{U}$ thus amounts to aligning $\mathcal{G}_Z$ to $\mathcal{G}_U$.

Suppose that the point sets are all generated from a sequential system, e.g., the Lidar-based SLAM, the registration of subsequent frames can be expressed as a state space model:

$$\mathbf{x}_t = 2\mathbf{x}_{t-1} - \mathbf{x}_{t-2} + \boldsymbol{\delta}_t, \quad (5.1a)$$

$$\mathcal{G}_Z(\mathbf{y}) = \mathcal{T}(\mathcal{G}_U, \mathbf{x}_t)(\mathbf{y}) + \epsilon_t(\mathbf{y}), \quad \forall \mathbf{y} \in \mathbb{R}^D \quad (5.1b)$$

where $\mathcal{T}(\mathcal{G}_U, \mathbf{x})(\mathbf{y}) = \sum_{i=1}^m \omega_i \mathcal{N}(\mathbf{y}; R\boldsymbol{\mu}_i + \mathbf{t}, R\Sigma_i R^T)$ is the density function of the GMM for $\mathcal{G}_U$ after the transformation $\mathcal{T}$, $\mathbf{y}$ is the argument of the density function, $\boldsymbol{\delta}_t$ and $\epsilon_t(\mathbf{y})$ denotes motion and observation noise. The motion function (5.1a) can be equivalently written as:

$$\mathbf{x}_t - \mathbf{x}_{t-1} = \mathbf{x}_{t-1} - \mathbf{x}_{t-2} + \boldsymbol{\delta}_t, \quad (5.2)$$

where $\mathbf{x}_{t-1} - \mathbf{x}_{t-2}$ can be regarded as the acceleration at time $t - 1$.

The above expression suggests that the motion model favors the case where the acceleration is time invariant. Viewed another way, it prefers $2\mathbf{x}_{t-1} - \mathbf{x}_t - \mathbf{x}_{t-2}$ to be close to zero, which is the curvature at $\mathbf{x}_{t-1}$. As such, $\mathbf{x}_t$ will change smoothly across time. Such priors are frequently used for smoothing signals in the literature of statistical signal processing [200]. On the other hand, the observation model can be mathematically interpreted as the density of the point set $\mathbf{Z}$ equals the density of transformed $\mathbf{U}$ plus independent Gaussian noise at all points $\mathbf{y} \in \mathbb{R}^D$. Assuming that $\boldsymbol{\delta}_t \sim \mathcal{N}(\mathbf{0}; \alpha_t^{-1} I_P)$ and $\epsilon_t(\mathbf{y}) \sim \mathcal{N}(\mathbf{0}; \beta_t^{-1})$, where α_t and β_t are the inverse variances and I_P is a $P \times P$ square identity matrix, the maximum a posterior estimate of $\mathbf{x}$ is then given by:

$$\mathbf{x} = \underset{\mathbf{x}}{\operatorname{argmin}} \|\alpha_t(\mathbf{x}_t - 2\mathbf{x}_{t-1} + \mathbf{x}_{t-2})\|_2^2 + \beta_t \int (\mathcal{G}_Z(\mathbf{y}) - \mathcal{T}(\mathcal{G}_U, \mathbf{x}_t)(\mathbf{y}))^2 d\mathbf{y}. \quad (5.3)$$

In other words, by solving the SSM, we select the transformation $\mathbf{x}$ that maximizes the similarity between the two GMMs while considering the smoothness of $\mathbf{x}_t$ across time. We notice that the second term in (5.3) is the L2 distance between two distributions. It has been pointed out in [71] that L2 distance is more robust to

outliers and local extrema than other distances, such as KL divergence. Although the integration in the L2 distance is tractable, its calculation generally takes excessive computational resources. As a remedy to this situation, it is both easier and computationally feasible to numerically approximate this integration, as well as estimate the inverse variance β_t following the Bayesian paradigm. Explicitly, we utilize the Gauss-Hermite quadrature rule, which is shown to be accurate for the integration over GMMs [100]. Let us consider L quadrature points $\{\mathbf{y}_1, \dots, \mathbf{y}_L\}$ with associating weights $\{w_1, \dots, w_L\}$. For brevity, we introduce the following notation: $\mathbf{z}_t = [\mathcal{G}_z(\mathbf{y}_1), \dots, \mathcal{G}_z(\mathbf{y}_L)]^T$, $h_t(\mathbf{x}_t) = [\mathcal{T}(\mathcal{G}_U, \mathbf{x}_t)(\mathbf{y}_1), \dots, \mathcal{T}(\mathcal{G}_U, \mathbf{x}_t)(\mathbf{y}_L)]^T$, and $\mathbf{w} = [w_1, \dots, w_L]$. As such, the observation model can be written as:

$$\mathbf{z}_t = h_t(\mathbf{x}_t) + \boldsymbol{\iota}_t, \quad (5.4)$$

where $\boldsymbol{\iota}_t \sim \mathcal{N}(\mathbf{0}; \beta_t^{-1}[\text{diag}(\mathbf{w})]^{-1})$, and $\text{diag}(\mathbf{w})$ is a $L \times L$ diagonal matrix with $\mathbf{w}$ on the diagonal. Note that the quadrature points downsample the observation, and therefore, will speed up the inference process. Similarly, existing methods [10, 11, 93, 94, 125] sometimes also downsample the point sets to make the algorithm scalable to the dense cases. They allocate the points into grid cells and represent each cell by the mean of the point distribution in this cell. Such downsampling approach, however, is sensitive to the grid size and the noise in the point sets. By contrast, the proposed method can handle noisy data and hence models the point sets more accurately.

Accordingly, both the motion and observation function can be described from a probabilistic perspective:

$$p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \alpha_t) \propto \alpha_t^{\frac{P}{2}} \exp \left\{ -\frac{\alpha_t}{2} (\mathbf{x}_t - (2\mathbf{x}_{t-1} - \mathbf{x}_{t-2}))^T (\mathbf{x}_t - (2\mathbf{x}_{t-1} - \mathbf{x}_{t-2})) \right\}, \quad (5.5a)$$

$$p(\mathbf{y}_t | \mathbf{x}_t, \beta_t) \propto \beta_t^{\frac{L}{2}} \exp \left\{ -\frac{\beta_t}{2} (\mathbf{z}_t - h_t(\mathbf{x}_t))^T \text{diag}(\mathbf{w}) (\mathbf{z}_t - h_t(\mathbf{x}_t)) \right\}. \quad (5.5b)$$

In (5.5b), if we consider all $\mathbf{y} \in \mathbb{R}^D$ and compute the L2 distance analytically, $L \rightarrow \infty$ and so the observation distribution is not well defined. Since we intend to infer β_t from the data, we resort to the quadrature rule mentioned above to choose L points from $\mathbb{R}^D$.

5.4 Stochastic Variational Bayesian Inference

Given the SSM, our ultimate goal is to estimate the transformation $\mathbf{x}_t$ and the noise parameters α_t and β_t given the observations $\mathbf{z}_t$ and the previous states $\mathbf{x}_{t-1}$ and $\mathbf{x}_{t-2}$. Concretely, we exploit the empirical Bayes framework, that is, maximizing the likelihood $p(\mathbf{z}_t|\mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \alpha_t, \beta_t)$ to produce the point estimates of α_t and β_t . The posterior distribution of $\mathbf{x}_t$ can subsequently be expressed as $p(\mathbf{x}_t|\mathbf{z}_t, \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \alpha_t, \beta_t)$. Unfortunately, both of these two distributions cannot be computed analytically, due to the strong nonlinearity in the observation model (5.4). To get around this problem, we maximize the evidence lower bound (ELBO) $\mathcal{L}$ of the likelihood $p(\mathbf{z}_t|\mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \alpha_t, \beta_t)$ instead:

$$\begin{aligned}\mathcal{L} &= \int q(\mathbf{x}_t) \log \frac{p(\mathbf{z}_t, \mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \alpha_t, \beta_t)}{q(\mathbf{x}_t)} d\mathbf{x}_t, \\ &\leq \log p(\mathbf{z}_t|\mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \alpha_t, \beta_t).\end{aligned}\tag{5.6}$$

The equality holds if and only if:

$$q(\mathbf{x}_t) = p(\mathbf{x}_t|\mathbf{z}_t, \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \alpha_t, \beta_t).\tag{5.7}$$

As mentioned before, $p(\mathbf{x}_t|\mathbf{z}_t, \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \alpha_t, \beta_t)$ is not available. Thus, we follow the variational Bayes approach and seek a tractable $q(\mathbf{x}_t)$ and point estimates of α_t and β_t to maximize $\mathcal{L}$. More specifically, we set $q(\mathbf{x}_t)$ to be Gaussian, that is, $q(\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}_t, C_t C_t^T)$, where C_t is the Cholesky decomposition of the covariance of $q(\mathbf{x}_t)$. As a result, we can equivalently parameterize $\mathbf{x}_t$ as:

$$\mathbf{x}_t = C_t \mathbf{e} + \boldsymbol{\mu}_t,\tag{5.8}$$

where $\mathbf{e} \sim \mathcal{N}(\mathbf{0}, I_P)$ follows a standard normal distribution. Here, we leverage the mean-field approximation and assume the covariance matrix of $q(\mathbf{x}_t)$ to be diagonal. As a consequence, the resulting ELBO can then be written as:

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_q[\log p(\mathbf{z}_t, \mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \alpha_t, \beta_t)] + \mathbb{H}[q(\mathbf{x}_t)], \\ &= \mathbb{E}_q[\log(\det(C_t)p(\mathbf{z}_t, C_t \mathbf{e} + \boldsymbol{\mu}_t)|\mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \alpha_t, \beta_t))] \\ &\quad + \log(\det(C_t)) + \mathbb{H}[\phi(\mathbf{e})],\end{aligned}\tag{5.9}$$

where $\mathbb{E}_q[g(\mathbf{x}_t)]$ denotes the expectation of $g(\mathbf{x}_t)$ over $q(\mathbf{x}_t)$, $\mathbb{H}$ denotes the entropy of a distribution, and $\phi(\mathbf{e})$ is the density function of $\mathbf{e}$. In (5.9), $\mathbb{H}[\phi(\mathbf{e})]$ is a constant, and $\log \det C_t$ can be simplified as $\log(\det(C_t) = \sum_{i=1}^P \log[C_t]_{ii})$, since C_t is a lower triangular matrix.

We follow the gradient ascent method to maximize the ELBO $\mathcal{L}$. The gradient of $\mathcal{L}$ w.r.t. $\boldsymbol{\mu}_t$, C_t , α_t , and β_t are calculated as:

$$\nabla_{\boldsymbol{\mu}_t} \mathcal{L} = \mathbb{E}_{\phi(\mathbf{e})} \left[-\alpha_t \boldsymbol{\Theta}_t + \beta_t (\nabla_{\mathbf{x}_t} h_t(\mathbf{x}_t))^T \boldsymbol{\Psi}_t \right], \quad (5.10a)$$

$$\nabla_{C_t} \mathcal{L} = \mathbb{E}_{\phi(\mathbf{e})} \left[(-\alpha_t \boldsymbol{\Theta}_t + \beta_t (\nabla_{\mathbf{x}_t} h_t(\mathbf{x}_t))^T \boldsymbol{\Psi}_t) \circ \mathbf{e} \right] + \Delta_{C_t}, \quad (5.10b)$$

$$\nabla_{\alpha_t} \mathcal{L} = \frac{P}{2\alpha_t} - \frac{1}{2} \mathbb{E}_{\phi(\mathbf{e})} [\boldsymbol{\Theta}_t^T \boldsymbol{\Theta}_t], \quad (5.10c)$$

$$\nabla_{\beta_t} \mathcal{L} = \frac{L}{2\beta_t} - \frac{1}{2} \mathbb{E}_{\phi(\mathbf{e})} [\boldsymbol{\Psi}_t^T \boldsymbol{\Psi}_t], \quad (5.10d)$$

where $\circ$ denotes element-wise product, Δ_{C_t} denotes a diagonal matrix with a diagonal $[\frac{1}{C_{t11}}, \dots, \frac{1}{C_{tPP}}]$, $\boldsymbol{\Theta}_t = \mathbf{x}_t - 2\mathbf{x}_{t-1} + \mathbf{x}_{t-2}$, and $\boldsymbol{\Psi}_t = \sqrt{\text{diag}(\mathbf{w})} (\mathbf{z}_t - h_t(\mathbf{x}_t))$.

The above gradients, however, cannot be calculated in closed form, due to the intractable expectations. Instead, we replace the exact gradients in (5.10a) by their stochastic approximations. Stochastic gradients are unbiased estimates of the exact gradient that can be evaluated in a computationally inexpensive manner. More precisely, we estimate the expectations over $\phi(\mathbf{e})$ by drawing one sample from $\phi(\mathbf{e})$ and evaluating the function value based on this sample. For example, the unbiased stochastic gradient of (5.10a) can be calculated as:

$$\nabla_{\boldsymbol{\mu}_t} \mathcal{L} \approx -\alpha_t \boldsymbol{\Theta}_t + \beta_t \nabla_{\mathbf{x}_t} h_t(\mathbf{x}_t)^T \boldsymbol{\Psi}_t. \quad (5.11)$$

The stochastic gradients of C_t , α_t , and β_t are computed in a similar fashion. Equipped with the stochastic gradients, we maximize $\mathcal{L}$ following Algorithm 2, where $\rho_t^{(k)}$ is the step size in iteration k . In practice, we employ ADAM [80] to determine $\rho_t^{(k)}$ adaptively. According to the Robbins-Monro theorem [144], the stochastic gradient ascent algorithm in Algorithm 2 is guaranteed to converge if the step sizes satisfy $\sum_k \rho_t^{(k)} = \infty$ and $\sum_k (\rho_t^{(k)})^2 < \infty$. In addition, we reduce the gradient variance and improve the convergence speed through the control variate in [112] and removal of score functions [148]. This stochastic variational Bayesian inference (Algorithm 2) can also be interpreted as a simulated annealing-type method.

Algorithm 2: Stochastic variational Bayesian inference.

Data: $\mathbf{z}_t, \mathbf{x}_{t-1}, \mathbf{x}_{t-2}$

Result: $\mathbf{x}_t$

Initialize $\boldsymbol{\mu}_t^{(0)}, C_t^0, \alpha_t^0, \beta_t^0, k=0$;

while *convergence criterion is not met* **do**

$k=k+1$;

$\mathbf{e} \sim \phi(\mathbf{e})$;

$\mathbf{x}_t^k = C_t^k \mathbf{e} + \mathbf{u}_t^k$;

 For a variable $\nu_t \sim \{\boldsymbol{\mu}_t, C_t, \alpha_t, \beta_t\}$,

 Calculate its stochastic gradient $(\nabla_{\nu} \mathcal{L})^{(k)}$; $\nu_t^{(k+1)} = \nu_t^{(k)} + \rho_t^{(k)} (\nabla_{\nu} \mathcal{L})^{(k)}$;

end

Algorithm 3: Variational Bayesian Point Set Registration based on Gaussian mixture models (GMMs-VB-PSR).

Data: Template point sets $\mathbf{U}_{1:T}$, Scene Point sets $\mathbf{Z}_{1:T}$

Result: $\mathbf{x}_{1:T}$

Estimate $\mathbf{x}_1$ and $\mathbf{x}_2$;

for $t \leftarrow 3$ **to** T **do**

 Represent U_t and Z_t by GMMs $\mathcal{G}_{U_t}$ and $\mathcal{G}_{Z_t}$;

 Calculate the quadrature points of $\mathcal{G}_{Z_t}$;

 Initialize $\mathbf{x}_t$ as $2\mathbf{x}_{t-1} - \mathbf{x}_{t-2}$;

 Employ Algorithm 2 to refine the initial estimation;

 Output $\mathbf{x}_t$;

end

It can easily escape from shallow local maxima [101, 202]. As opposed to the proposed method, most existing registration methods [10, 71, 93, 94, 115, 153] are susceptible to local extrema. After obtaining the transformation $\mathbf{x}_t$ at time t , we then proceed to time $t + 1$. The overall variational Bayesian point set registration based on Gaussian mixture models (GMMs-VB-PSR) algorithm is summarized in Algorithm 3. The computational complexity at each time step is $\mathcal{O}(LP)$, where L is a small number typically less than 100. Thus the proposed registration method is suitable for real-time applications where the number of points is huge.

5.5 Experimental Results

In this section, we benchmark the proposed GMMs-VB-PSR method with the commonly-used ICP [10] and NDT [11], the representative probability-based method CPD [125], and the most recent filtering-based method CSCIF [94]. We simulate

(A) Trajectory of the vehicle in the 2D case. (B) Trajectory of the vehicle in the 3D case.

FIGURE 5.2: Ground truth in the 2D (a) and the 3D (b) case for one dataset.

synthetic data from the proposed SSM in (6.6). Specifically, the initial translation and angular velocity on each axis at $t = 1$ are set to be $0.05n$ and 0.02rad . The translation and angular acceleration on each axis are set to be $0.005\text{m}/(\Delta t)^2$ and $0.002\text{rad}/(\Delta t)^2$. We set $\alpha_t = 1/0.005^2\text{m}^2$ for the translation vector and $\alpha_t = 1/0.002^2\text{rad}^2$ for the angles. The vehicle starts from the origin point. We consider both 2D and 3D case. Examples of the trajectories in these two cases are shown in Figure 5.2a and Figure 5.2b respectively.

The point sets are sampled from the distributions of the GMMs at each time instant. The initial GMM at $t = 1$ is assumed to have two equally-weighted components. The mean and covariance of the GMMs at time t is changed according to the rigid transformation specified by $\mathbf{x}_t$. We set the number of points as 5×10^3 and 1×10^5 for 2D and 3D case respectively. The number of frames T is 50. We set the grid step of NDT as 0.1 m . Additionally, since CSCIF cannot be applied to dense points directly, we downsample the input points with a grid filter for CSCIF, the grid size of which is 1.5 m . We do the same for CPD in the 3D case, but the grid size 0.5 m . The transformation $\mathbf{x}_t$ at t is initialized as $\mathbf{x}_{t-1} - 2\mathbf{x}_{t-2}$ for all the methods. All results presented here have been averaged over 100 data sets.

We depict the root mean square error (RMSE) of the estimated $\mathbf{x}_t$ as a function of time t in Figure 5.3a-Figure 5.3d. It shows that the RMSE of the proposed GMMs-VB-PSR achieves the highest accuracy in almost all cases. However, the RMSEs of CSCIF and CPD are several orders of magnitude larger and increase

TABLE 5.1: Average runtime per frame for all methods in 2D and 3D cases.

Algorithm	ICP	NDT	CPD	CSCIF	GMMs-VB-PSR
2D case	0.03 s	0.58 s	0.53 s	17.82 s	0.29 s
3D case	0.63 s	23.51 s	1.64 s	48.35 s	0.88 s

sharply with time t . The probable reasons are that the downsampling by grid filters cause serious information loss, and deteriorate the performance. By contrast, the sampled quadrature points in GMMs-VB-PSR can approximate the GMMs more accurately. On the other hand, ICP and NDT perform better than CSCIF and CPD, since they operated on the full point set. Unfortunately, both ICP and NDT are sensitive to the initial misalignment and get stuck in inadequate solutions easily. For instance, ICP has a poor performance at the initial frames in the 3D case. By contrast, the stochastic gradients employed in GMMs-VB-PSR are more capable of escaping from the local maxima and hence yields higher accuracy.

We present the average registration time per frame for all methods in Table 5.1. We can observe that ICP achieves the fastest processing speed due to its simplicity. The computational time of the proposed GMMs-VB-PSR is almost the same as ICP in the 3D case. However, CSCIF has a cubic computational complexity and thus takes much longer time than the other methods. To assure sufficient accuracy, we chose a relatively small grid size for NDT, resulting in a substantial computational cost in the 3D case [93].

Furthermore, we present the memory and CPU usage of all methods without downsampling in Figure 5.4. It demonstrates that GMMs-VB-PSR has low CPU and memory usage. Thus the proposed method favours the practical SLAM applications where the computing resources are limited.

Last, we test all the algorithms with a different number of points in the 3D case and present the log-log scale plots of their average computational time in Figure 5.5. Moreover, we calculate the empirical time complexities w.r.t. the number of points from the curves in Figure 5.5 and list the results as below. We can observe that GMMs-VB-PSR has the lowest complexity, while the complexity of CSCIF is the highest, corresponding to the computational complexities listed in Section 5.1.

FIGURE 5.3: Logarithm of the RMSE for the translation vector $\mathbf{x}^t$ and the rotation vector $\mathbf{x}^\theta$ as a function of time t resulting from all methods in both 2D and 3D case.

Algorithm	ICP	NDT	CPD	CSCIF	GMMs-VB-PSR
Time Complexity	$\Theta(M^{0.58})$	$\Theta(M^{0.95})$	$\Theta(M^{1.62})$	$\Theta(M^{2.62})$	$\Theta(M^{0.26})$

5.6 Conclusions

In this chapter, we proposed a novel Bayesian state space model for the point registration problem in Lidar-based SLAM. The point sets were described by GMMs.

FIGURE 5.4: Average memory (a) and CPU (b) usages of all the test methods in 2D and 3D cases without downsampling. CPD and CSCIF run out of memory in the 3D case.

FIGURE 5.5: Average run time of ICP, NDT, CPD, CSCIF, and GMMs-VB-PSR for different number of points.

The transformation (i.e., the hidden states in the SSM) was assumed to change smoothly across time, and the observation was the density of the GMM at selected quadrature points given the transformation. An efficient stochastic variational Bayesian inference algorithm was derived for learning this model. Numerical results showed that the proposed method could achieve higher accuracy with a reasonable amount of computational time and resources when compared to the existing registration methods.

Chapter 6

Variational Bayesian Point Set Registration Based on Kernel Density

One of the main downsides to Lidar sensors is that the information will be lost when an object is occluded by another or of an inconsistent shape (see Chapter 5). In addition, outliers, noise and initial misalignment are also issues to be addressed during the registration process in Lidar-based SLAM. To overcome these challenges, we introduced the variational Bayesian point set registration method based on Gaussian mixture models (GMMs-VB-PSR) in Chapter 5, and validated its performance on clustered synthetic data. The proposed GMMs-VB-PSR is heavily dependent on the meaningful groups of the point data. However, the point cloud (i.e., Velodyne data) extracted from Lidar sensors (Figure 6.1) are in the form of discrete circles. Its density is gradually sparsening from the centre towards faraway. Such point structures pose challenges for GMMs-VB-PSR due to two reasons. Firstly, the number of mixture components is difficult to determine, and heuristic approaches often have a high computational cost. Secondly, even an appropriate number has been chosen; the representation through clustering may not be reliable.

To overcome this limitation, we propose another variational Bayesian point set registration method based on kernel density (KD-VB-PSR) in this chapter. It works for any general point sets and thus can handle practical Velodyne data (Figure 6.1). Similarly to GMMs-VB-PSR, the proposed KD-VB-PSR employs a state

FIGURE 6.1: Point cloud collected from Velodyne VLP-16 LiDAR.

space model to describe the sequential registration problem. The state is the transformation variable, and the observation is the L2 distance between two densities representing the point sets. However, unlike GMMs-VB-PSR, here we represent point sets by Gaussian kernel densities, where each kernel corresponds to a point. The mean is set to be the point centroid, and the covariance is initialized as an identity matrix. Due to the massive amount of points, the integration of L2 distance between two kernel densities is cumbersome. To ease this issue, we sample randomized Quasi-Monte Carlo (RQMC) [89] points to approximate this integration numerically. Compared with the standard random Monte Carlo points, the constructed RQMC points can cover the whole space evenly with a small set. The state estimation is then naturally converted into a posterior estimation problem. Here, we stick to stochastic variational Bayesian inference to approximate this posterior. The variational distribution is Gaussian. This inference can be viewed as a smoothing-based optimization and can escape from shallow local extrema easily [101, 202]. In addition, the employed stochastic gradients can also help to avoid poor solutions [5, 15, 129]. Considering the huge amount of points and real-time requirement, we further increase the processing speed via the SVRG method [73]. This algorithm optimizes the finite sums on a minibatch and converges reliably. As such, it can be used to relieve the computational load resulting from the calculation of kernel densities. The adoption of SVRG may produce larger variances for the stochastic gradients. As a remedy to this situation, we employ the control variates

[112] and dismiss the score functions [148] to reduce the variances and guarantee the convergence speed.

Conventional registration algorithms for Lidar-based SLAM include ICP and NDT. However, ICP is sensitive to outliers, noise, missing partial structures, local extrema, and initial misalignment; NDT is subject to proper cell sizes and works for cases where there is a large portion of overlap. On the other hand, probability-based and filtering-based methods overcome the above challenges but are confronted with a high computational cost. By contrast, the proposed KD-VB-PSR successfully enhances robustness but avoids a dramatically increased cost. Further experimental results presented in this chapter will illustrate its advantages in detail.

6.1 Related Work

Point set registration is a fundamental problem in a wide range of applications, such as shape recognition, image retrieval, and medical image analysis. The registration can be classified as rigid and non-rigid. The rigid point set registration plays an important role in the field of robotics. For instance, the Lidar-based SLAM [59, 134, 138] match consecutive frames to estimate the relative transformation for localization. However, the performance of registration in Lidar-based SLAM is profoundly affected by various types of errors. First of all, there exist point position noise caused by the sensor and the environment. Secondly, the received points usually are accompanied by outliers and missing partial structures. Last but not least, the initial alignment often has a large deviation.

The precision is guaranteed only when these errors are fully characterized. However, most existing methods have only a partial description. The classic ICP [10] build correspondences between neighbouring sets and minimize the distance by least square errors. It is easy to implement but heavily influenced by noise, outliers, missing parts, and initial misalignment. NDT [11] is the other widely applied method in Liadr-based SLAM. It can resist outliers and missing structures. However, it works well only when there initially exists a large overlap. CPD [125] and GMM.L2 [71] are probability-based methods. They describe the point sets by continuous probability densities and make no explicit correspondences. This representation ensures the accuracy under outliers and missing partial structures,

but result in a high computational cost. Thus CPD and GMM_L2 are not directly applicable to Lidar-based SLAM where the number of points is huge. People also proposed filtering-based methods [93, 94, 115, 153] to describe the system noise. They formulate the registration problem as a state space model. The system state is the transformation variables, and the observations are the point sets. They approximate the system state in a recursive manner, where the iteration number represents the artificial “time” in the state space model. These filtering schemes have proven to be effective on small datasets [94]. However, they have high computational complexity, consequently, cannot be applied to Lidar-based SLAM directly. Although techniques like grid-based downsampling [136], clustering and filtering can make probability-based and filtering-based algorithms applicable to dense points, they will also lead to accuracy reduction.

6.2 Problem Formulation

Let $\mathbf{U} = \{\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_M\}$ and $\mathbf{Z} = \{\mathbf{z}_1, \mathbf{z}_2, \dots, \mathbf{z}_N\}$ in $\mathbb{R}^D$ be the two finite point sets to be registered, where $\mathbf{U}$ is the moving model set, and $\mathbf{z}$ is the fixed scene set. The set sizes M and N are not required to be identical. The goal of registration is to find the relative spatial transformation that most accurately aligns the model set to the scene set. Registration can be grouped into rigid and non-rigid. Rigid registration yields a rigid transformation that maps the points in one set to the other without changing the distance between two points. By contrast, nonrigid registration involves a nonlinear transformation parametrized by a thin plate spline [71] or eigenvalues [125]. In the context of Lidar-based SLAM, we focus on the rigid cases. Consequently, we denote the transformation as $\mathbf{x} = [\mathbf{x}^t; \mathbf{x}^\theta]$, where $\mathbf{x}^t$ and $\mathbf{x}^\theta$ are the translation and orientation vectors. As such, provided $\mathbf{x}$, the transformed model set $\mathcal{T}(\mathbf{U}, \mathbf{x})$ can be calculated as $R(\mathbf{x}^\theta)\mathbf{u}_i + \mathbf{x}^t$, where $\mathbf{u}_i$ is a point in the original $\mathbf{U}$ set, and R is the rotation matrix converted from rotation angles $\mathbf{x}^\theta$.

The rotation matrix $R(\mathbf{x}_k^\theta)$ is formulated by the following rotation parameters:

$$R(\mathbf{x}_k^\theta) = R(\theta_x)R(\theta_y)R(\theta_z) \quad (6.1)$$

$$R(\theta_x) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_x) & \sin(\theta_x) \\ 0 & -\sin(\theta_x) & \cos(\theta_x) \end{bmatrix}, \quad (6.2)$$

$$R(\theta_y) = \begin{bmatrix} \cos(\theta_y) & 0 & -\sin(\theta_y) \\ 0 & 1 & 0 \\ \sin(\theta_y) & 0 & \cos(\theta_y) \end{bmatrix}, \quad (6.3)$$

$$R(\theta_z) = \begin{bmatrix} \cos(\theta_z) & \sin(\theta_z) & 0 \\ -\sin(\theta_z) & \cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (6.4)$$

In practice, the point locations are affected by noise from the sensors or environment, resulting in accuracy reduction. To overcome this difficulty, we represent both point sets by kernel densities and match them as a whole instead of point-wise correspondences. The kernel densities are all set to be Gaussian. Each kernel corresponds to a point. The mean is the point location, and the covariance is initialized to be an identity matrix. As such, we represent the transformed $\mathcal{T}(U, \mathbf{x})$ and $\mathbf{Z}$ set by $\sum_{i=1}^M \frac{1}{M} \mathcal{N}_{h_U}(q; \mathcal{T}(\mathbf{u}_i), \Lambda_i)$ and $\sum_{j=1}^N \frac{1}{N} \mathcal{N}_{h_Z}(q; \mathbf{z}_j, \Gamma_j)$, respectively. Here h_U and h_Z denote their respective bandwidths and are initialized empirically. Though selecting an appropriate bandwidth has been widely studied in the literature, we noticed that the registration results are similar for most reasonable bandwidth values [71] for the problem at hand.

Based on the above kernel representations, an intuitive metric for assessing the registration results is the similarity between the two densities. Following [71], we choose the L2 distance as it is robust and has a closed-form expression:

$$d_{L_2}(U, Z, \mathbf{x}) = \int_{\mathbb{R}^3} (\mathcal{K}_Z(\mathbf{q}) - \mathcal{K}_{\mathcal{T}(U, \mathbf{x})}(\mathbf{q}))^2 d\mathbf{q}. \quad (6.5)$$

Point clouds are sequentially collected in Lidar-based SLAM. The registration process is conducted continuously and iteratively throughout the measuring time. Consequently, we can explicitly model the registration in Lidar-based SLAM as a state space model (SSM):

$$\mathbf{x}_t - \mathbf{x}_{t-1} = \mathbf{x}_{t-1} - \mathbf{x}_{t-2} + \boldsymbol{\delta}_{t-1}, \quad (6.6a)$$

$$\mathcal{K}_Z(\mathbf{q}) = \mathcal{K}_{\mathcal{T}(U, \mathbf{x}_t)}(\mathbf{q}) + \boldsymbol{\epsilon}_t, \quad \forall \mathbf{q} \in \mathbb{R}^D, \quad (6.6b)$$

where (6.6a) is the process model, and (6.6b) is the observation model. $\boldsymbol{\delta}_{t-1}$ is the process noise that follows the normal distribution $\mathcal{N}(\mathbf{0}; \Gamma_{t-1})$, and $\boldsymbol{\epsilon}_t$ is a Gaussian

process. Every finite collection of $\mathbf{q}$ from (6.6b) thus follows a multivariate normal distribution, and we denote it as $\mathcal{N}(\mathbf{0}; \Lambda_t)$.

Equivalently, the process model (6.6a) can be rewritten as

$$\mathbf{x}_t - \mathbf{x}_{t-1} = \mathbf{x}_{t-1} - \mathbf{x}_{t-2} + \boldsymbol{\delta}_{t-1}, \quad (6.7)$$

which describes a smooth motion from time $t - 2$ to t .

On the other hand, the observation model (6.6b) is interpreted from the L2 distance in (6.5), where $\mathbf{q}$ is a point in the point space.

For brevity, we further rewrite the above SSM in (6.6) as:

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{x}_{t-2}) + \boldsymbol{\delta}_{t-1} \quad (6.8a)$$

$$\mathbf{z}_t = h(\mathbf{x}_t) + \boldsymbol{\epsilon}_t, \quad (6.8b)$$

where f and h denote the motion and observation functions, $\mathbf{z}_t$ is referred to as the observation.

We denote the dimension of the observation vector as L . Then we further assume $\Gamma_{t-1}^{-1} = \boldsymbol{\alpha}_{t-1} \circ I_P$ and $\Lambda_t^{-1} = \boldsymbol{\beta}_t \circ I_L$. In particular, $\boldsymbol{\alpha}_{t-1}$ and $\boldsymbol{\beta}_t$ are two vectors, $\circ$ denotes a elementwise product, I_P and I_L are two identity matrices with size $P \times P$ and $L \times L$, respectively. As such, the motion and observation noise now are characterized by solely two vectors.

Taking a Bayesian probabilistic perspective, the above SSM in (6.6) can be expressed in a probabilistic way as follows:

$$\begin{aligned} & p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \boldsymbol{\alpha}_{t-1},) \\ & \propto \sqrt{\prod_{i=1}^P (\boldsymbol{\alpha}_t)_i} \exp \left\{ -\frac{1}{2} (\mathbf{x}_t - f(\mathbf{x}_{t-1}, \mathbf{x}_{t-2}))^T \text{diag}(\boldsymbol{\alpha}_{t-1}) (\mathbf{x}_t - f(\mathbf{x}_{t-1}, \mathbf{x}_{t-2})) \right\}, \end{aligned} \quad (6.9a)$$

$$\begin{aligned} & p(\mathbf{z}_t | \mathbf{x}_t, \boldsymbol{\beta}_t) \\ & \propto \sqrt{\prod_{j=1}^L (\boldsymbol{\beta}_t)_j} \exp \left\{ -\frac{1}{2} (\mathbf{z}_t - h(\mathbf{x}_t))^T \text{diag}(\boldsymbol{\beta}_t) (\mathbf{z}_t - h(\mathbf{x}_t)) \right\}. \end{aligned} \quad (6.9b)$$

Furthermore, the estimation of transformation $\mathbf{x}_t$ is converted into a posterior estimation problem:

$$p(\mathbf{x}_t | \mathbf{z}_t, \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \boldsymbol{\alpha}_{t-1}, \boldsymbol{\beta}_t) \propto p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \boldsymbol{\alpha}_{t-1}) p(\mathbf{z}_t | \mathbf{x}_t, \boldsymbol{\beta}_t). \quad (6.10)$$

6.3 Variational Bayesian Point Set Registration

We represented the point sets by kernel densities and further modelled the registration as a posterior estimation problem in Section 6.2. In this section, we will introduce the stochastic variational Bayesian inference to approximate the resulting posterior and hence estimate $\mathbf{x}_t$. Additionally, we adopt RQMC points, SVRG method, as well as variance reduction techniques to improve the computational efficiency. The proposed method is thus robust to noise, outliers, missing partial structures, and applicable to dense Lidar point clouds without increasing the computational cost too much.

6.3.1 Stochastic Variational Bayesian Inference

We described the registration problem by the SSM in (6.8) in Section 6.2. The state of the SSM in (6.8) can be approximated by filtering methods, such as EKF [58], UKF [184], PF [32], or MAP [8] techniques. Filtering estimates the full distribution of $\mathbf{x}_t$, while MAP [8] seeks the mode of the resulting posterior in (6.10). However, both filtering and MAP have limitations and are not well adapted to the problem here. Filtering methods have high computational complexity, inappropriate for the case here where the dimension of the observation vector L is quite large. Moreover, both filtering and MAP are subject to accurate noise priors. However, we have no prior knowledge about $\boldsymbol{\alpha}_{t-1}$ and $\boldsymbol{\beta}_t$. Furthermore, filtering and MAP often can not escape from poor solutions very nicely.

To overcome these challenges, we employ a stochastic variational Bayesian inference for approximation in this work. The intractable posterior in (6.10) is approximated by a simple variational distribution $q(\mathbf{x}_t)$ so that $p(\mathbf{x}_t | \mathbf{z}_t, \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \boldsymbol{\alpha}_{t-1}, \boldsymbol{\beta}_t) \approx q(\mathbf{x}_t)$. We restrict $q(\mathbf{x}_t)$ to be within the Gaussian family. Intuitively, the selection of $q(\mathbf{x}_t)$ should be always with the intention of making $q(\mathbf{x}_t)$ as similar to the true

posterior $p(\mathbf{x}_t | \mathbf{z}_t, \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \boldsymbol{\alpha}_{t-1}, \boldsymbol{\beta}_t)$ as possible. In most variational Bayes, the similarity is measured by the KL divergence:

$$KL(q|p) = - \int q(\mathbf{x}_t) \log \frac{p(\mathbf{x}_t | \mathbf{z}_t, \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \boldsymbol{\alpha}_{t-1}, \boldsymbol{\beta}_t)}{q(\mathbf{x}_t)} d\mathbf{x}_t. \quad (6.11)$$

Inference is performed through iteratively minimizing the KL divergence in (6.11) until the convergence is reached.

This minimization of KL divergence is equivalent to the maximization of the ELBO (see Appendix A):

$$\begin{aligned} \mathcal{L} &= \mathbb{E}_q \left[\log \frac{p(\mathbf{z}_t, \mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \boldsymbol{\alpha}_{t-1}, \boldsymbol{\beta}_t)}{q(\mathbf{x}_t)} \right], \\ &= \mathbb{E}_q [\log p(\mathbf{z}_t, \mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \boldsymbol{\alpha}_{t-1}, \boldsymbol{\beta}_t)] + H[q(\mathbf{x}_t)], \\ &= \mathbb{E}_q [\log p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \boldsymbol{\alpha}_{t-1}) p(\mathbf{z}_t | \mathbf{x}_t, \boldsymbol{\beta}_t)] + H[q(\mathbf{x}_t)], \end{aligned} \quad (6.12)$$

where $\mathbb{E}_q$ denotes the expectation over q distribution, and $H[q(\mathbf{x}_t)] = -\mathbb{E}_q[\log q(\mathbf{x}_t)]$ is the entropy of $q(\mathbf{x}_t)$.

Through maximizing $\mathcal{L}$, we can obtain $q(\mathbf{x}_t)$ as well as $\boldsymbol{\alpha}_{t-1}$, and $\boldsymbol{\beta}_t$. We perform the maximization by gradient ascent. However, the gradients of the expectation forms in (6.12) w.r.t. $\mathbf{x}_t$ are unavailable since the density $q(\mathbf{x}_t)$ is also parameterized by $\mathbf{x}_t$. To resolve this issue, we apply a reparameterization trick:

$$\mathbf{x}_t = C_t \mathbf{e} + \boldsymbol{\mu}_t, \quad (6.13)$$

where $\mathbf{e}$ is a P dimensional random vector following the standard normal distribution $\phi(\mathbf{e})$, $\boldsymbol{\mu}_t$ denotes a the real mean vector and the scale matrix C_k is a $P \times P$ lower triangular positive definite matrix. All the diagonal elements of C_t are strictly positive.

After the reparameterization trick, the $\mathcal{L}$ in (6.12) is elaborated by:

$$\mathcal{L} = \mathbb{E}_{\phi(\mathbf{e})} [\log p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \boldsymbol{\alpha}_{t-1}) p(\mathbf{z}_t | \mathbf{x}_t, \boldsymbol{\beta}_t)] + \log |C_t| + H[\phi(\mathbf{e})], \quad (6.14)$$

where $\log |C_t| = \sum_{d=1}^P \log(C_t)_{dd}$ and $H[\phi(\mathbf{e})]$ is the entropy of $\phi(\mathbf{e})$. Here $H[\phi(\mathbf{e})]$ is a constant.

Since the density $\phi(\mathbf{e})$ is independent of $\mathbf{x}_t$, the gradient of the expectations in (6.12) can now be expressed as expectations of the gradient directly. Provided that $\phi(\mathbf{e})$ is differentiable and permits a straightforward simulation of independent samples, we have the following gradients:

$$\nabla_{\boldsymbol{\mu}_t} \mathcal{L} = \mathbb{E}_{\phi(\mathbf{e})} [\nabla_{\boldsymbol{\mu}_t} (\log p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \boldsymbol{\alpha}_{t-1}) p(\mathbf{z}_k | \mathbf{x}_t, \boldsymbol{\beta}_t))], \quad (6.15a)$$

$$\nabla_{C_t} \mathcal{L} = \mathbb{E}_{\phi(\mathbf{e})} [\nabla_{C_t} (\log p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \boldsymbol{\alpha}_{t-1}) p(\mathbf{z}_k | \mathbf{x}_t, \boldsymbol{\beta}_t))] + \Delta_{C_t}, \quad (6.15b)$$

$$\nabla_{\boldsymbol{\alpha}_{t-1}} \mathcal{L} = \mathbb{E}_{\phi(\mathbf{e})} [\nabla_{\boldsymbol{\alpha}_{t-1}} (\log p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \boldsymbol{\alpha}_{t-1}) p(\mathbf{z}_k | \mathbf{x}_t, \boldsymbol{\beta}_t))], \quad (6.15c)$$

$$\nabla_{\boldsymbol{\beta}_t} \mathcal{L} = \mathbb{E}_{\phi(\mathbf{e})} [\nabla_{\boldsymbol{\beta}_t} (\log p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \boldsymbol{\alpha}_{t-1}) p(\mathbf{z}_k | \mathbf{x}_t, \boldsymbol{\beta}_t))], \quad (6.15d)$$

where Δ_{C_t} denotes a diagonal matrix with elements $[\frac{1}{C_{t11}}, \dots, \frac{1}{C_{tPP}}]$.

Though the exact gradients in (6.15) are of closed forms, they cannot be calculated analytically. We thus resort to Monte Carlo methods for approximation. We proceed by drawing a sample $\mathbf{e}^{(s)}$ from the distribution $\phi(\mathbf{e})$, and thus deterministically obtain a sample of $\mathbf{x}_t$ computed as $C_t \mathbf{e}^{(s)} + \boldsymbol{\mu}_t$. The expectation forms of the gradients in (6.15) are then approximated as follows:

$$\nabla_{\boldsymbol{\mu}_t} \mathcal{L} \approx \nabla_{\boldsymbol{\mu}_t} (\log p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \boldsymbol{\alpha}_{t-1}) p(\mathbf{z}_k | \mathbf{x}_t, \boldsymbol{\beta}_t)) \quad (6.16a)$$

$$\nabla_{C_t} \mathcal{L} \approx \nabla_{C_t} (\log p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \boldsymbol{\alpha}_{t-1}) p(\mathbf{z}_k | \mathbf{x}_t, \boldsymbol{\beta}_t)) + \Delta_{C_t} \quad (6.16b)$$

$$\nabla_{\boldsymbol{\alpha}_{t-1}} \mathcal{L} \approx \nabla_{\boldsymbol{\alpha}_{t-1}} (\log p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \boldsymbol{\alpha}_{t-1}) p(\mathbf{z}_k | \mathbf{x}_t, \boldsymbol{\beta}_t)) \quad (6.16c)$$

$$\nabla_{\boldsymbol{\beta}_t} \mathcal{L} \approx \nabla_{\boldsymbol{\beta}_t} (\log p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \boldsymbol{\alpha}_{t-1}) p(\mathbf{z}_k | \mathbf{x}_t, \boldsymbol{\beta}_t)). \quad (6.16d)$$

The stochastic gradients in (6.16a) are the unbiased estimations of the exact gradients in (6.15) [5, 15, 129]. To gain more intuition, we further express them as below:

$$\nabla_{\boldsymbol{\mu}_t} \mathcal{L} \approx -\boldsymbol{\alpha}_{t-1} \circ \Theta_t + (\boldsymbol{\beta}_t \circ \nabla_{\mathbf{x}_t} h(\mathbf{x}_t))^T \Psi_t \quad (6.17a)$$

$$\nabla_{C_t} \mathcal{L} \approx (-\boldsymbol{\alpha}_{t-1} \circ \Theta_t + (\boldsymbol{\beta}_t \circ \nabla_{\mathbf{x}_t} h(\mathbf{x}_t))^T \Psi_t) \circ \mathbf{e} + \Delta_{C_t} \quad (6.17b)$$

$$\nabla_{\boldsymbol{\alpha}_{t-1}} \mathcal{L} = \frac{1}{2} \mathbf{1} \oslash \boldsymbol{\alpha}_{t-1} - \frac{1}{2} \Theta_t \circ \Theta_t \quad (6.17c)$$

$$\nabla_{\boldsymbol{\beta}_t} \mathcal{L} = \frac{L}{2} \mathbf{1} \oslash \boldsymbol{\beta}_t - \frac{1}{2} \Psi_t \circ \Psi_t, \quad (6.17d)$$

where $\mathbf{1}$ is a vector with all ones, $\oslash$ is the elementwise division. Explicitly, $\Theta_t = \mathbf{x}_t - 2\mathbf{x}_{t-1} + \mathbf{x}_{t-2}$, and $\Psi_t = (\mathbf{z}_t - h(\mathbf{x}_t))$.

Algorithm 4: Stochastic variational Bayesian inference.

Data: $\mathbf{z}_t, \mathbf{x}_{t-1}, \mathbf{x}_{t-2}$ **Result:** $\mathbf{x}_t$ Initialize $\boldsymbol{\mu}_t^0, C_t^0, \boldsymbol{\alpha}_{t-1}^0, \boldsymbol{\beta}_t^0$, iteration number $k=0$;**while** *convergence criterion is not met* **do** $k=k+1$; $\mathbf{e} \sim \phi(\mathbf{e})$; $\mathbf{x}_t^{(k)} = C_t^{(k)} \mathbf{e} + \mathbf{u}_t^{(k)}$; For a variable $\boldsymbol{\nu}_t \sim \{\boldsymbol{\mu}_t, C_t, \boldsymbol{\alpha}_{t-1}, \boldsymbol{\beta}_t\}$, Calculate its stochastic gradient $(\nabla_{\boldsymbol{\nu}} \mathcal{L})^{(k)}$;; $\boldsymbol{\nu}_t^{(k+1)} = \boldsymbol{\nu}_t^{(k)} + \rho_t^{(k)} (\nabla_{\boldsymbol{\nu}} \mathcal{L})^{(k)}$;**end**

With the stochastic gradients in (6.17a), we can maximize the ELBO $\mathcal{L}$ in (6.12) in an iterative scheme. The whole stochastic approximation algorithm is summarized in Algorithm 4. According to the theories in [144], when a schedule of the learning rate ρ_t satisfies $\sum_k \rho_t^{(k)} = \infty$ and $\sum_k (\rho_t^{(k)})^2 < \infty$, the iteration scheme in Algorithm 4 is guaranteed to converge to a local maximum of $\mathcal{L}$ or to the global maximum if $\mathcal{L}$ is concave. In practice, we employ the ADAM [80] method to determine the learning rates adaptively.

6.3.2 Improvement of the Computational Efficiency

In previous sections, we proposed to model the point sets as kernel densities, describe the registration problem by a state space model, and solve the formulated SSM through stochastic variational Bayesian inference. Such a scheme ensures the accuracy and robustness in scenarios where there exist noise, outliers, and missing partial structures. Apart from the accuracy and robustness, the computational cost is also of great significance, especially in intelligent vehicle transportation, such as Lidar-based SLAM [59, 134, 138]. Though the inference in Section 6.3.1 has a linear computational complexity, the dimension of the observation vector is huge (L is infinite) and the number of points is extremely large. Consequently, the key bottlenecks that restrict high-performance computing are the complicated integration and density calculation in (6.6b). To resolve this issue, we resort to RQMC points, SVRG method, and variance reduction techniques.

First of all, the calculation of the L2 distance in (6.6b) is computationally demanding due to the complicated forms. To alleviate this problem, we approximate the integration by a set of RQMC points. Compared with the Monte Carlo points generated from a pseudorandom sequence, Quasi-Monte Carlo uses a low-discrepancy sequence to select approximation points. Consequently, RQMC points yields a more evenly distribution with the same or fewer numbers of points. Suppose that we sample L D -dimensional RQMC points $\mathbf{y} = \{\mathbf{y}_1, \dots, \mathbf{y}_L\}$. Since $\mathbf{y} \in (0, 1)^D$, we thus need to map them to the whole $\mathbb{R}^D$ space. Considering that points too far away from the centroid contribute little to the integration, we map $\mathbf{y}$ only to the point space (i.e., the space where points exist) through a mapping function g . As a result, we obtain a set of discrete points $g(\mathbf{y}) = \{g(\mathbf{y}_1), \dots, g(\mathbf{y}_L)\}$ that are evenly distributed in the point space. The observation model in (6.6b) is then rewritten as:

$$\mathcal{K}_Z(g(\mathbf{y})) = \mathcal{K}_{\mathcal{T}(U, \mathbf{x}_t)}(g(\mathbf{y})) + \boldsymbol{\eta}_t, \quad (6.18)$$

where $\boldsymbol{\eta}_t$ is the additive noise distributed as $\mathcal{N}(\mathbf{0}; \boldsymbol{\beta}_t \circ I_L)$.

Through the RQMC points, we successfully transform the infinite integration in (6.6b) into a finite sum of discrete points expressed as (6.18), and hence significantly reduce the calculation cost. The $\mathbf{z}_t$ and $h(\mathbf{x}_t)$ now are claculated as $\mathcal{K}_Z(g(\mathbf{y}))$ and $\mathcal{K}_{\mathcal{T}(U, \mathbf{x}_t)}(g(\mathbf{y}))$, respectively.

The kernel densities in (6.18) consist of finite sums. The number of kernels equals the number of points, which is extremely huge for calculation. We can reduce the computational burden resulting from the calculation of kernel densities by applying SVRG algorithm [73]. SVRG is a variance reduction method for objective functions with finite sums. It does not require explicit storage of gradients. To apply SVRG, we firstly elaborate the $\log p(\mathbf{z}_t | \mathbf{x}_t, \boldsymbol{\beta}_t)$ term in (6.12) as:

$$\log p(\mathbf{z}_t | \mathbf{x}_t, \boldsymbol{\beta}_t) = -\frac{1}{2} (\boldsymbol{\beta}_t \circ (\mathbf{z}_t - h(\mathbf{x}_t)))^T (\mathbf{z}_t - h(\mathbf{x}_t)) + \sum_{j=1}^L \log(\boldsymbol{\beta}_t)_i + L \log(2\pi). \quad (6.19)$$

The calculation bottleneck lies in term $-\frac{1}{2} (\boldsymbol{\beta}_t \circ (\mathbf{z}_t - h(\mathbf{x}_t)))^T (\mathbf{z}_t - h(\mathbf{x}_t))$, where each element in $\mathbf{z}_t$ is a sum of N components, and each element of $h(\mathbf{x}_t)$ is a sum

of M components. We further rewrite it as:

$$\begin{aligned}
-\frac{1}{2} (\boldsymbol{\beta}_t \circ (\mathbf{z}_t - h(\mathbf{x}_t)))^T (\mathbf{z}_t - h(\mathbf{x}_t)) &= -\frac{1}{2} \sum_{i=1, j=1}^{i=M, j=M} (\boldsymbol{\beta}_t \circ (\mathbf{z}_t)_i)^T (\mathbf{z}_t)_j \\
&\quad + \sum_{i=1, j=1}^{i=M, j=N} (\boldsymbol{\beta}_t \circ (\mathbf{z}_t)_i)^T (\mathbf{h}(\mathbf{x}_t))_j \\
&\quad - \frac{1}{2} \sum_{i=1, j=1}^{i=N, j=N} (\boldsymbol{\beta}_t \circ (\mathbf{h}(\mathbf{x}_t))_i)^T (\mathbf{h}(\mathbf{x}_t))_j,
\end{aligned} \tag{6.20}$$

where $(\mathbf{z}_t)_i$ is a vector containing density values from the i^{th} component, and each sum term is calculated and updated via the SVRG technique, as indicated in Algorithm 5.

Explicitly, let us denote one sum in (6.20) as $\kappa = \sum_{i=1}^V \kappa_i$. In SVRG, instead of calculating the entire sums, we randomly select a batch with size W to approximate. To make the batch distribute as evenly as possible, we apply another RQMC sampling. We sample a W -size set of points $s = s_1, \dots, s_W$ from a one-dimensional space, and then map this set to integers located within $[0, V]$. The resulting subset of indices for term κ is referred to as J_κ .

Algorithm 5: SVRG algorithm for term κ

Result: φ , where $\varphi \in \{\boldsymbol{\mu}_t, C, \boldsymbol{\beta}_t\}$

initialize φ^0 ;

for $s = 1, 2, \dots$ **do**

$\tilde{\varphi} = \tilde{\varphi}_{s-1}$;

$\tilde{\zeta} = \frac{1}{V} \sum_{i=1}^V \nabla_{\kappa_i}(\tilde{\varphi})$;

$\varphi = \tilde{\varphi}$;

for $k \leftarrow 1$ **to** $m = \frac{V}{W}$ **do**

$k = k + 1$;

 Randomly select a batch J_κ ;

$\varphi^{(k+1)} = \varphi^{(k)} + \rho_t^{(k)} (\nabla_{\kappa_{J_\kappa}}(\varphi^{(k)}) - \nabla_{\kappa_{J_\kappa}}(\tilde{\varphi}) + \tilde{\zeta} + \vartheta)$, where ϑ is the gradient of other terms in $\mathcal{L}$

end

end

SVRG can significantly reduce the computational cost resulting from the calculation of a large number of kernels. However, this SVRG approximation may also

lead to an increase in the variance of the stochastic gradients. To address this issue, we further employ control variates and dismiss the score functions in the calculation of gradients for reducing the variances and, consequently, improving the convergence speed.

6.4 Experimental Results

In this section, we present experimental results on the application of the proposed registration method to point clouds extracted from KITTI [49] Velodyne dataset. Quantitative comparisons with the state-of-the-art registration methods, such as ICP [10], NDT [11], CPD [125], CSCIF [94], and GMM-L2 [71], are also presented. The robustness of the proposed approach to noise, outliers, and missing partial structuresⁱ is explored.

6.4.1 Experiment Setting

The test is based on the widely used KITTI Velodyne dataset [49]. KITTI dataset contains 11 Velodyne sequences, we evenly select 10 frames from each to serve as the initial frames. In order to simulate the true transformations in KITTI, we set the initial velocity and angular velocity as $[0.8, 0.1, 0.8]\text{m}$ and $[0.002, 0.002, 0.002]\text{rad}$ per frame. The initial velocity and angular velocity acceleration are set to be $[0.001, 1e-4, 0.001]\text{m/frame}^2$ and $[0.0001, 0.0001, 0.0001]\text{rad/frame}^2$. To avoid decreased frame quality caused by over-transformation, we transform each frame only for 10 time steps according to (6.6a). The information vector α_{t-1} in (6.6a) is set to be $[1/0.16^2, 1/0.02^2, 1/0.16^2, 1/0.0002^2, 1/0.0002^2, 1/0.0002^2]$.

The acquired point sets are accomplished by position noise at every time step. We define the noise level by the ratio of the noise amplitude to the absolute values of the point locations. In this case, it is set to be 20%. Apart from the noise, the outliers and missing partial structures may also be present in practical scenarios. Therefore, we consider three scenarios: 1) noise; 2) noise + outliers; 3) noise + missing partial structures. Furthermore, we subdivide scenario 2) and 3) into

ⁱIn practice, Lidar may produce sparse points when occlusion occurs, or the object is too far away, and the scanned geometry is incomplete and contains lost structures.

three sub-scenarios. For instance, for case 2), we have three sub-cases: 2.a) noise + the moving set contains outliers; 2.b) noise + the fixed set contains outliers; 2.c) outliers + both set contain noise. Consequently, we have 7 cases in total to test.

Specifically, we set the number of additive outliers as 10,000 for scenario 2). Following [92–95], we first randomly select and copy 10,000 points from the original set, and then add Gaussian noise to their positions. The standard deviations are sampled uniformly from $(0, 10)$. We refer to the produced points as outliers and add these outliers to the point set. The lost structures are usually caused by three factors: the occlusion, the distance, and the sensor property. In particular, the occlusion will cause the loss of points in a specific area. Moreover, when the sensor is moving away from the scene, the generated points are gradually and uniformly becoming sparse. Last, sometimes points are randomly missing due to the inherent property of the Lidar sensor. To simulate the actual missing structures, we delete a portion of points by combining various point-omitting strategies, including deleting points uniformly and randomly, removing points in a specific area. Explicitly, we first select a random area and omit the points in this area. Next, we rasterize the point space and delete points uniformly and randomly. The number of omitting points is set to be 30,000.

6.4.2 Evaluation Results

We compare six algorithms: classical ICP, NDT, CPD, GMM_L2, CSCIF, and the proposed KD-VB-PSR under the setting in Section 6.4. The grid size of NDT is set to be 0.5m. We downsample the point sets by a grid filter with size 1m for CPD as it requires high computing memory. Both GMM_L2 and CSCIF are not directly applicable to large datasets because of the high computational complexity. As a remedy, we downsample the point sets through grid filters with a size of 2m and 6m for GMM_L2 and CSCIF, respectively.

The comparison between these algorithms in terms of noise, outliers, and missing partial structures is presented in Figure 6.2. The errors shown here are the average RMSE across the 110 datasets.

FIGURE 6.2: Average translation and rotation angles RMSE of ICP, NDT, and CPD in scenarios where exist noise, outliers, and missing partial structures.

FIGURE 6.2: Average translation and rotation angles RMSE of GMM_L2, CSCIF, and KD-VB-PSR in scenarios where exist noise, outliers, and missing partial structures.

From Figure 6.2, we can draw the following conclusions:

- The proposed KD-VB-PSR achieves the best accuracy among all the algorithms. The performance does not change much under various scenarios. As

a consequence, the proposed KD-VB-PSR shows robustness to noise, outliers, and missing partial structures.

- Both NDT and GMM_L2 are insensitive to the disturbance caused by noise, outliers, and missing partial structures because they are probability-based. However, the curves of CPD vary from each other, perhaps owing to solely one point cloud representing by the probability.
- ICP works fine when only noise occurs. However, it is sensitive to outliers and missing partial structures.
- The performance of CPD, GMM_L2 and CSCIF degrades due to downsampling. Therefore, those methods are inappropriate for points of large size.
- The proposed KD-VB-PSR is superior to NDT, perhaps due to its ability to escape from local extrema.

We evaluate the average registration time per frame of all the six algorithms. The results are presented in Table 6.1. We can observe that ICP is the most efficient algorithm. NDT is noticeably slower than ICP because of the interpolation. Despite the downsampling, CPD, GMM_L2 and CSCIF are still computationally demanding. The complicated cost functions of CPD and GMM_L2 lead to their slow speed. CSCIF has a cubic computational complexity caused by the inversion of the uncertainty matrix. Therefore, from the perspective of computational efficiency, CPD, GMM_L2 and CSCIF are not suitable for large point sets even with downsampling, either.

6.5 Conclusion

To address the point registration problem in Lidar-based SLAM, we presented an algorithm based on kernel densities and variational Bayesian inference in this chapter. We employed Gaussian kernel densities to represent the point sets, where the centroids are the point locations, and the initial covariances are all identity matrices. We further modelled the registration process as a state space model since Lidar-based SLAM is a sequential system. The relative transformation is encoded as the hidden variable, and the observation is the L2 distance between two kernel

TABLE 6.1: Average runtime per frame for ICP, NDT, CPD, GMM_L2, CSCIF, and KD-VB-PSR in scenarios where exist noise, outliers, and missing partial structures.

Algorithm	ICP	NDT	CPD	GMM_L2	CSCIF	KD-VB-PSR
Noise	1.08 s	44.59 s	352.67 s	766.21 s	667.90 s	12.26 s
Noise+the moving set contains noise	15.23 s	36.75 s	502.23 s	1356.40 s	1100.30 s	11.70 s
Noise+the fixed set contains noise	1.00 s	23.51 s	478.61 s	1571.90 s	1280.10 s	8.88 s
Noise+both set contain noise	7.06 s	23.51 s	529.70 s	3134.30 s	1690.70 s	9.12 s
Noise+the moving set contains missing partial structures	1.00 s	41.53 s	262.09 s	565.60 s	588.30 s	11.03 s
Noise+the fixed set contains missing partial structures	2.11 s	39.94 s	289.73 s	578.10 s	602.40 s	10.86 s
Noise+both set contain missing partial structures	1.81 s	35.29 s	168.42 s	410.90 s	476.80 s	10.09 s

densities. This state space model was solved under a stochastic variational Bayesian inference framework. In the implementation, we adopted RQMC points in the L2 integration, and applied the SVRG approach in the calculations of kernel densities, to reduce the computational cost. Since the approximation may increase the variances of the gradients, we employed variance reduction techniques to minimise the variances and further improve the convergence speed. The resulting algorithm is accurate and robust but remains a real-time speed. We validated its effectiveness and efficiency on Velodyne data extracted from the KITTI dataset under various challenging conditions, such as noise, outliers, and missing partial structures. The demonstrated advantages have shown its potential for further deployment on real robotic platforms targeted at specific missions.

Chapter 7

Conclusion & Future Work

7.1 Conclusion

We developed a set of variational Bayesian inference algorithms for visual and Lidar-based SLAM in this thesis, including online and offline SLAM estimation, variational Bayesian point set registration based on Gaussian Mixture Models and kernel densities. We made both theoretical and practical contributions.

The online solution proposed in Chapter 3 addresses the estimation of the most recent vehicle pose in SLAM problems. From the Bayesian perspective, this estimation involves the evaluation of the posterior over the current pose and the local map. Existing solutions are limited by their high computational complexity and unguaranteed convergence. Moreover, most of them are sensitive to noise priors. To overcome these challenges, we proposed a stochastic variational Bayesian inference (online SVBI) for approximation. Instead of evaluating the posterior directly, the proposed online SVBI alternatively employs a Gaussian distribution to approach the true probability. The uncertainty matrix of this Gaussian distribution is diagonal, resulting in a linear computational complexity. The approximation scheme is based on stochastic gradients, which can help the algorithm to escape from local extrema. We updated the noise values adaptively to minimize the effect caused by inaccurate priors. The online SVBI is thus robust to inaccurate noise pre-estimates. Extensive experimental results conducted on synthetic and real-life datasets demonstrated that the proposed online SVBI outperforms the state-of-the-art online algorithms in terms of accuracy, efficiency and robustness.

Apart from the online formulation, SLAM can also be addressed via offline techniques. Offline SLAM estimates the entire vehicle trajectory as well as the global map. Among all the offline algorithms, graph-based SLAM is the state-of-the-art. Its computational complexity is linear w.r.t. the number of features, however, cubical with the number of frames. Moreover, the performance of graph-based SLAM is heavily affected by the noise priors. To resolve these issues, we developed an offline algorithm based on stochastic variational Bayesian inference (offline SVBI) in Chapter 4. Similarly, sensor data observed from the vehicle is described via a conditional probability distribution, according to the Bayesian theory. The offline SVBI approximates the unknown posterior by a Gaussian distribution, the uncertainty of which is tri-diagonal. The antidiagonals encode the internal relations between frames. The resulting computational complexity is thus linear with the number of features as well as the number of frames. The environmental map is global and estimated together with the posterior. Moreover, we update the noise variances adaptively. As a result, the offline SVBI is robust to noise priors. The optimization still relies on the stochastic gradient ascent, and consequently, the offline SVBI can escape from local extrema. The experiments conducted on synthetic and real-life datasets have demonstrated that the offline SVBI has linear complexity, improved accuracy, and enhanced robustness. Compared to the graph-based SLAM, it is more scalable to SLAM problems where the number of frames is large.

A novel framework for point set registration in Lidar-based SLAM was proposed in Chapter 5. It describes the point sets as Gaussian Mixture Models (GMMs) and represents the registration problem by a state space model. The hidden state is the relative transformation to be estimated, and the observations are the L2 distance between the densities of the two GMMs. We employ the stochastic variational Bayesian inference to solve the state space model. The noise variances are estimated adaptively during the updates. As a result, the resulting variational Bayesian point set registration based on GMMs (GMMs-VB-PSR) retains a linear computational complexity and is robust to noise and local extrema. Furthermore, we approximate the complicated integration of the L2 distance by a set of quadrature points to improve the registration speed. The test conducted on synthetic 3D data showed that GMMs-VB-PSR has a comparable speed to the most efficient ICP. In addition, GMMs-VB-PSR produced more reliable results in the simulations when compared to the common registration algorithms.

The GMMs-VB-PSR proposed in Chapter 5 is heavily dependant on distinct groups or meaningful separations. However, there exist scenarios where the data can not be fully represented through clustering. To address this issue, we proposed a robust and computationally efficient registration method based on kernel densities (KD-VB-PSR) in Chapter 6. It is applicable to any general point sets. Considering the sequential Lidar-based SLAM system, we stick to the state space model for describing the registration problem. We employ kernel densities to represent the point sets. Each kernel is a Gaussian density function, the mean of which is the point centroid and the covariance is initially as an identity matrix. As a result, no extra clustering is required. We refer to the transformation as the state, and the L2 distance between two densities as the observations. The approximation of the transformation relies on the stochastic variational Bayesian inference, leading to a linear computational complexity and an improved capacity to escape from local extrema. To reduce the computational cost, we adopt Randomized Quasi-Monte-Carlo (RQMC) points to approximate the integration in the L2 distance, and SVRG algorithm to simplify the calculation of the gradients. Furthermore, variance reduction techniques are employed to contribute to a faster convergence speed. Tests conducted on Velodyne data from KITTI dataset demonstrated that the proposed KD-VB-PSR produced more consistent estimations in terms of noise, outliers and missing partial structures.

7.2 Future Work

The theoretical analysis and practical experiments presented in this thesis have demonstrated the advantages of applying variational Bayesian inference in visual and Lidar-based SLAM tasks. The successes of these methods attract us to make more exploration and in-depth study. The following are the future steps.

- **Real robotic platforms**, such as aerial drones, ground vehicles, and IoT (Internet of Things) devices, have been applied in various tasks like transportation, military affairs, manufacturing, among others. With the rapid development of electronic technology, mobile robotics have evolved substantially in the last few decades. The hardware is either custom-designed or fully integrated, with dedicated equipment including robust mechanics, advanced sensors, powerful processing units, and expansion ports. The software

systems and hardware platforms need to be integrated in order to test and verify the performance in a designated task fully. In this thesis, we have proposed variational Bayesian SLAM algorithms and validated on both synthetic and real-world data. These algorithms have shown exceptional performance, and we intend to apply them to richer variants of the SLAM problem. For instance, we plan to execute and assess the SLAM navigation on our university campus with unmanned grounded vehicles (UGVs) or unmanned aerial vehicles (UAVs). According to the available platforms and environmental conditions (e.g., indoor or outdoor, dynamic or static, light condition, obstacles), we will modify the proposed SLAM algorithms, integrate them into the software platform, and then control the robot movement to sweep the unexplored environment.

- **Multi-sensor fusion** addresses the combination of sensor data from multiple sources. Compared to single sensors, sensor fusion can achieve lower errors and higher reliability. It has long been a classic means for performance improvement in robotics tasks such as SLAM, visual odometry, and ego-motion estimation. Conventional fusion techniques are based on filtering. However, the growing uncertainty concerning the initial reference pose, as well as the global unobservability of the yaw angle and robot position are intrinsic to the fusion problem; this poses a challenge to the linearization-based filtering approaches. Another significant difficulty in designing the Kalman filters for fusion tasks lies in the unknown noise variances. Inaccurate noise priors may seriously reduce the Kalman filters' performance, and further make them inconsistent and divergent. The recently proposed optimization-based visual-inertial frameworks rely on nonlinear least-squares minimization. Despite the performance improvement, they are still limited in the noise priors and initial estimates. Moreover, the optimization generally has a higher computational cost. The proposed stochastic variational Bayesian inference has shown its great advantages on efficiency and robustness. It has linear complexity and estimates noise adaptively. Therefore, we can expect its applications in the multi-sensor tasks, especially in low power and real-time systems.

Data association is defined as associating a set of sensor observations with the environmental map elements. In the Bayesian paradigm, the feature-based SLAM solutions comprise two separate steps: determine the associations and estimate the vehicle poses and the environmental map given the

associations. The estimated pose correlates the association results. Provided m features and n observations, there are nm possible associations. Picking the wrong one can result in catastrophic consequences, such as large error and system divergence. The proposed online and offline SVBI assume the known data association. However, since both of them produce full distributions, we can apply the covariances in making association decisions. Explicitly, establish the associations that are most consistent with the collected evidence given reliable distributions of the vehicle pose. Furthermore, we can advocate a fully integrated Bayesian framework for SLAM under data association uncertainty and unknown feature number. Consequently, the resulting frame is not only applicable to scenarios where the association information is unknown but robust to the association ambiguity. Last, we can also apply the uncertainty matrices produced by online or offline SVBI to the loop closing task.

- **Point Completion Network** focuses on the problem of reconstructing the scene or objects from partial point clouds. This reconstruction lies at the core of a large amount of computer vision and robotics applications. Real-world 3D data, e.g., Velodyne data, are often not sufficient to fully characterize the geometric and semantic information due to missing points or partial shape. Existing shape completion methods can be grouped into geometry-based, alignment-based and learning-based. Geometry-based approaches require no external input and rely on geometric cues to recover the complete shapes. However, they can only handle relatively small gaps where the missing structures can be inferred directly. Alignment-based approaches match the partial input with deformable templates. Their performance is heavily dependant on the perfect matches. Moreover, they are prone to fail if there exists noise or a certain number of wrong correspondences between the input data and the template. Furthermore, the optimizations in alignment-based algorithms are computationally expensive. Hence alignment-based methods are usually impractical for online applications. Learning-based approaches complete the partial shapes by learning a parameterized reconstruction model (often a deep neural network). The learned model is the generalization inferred from the training dataset. Once new structures received, they can be passed through the learned model directly to produce the complete shapes. However, most learning-based approaches represent shapes as voxels, inconvenient for

preserving complete geometric information. A real-time point cloud shape completion can significantly contribute to the registration performance and map quality in Lidar-based SLAM.

Appendix A

Proofs for Part I

In this Appendix, we first provide details on the derivations needed for the ELBO and its equivalence to KL divergence. Also, the detailed derivations of the exact gradients are summarized as well.

A.1 Derivation of ELBO and its equivalence to KL divergence

The loglikelihood in (3.7) can be calculated as:

$$\begin{aligned}\log p(\mathbf{z}_k | \mathbf{x}_{k-1}, \boldsymbol{\alpha}_k, \boldsymbol{\beta}_k) &= \log \int p(\mathbf{x}_k, \mathbf{z}_k | \mathbf{x}_{k-1}, \boldsymbol{\alpha}_k, \boldsymbol{\beta}_k) \frac{q(\mathbf{x}_k)}{q(\mathbf{x}_k)} d\mathbf{x}_k \\ &= \log \left(\mathbb{E}_q \left[\frac{p(\mathbf{x}_k, \mathbf{z}_k | \mathbf{x}_{k-1}, \boldsymbol{\alpha}_k, \boldsymbol{\beta}_k)}{q(\mathbf{x}_k)} \right] \right).\end{aligned}\quad (\text{A.1})$$

According to Jensen's inequality, the lower bound of the above expression in (A.1) is called ELBO and expressed as (3.8).

On the other hand, the KL divergence of $q(\mathbf{x}_k)$ from the posterior p is defined as:

$$\text{KL}(q||p) = - \int q(\mathbf{x}_k) \log \left\{ \frac{p(\mathbf{x}_k | \mathbf{x}_{k-1}, \mathbf{z}_k, \boldsymbol{\alpha}_k, \boldsymbol{\beta}_k)}{q(\mathbf{x}_k)} \right\} d\mathbf{x}_k. \quad (\text{A.2})$$

More concretely, (A.2) can be rewritten as:

$$\mathbf{KL}(q||p) = - \int q(\mathbf{x}_k) \log \left\{ \frac{p(\mathbf{x}_k, \mathbf{z}_k | \mathbf{x}_{k-1}, \boldsymbol{\alpha}_k, \boldsymbol{\beta}_k)}{q(\mathbf{x}_k)} \right\} d\mathbf{x}_k + \int q(\mathbf{x}_k) \log p(\mathbf{z}_k | \boldsymbol{\beta}_k) d\mathbf{x}_k, \quad (\text{A.3})$$

where the first half is just ELBO $\mathcal{L}$ in (3.8), whereas the second half equals $\log p(\mathbf{z}_k | \boldsymbol{\beta}_k)$.

By rearranging (A.2), we obtain:

$$\mathcal{L} = \log p(\mathbf{z}_k | \boldsymbol{\beta}_k) - \mathbf{KL}(q||p), \quad (\text{A.4})$$

which reveals that minimizing the KL divergence amounts to maximizing the ELBO $\mathcal{L}$ in (3.8).

A.2 Derivation of exact gradients

The exact gradients of $\mathcal{L}$ w.r.t. $\boldsymbol{\mu}_k$, U_k , $\boldsymbol{\alpha}_k$, and $\boldsymbol{\beta}_k$ are calculated as:

$$\nabla_{\boldsymbol{\mu}_k} \mathcal{L} = \mathbb{E}_{\phi(e)} [\nabla_{\boldsymbol{\mu}_k} \log p(\mathbf{z}_k, \mathbf{x}_k | \mathbf{x}_{k-1}, \boldsymbol{\alpha}_k, \boldsymbol{\beta}_k)], \quad (\text{A.5a})$$

$$\nabla_{U_k} \mathcal{L} = \mathbb{E}_{\phi(e)} [\nabla_{U_k} \log p(\mathbf{z}_k, \mathbf{x}_k | \mathbf{x}_{k-1}, \boldsymbol{\alpha}_k, \boldsymbol{\beta}_k)] - \nabla_{U_k} \log |U_k|, \quad (\text{A.5b})$$

$$\nabla_{\boldsymbol{\alpha}_k} \mathcal{L} = \mathbb{E}_{\phi(e)} [\nabla_{\boldsymbol{\alpha}_k} \log p(\mathbf{x}_k | \mathbf{x}_{k-1}, \boldsymbol{\alpha}_k)], \quad (\text{A.5c})$$

$$\nabla_{\boldsymbol{\beta}_k} \mathcal{L} = \mathbb{E}_{\phi(e)} [\nabla_{\boldsymbol{\beta}_k} \log p(\mathbf{z}_k | \mathbf{x}_k, \boldsymbol{\beta}_k)], \quad (\text{A.5d})$$

where $\nabla_{U_k} \mathcal{L}$ in (A.5b) is a partial derivative w.r.t. the diagonal matrix U_k .

We denote $-\frac{1}{2}(\boldsymbol{\alpha}_k \circ (\mathbf{x}_k - f(\mathbf{x}_{k-1})))^T (\mathbf{x}_k - f(\mathbf{x}_{k-1}))$ as Ψ , and $-\frac{1}{2}(\boldsymbol{\beta}_k \circ (\mathbf{z}_k - h(\mathbf{x}_k)))^T (\mathbf{z}_k - h(\mathbf{x}_k))$ as Θ . As such, with PDF expressed in (3.5), the expressions in (A.5) can be rewritten as:

$$\nabla_{\boldsymbol{\mu}_k} \mathcal{L} = \mathbb{E}_{\phi(e)} [\nabla_{\boldsymbol{\mu}_k} \Psi + \nabla_{\boldsymbol{\mu}_k} \Theta], \quad (\text{A.6a})$$

$$\nabla_{U_k} \mathcal{L} = \mathbb{E}_{\phi(e)} \left[\nabla_{U_k} \Psi + \nabla_{U_k} \Theta \right] - \nabla_{U_k} \log |U_k|, \quad (\text{A.6b})$$

$$\nabla_{\boldsymbol{\alpha}_k} \mathcal{L} = \mathbb{E}_{\phi(e)} \left[\nabla_{\boldsymbol{\alpha}_k} \log \sqrt{\prod_{i=1}^D (\boldsymbol{\alpha}_k)_i} + \nabla_{\boldsymbol{\alpha}_k} \Psi \right]$$

$$= \mathbb{E}_{\phi(\mathbf{e})} [\nabla_{\boldsymbol{\alpha}_k} \Psi] + \frac{1}{2} \mathbf{1} \oslash \boldsymbol{\alpha}_k \quad (\text{A.6c})$$

$$\begin{aligned} \nabla_{\boldsymbol{\beta}_k} \mathcal{L} &= \mathbb{E}_{\phi(\mathbf{e})} \left[\nabla_{\boldsymbol{\beta}_k} \log \sqrt{\prod_{j=1}^M (\boldsymbol{\beta}_k)_j} + \nabla_{\boldsymbol{\beta}_k} \Theta \right] \\ &= \mathbb{E}_{\phi(\mathbf{e})} [\nabla_{\boldsymbol{\beta}_k} \Theta] + \frac{1}{2} \mathbf{1} \oslash \boldsymbol{\beta}_k. \end{aligned} \quad (\text{A.6d})$$

Based on the chain rules $\nabla_{\boldsymbol{\mu}_k} \mathcal{L} = \nabla_{\mathbf{x}_k} \mathcal{L}$, $\nabla_{U_k^{-1}} \mathcal{L} = \nabla_{\mathbf{x}_k} \mathcal{L} \cdot \mathbf{e}^T$ and $\nabla_{U_k} \mathcal{L} = -U_k^{-T} (\nabla_{U_k^{-1}} L) U_k^{-T}$, we can expand (A.6) to the expressions in (3.11).

List of Author's Publications

Journal Articles

- **Xiaoyue Jiang**, Michael Hoy, Hang Yu and Justin Dauwels, “A Linear-complexity Approach for Visual SLAM Problem: Stochastic Variational Bayesian Inference,” *IEEE Transactions on Intelligent Transportation Systems*, 2019 [submitted].

Conference Proceedings

- **Xiaoyue Jiang**, Michael Hoy, Hang Yu and Justin Dauwels, “Linear-complexity Stochastic Variational Bayes Inference for SLAM,” in *IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1-6, Oct. 2017.
- **Xiaoyue Jiang**, Hang Yu, Michael Hoy and Justin Dauwels, “Robust Linear-complexity Approach to Full SLAM Problems: Stochastic Variational Bayes Inference,” in *IEEE 90th Vehicular Technology Conference (VTC)*, Sep. 2019.
- **Xiaoyue Jiang**, Hang Yu, Michael Hoy and Justin Dauwels, “Variational Bayesian Point Set Registration,” in *IEEE 90th Vehicular Technology Conference (VTC)*, Sep. 2019.

Bibliography

- [1] Emile HL Aarts and Jan HM Korst. Boltzmann machines and their applications. In *International Conference on Parallel Architectures and Languages Europe*, pages 34–50. Springer, 1987.
- [2] Nikolay Atanasov, Jerome Le Ny, Kostas Daniilidis, and George J Pappas. Decentralized active information acquisition: Theory and application to multi-robot SLAM. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4775–4782. IEEE, 2015.
- [3] Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. Deep learning for hate speech detection in tweets. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 759–760, 2017.
- [4] Tim Bailey, Juan Nieto, Jose Guivant, Michael Stevens, and Eduardo Nebot. Consistency of the EKF-SLAM algorithm. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3562–3568. IEEE, 2006.
- [5] Jack Baker, Paul Fearnhead, Emily B Fox, and Christopher Nemeth. Control variates for stochastic gradient MCMC. *Statistics and Computing*, 29(3):599–615, 2019.
- [6] Ian Baldwin. Road vehicle localization with 2D push-broom LIDAR and 3D priors. In *2012 IEEE International Conference on Robotics and Automation*, pages 2611–2617. IEEE, 2012.
- [7] Ian Baldwin and Paul Newman. Laser-only road-vehicle localization with dual 2D push-broom Lidars and 3D priors. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2490–2497. IEEE, 2012.

- [8] Robert Bassett and Julio Deride. Maximum a posteriori estimators as a limit of Bayes estimators. *Mathematical Programming*, 174(1-2):129–144, 2019.
- [9] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European Conference on Computer Vision*, pages 404–417. Springer, 2006.
- [10] Paul J Besl and Neil D McKay. Method for registration of 3D shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, volume 1611, pages 586–606. International Society for Optics and Photonics, 1992.
- [11] Peter Biber and Wolfgang Straßer. The normal distributions transform: A new approach to laser scan matching. In *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)(Cat. No. 03CH37453)*, volume 3, pages 2743–2748. IEEE, 2003.
- [12] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.
- [13] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, 112(518):859–877, 2017.
- [14] Léon Bottou. Stochastic gradient learning in neural networks. *Proceedings of Neuro-Nimes*, 91(8):12, 1991.
- [15] J Bourrelly. Parallelization of a neural network learning algorithm on a hypercube. *Hypercube and Distributed Computers*. Elsevier Science Publishing, 1989.
- [16] Brendan Burns and Oliver Brock. Sampling-based motion planning with sensing uncertainty. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 3313–3318. IEEE, 2007.
- [17] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The EuRoC micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35(10):1157–1163, 2016.

- [18] Richard H Byrd, Samantha L Hansen, Jorge Nocedal, and Yoram Singer. A stochastic quasi-Newton method for large-scale optimization. *SIAM Journal on Optimization*, 26(2):1008–1031, 2016.
- [19] Henry Carrillo, Philip Dames, Vijay Kumar, and José A Castellanos. Autonomous robotic exploration using occupancy grid maps and graph slam based on shannon and rényi entropy. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 487–494. IEEE, 2015.
- [20] Tom Carter. An introduction to information theory and entropy. *Complex Systems Summer School, Santa Fe*, 2007.
- [21] José A Castellanos, José Neira, and Juan D Tardós. Limits to the consistency of EKF-based slam. *IFAC Proceedings Volumes*, 37(8):716–721, 2004.
- [22] Koray Celik, Soon-Jo Chung, and Arun Somani. Mono-vision corner SLAM for indoor navigation. In *2008 IEEE International Conference on Electro/Information Technology*, pages 343–348. IEEE, 2008.
- [23] Dmitry Chetverikov, Dmitry Svirkov, Dmitry Stepanov, and Pavel Krsek. The trimmed iterative closest point algorithm. In *Object Recognition Supported by User Interaction for Service Robots*, volume 3, pages 545–548. IEEE, 2002.
- [24] Chiang-Heng Chien, Chen-Chien Hsu, Wei-Yen Wang, Wen-Chung Kao, and Chiang-Ju Chien. FPGA-implemented corner feature extracting simultaneous localization and mapping. In *2016 IEEE 6th International Conference on Consumer Electronics-Berlin (ICCE-Berlin)*, pages 98–99. IEEE, 2016.
- [25] Margarita Chli and Andrew J Davison. Active matching for visual tracking. *Robotics and Autonomous Systems*, 57(12):1173–1187, 2009.
- [26] Young-Ho Choi, Tae-Kyeong Lee, and Se-Young Oh. A line feature based SLAM with low grade range sensors using geometric constraints and active exploration for mobile robot. *Autonomous Robots*, 24(1):13–27, 2008.
- [27] Haili Chui and Anand Rangarajan. A feature registration framework using mixture models. In *Proceedings IEEE Workshop on Mathematical Methods in Biomedical Image Analysis. MMBIA-2000 (Cat. No. PR00737)*, pages 190–197. IEEE, 2000.

- [28] Alan Anthony Coelho. Optimum Levenberg–Marquardt constant determination for nonlinear least-squares. *Journal of Applied Crystallography*, 51(2):428–435, 2018.
- [29] Amaury Dame, Victor A Prisacariu, Carl Y Ren, and Ian Reid. Dense reconstruction using 3D object shape priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1288–1295, 2013.
- [30] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):1052–1067, 2007.
- [31] Paloma de la Puente and Diego Rodríguez-Losada. Feature based graph-SLAM in structured environments. *Autonomous Robots*, 37(3):243–260, 2014.
- [32] Pierre Del Moral. Non-linear filtering: interacting particle resolution. *Markov processes and related fields*, 2(4):555–581, 1996.
- [33] Frank Dellaert and Michael Kaess. Square Root SAM: Simultaneous localization and mapping via square root information smoothing. *The International Journal of Robotics Research*, 25(12):1181–1203, 2006.
- [34] Tue-Cuong Dong-Si and Anastasios I Mourikis. Motion tracking with fixed-lag smoothing: Algorithm and consistency analysis. In *2011 IEEE International Conference on Robotics and Automation*, pages 5655–5662. IEEE, 2011.
- [35] Ethan Eade and Tom Drummond. Scalable monocular SLAM. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Volume 1*, pages 469–476. IEEE Computer Society, 2006.
- [36] Felix Endres, Jürgen Hess, Nikolas Engelhard, Jürgen Sturm, Daniel Cremers, and Wolfram Burgard. An evaluation of the RGB-D SLAM system. In *International Conference on Robotics and Automation (ICRA)*, volume 3, pages 1691–1696, 2012.
- [37] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3):611–625, 2017.

- [38] Jakob Engel, Thomas Schöps, and Daniel Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *European Conference on Computer Vision*, pages 834–849. Springer, 2014.
- [39] Jakob Engel, Jörg Stückler, and Daniel Cremers. Large-scale direct SLAM with stereo cameras. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1935–1942. IEEE, 2015.
- [40] Chris Engels, Henrik Stewénus, and David Nistér. Bundle adjustment rules. *Photogrammetric Computer Vision*, 2(2006), 2006.
- [41] Maryam Fatemi, Lennart Svensson, Lars Hammarstrand, and Malin Lundgren. Variational Bayesian EM for SLAM. In *2015 IEEE 6th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pages 501–504. IEEE, 2015.
- [42] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.
- [43] J Michael Fitzpatrick and Jay B West. The distribution of target registration error in rigid-body point-based registration. *IEEE Transactions on Medical Imaging*, 20(9):917–927, 2001.
- [44] Daniel Foreman-Mackey, David W Hogg, Dustin Lang, and Jonathan Goodman. emcee: the MCMC hammer. *Publications of the Astronomical Society of the Pacific*, 125(925):306, 2013.
- [45] Duncan Frost, Victor Prisacariu, and David Murray. Recovering stable scale in monocular SLAM using object-supplemented bundle adjustment. *IEEE Transactions on Robotics*, 34(3):736–747, 2018.
- [46] Jun Fu, Jing Liu, Yuhang Wang, Jin Zhou, Changyong Wang, and Hanqing Lu. Stacked deconvolutional network for semantic segmentation. *IEEE Transactions on Image Processing*, 2019.
- [47] Dorian Gálvez-López, Marta Salas, Juan D Tardós, and JMM Montiel. Real-time monocular object SLAM. *Robotics and Autonomous Systems*, 75:435–449, 2016.

- [48] Ravi Garg, Vijay Kumar BG, Gustavo Carneiro, and Ian Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *European Conference on Computer Vision*, pages 740–756. Springer, 2016.
- [49] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.
- [50] Sébastien Granger and Xavier Pennec. Multi-scale EM-ICP: A fast and robust approach for surface registration. In *European Conference on Computer Vision*, pages 418–432. Springer, 2002.
- [51] Giorgio Grisetti, Rainer Kummerle, Cyrill Stachniss, and Wolfram Burgard. A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.
- [52] Giorgio Grisetti, Gian Diego Tipaldi, Cyrill Stachniss, Wolfram Burgard, and Daniele Nardi. Fast and accurate SLAM with Rao–Blackwellized particle filters. *Robotics and Autonomous Systems*, 55(1):30–38, 2007.
- [53] Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2616–2625, 2017.
- [54] Jonas Haeling, Marc Necker, and Andreas Schilling. Towards immersive stereo vision from a mobile platform. In *Eleventh International Conference on Machine Vision (ICMV 2018)*, volume 11041, page 110410V. International Society for Optics and Photonics, 2019.
- [55] SangUk Han, Madhav Achar, SangHyun Lee, and Feniosky Peña-Mora. Empirical assessment of a RGB-D sensor on motion capture and action recognition for construction worker monitoring. *Visualization in Engineering*, 1(1):6, 2013.
- [56] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.
- [57] Carsten Hasberg, Stefan Hensel, and Christoph Stiller. Simultaneous localization and mapping for path-constrained motion. *IEEE Transactions on Intelligent Transportation Systems*, 13(2):541–552, 2011.

- [58] Simon Haykin. *Kalman filtering and neural networks*, volume 47. John Wiley & Sons, 2004.
- [59] Sebastian Hening, Corey A Ippolito, Kalmanje S Krishnakumar, Vahram Stepanyan, and Mircea Teodorescu. 3D LiDAR SLAM integration with GPS/INS for UAVs in urban GPS-degraded environments. In *AIAA Information Systems-AIAA Infotech@ Aerospace*, page 0448. 2017.
- [60] Matthew D Hoffman. Stochastic structured mean-field variational inference. *arXiv preprint arXiv:1404.4114*, 2014.
- [61] Guoquan P Huang, Anastasios I Mourikis, and Stergios I Roumeliotis. On the complexity and consistency of UKF-based slam. In *2009 IEEE International Conference on Robotics and Automation*, pages 4401–4408. IEEE, 2009.
- [62] Shoudong Huang and Gamini Dissanayake. Convergence analysis for extended Kalman filter based slam. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 412–417. IEEE, 2006.
- [63] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. Ccnet: Criss-cross attention for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 603–612, 2019.
- [64] Viorela Ila, Lukas Polok, Marek Solony, and Klemen Istenic. Fast incremental bundle adjustment with covariance recovery. In *2017 International Conference on 3D Vision (3DV)*, pages 175–184. IEEE, 2017.
- [65] Viorela Ila, Lukas Polok, Marek Solony, Pavel Smrz, and Pavel Zemcik. Fast covariance recovery in incremental nonlinear least square solvers. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4636–4643. IEEE, 2015.
- [66] Vadim Indelman. Bundle adjustment without iterative structure estimation and its application to navigation. In *Proceedings of the 2012 IEEE/ION Position, Location and Navigation Symposium*, pages 748–756. IEEE, 2012.
- [67] Vadim Indelman, Richard Roberts, Chris Beall, and Frank Dellaert. Incremental light bundle adjustment. Georgia Institute of Technology, 2012.

-
- [68] Lester Ingber. Very fast simulated re-annealing. *Mathematical and computer modelling*, 12(8):967–973, 1989.
- [69] Johan Ludwig William Valdemar Jensen et al. Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta mathematica*, 30:175–193, 1906.
- [70] Yonghoon Ji, Atsushi Yamashita, and Hajime Asama. RGB-D SLAM using vanishing point and door plate information in corridor environment. *Intelligent Service Robotics*, 8(2):105–114, 2015.
- [71] Bing Jian and Baba C Vemuri. Robust point set registration using Gaussian Mixture Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1633–1645, 2010.
- [72] Xiaoyue Jiang, Michael Hoy, Hang Yu, and Justin Dauwels. Linear-complexity stochastic variational bayes inference for slam. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2017.
- [73] Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, pages 315–323, 2013.
- [74] Michael Kaess, Viorela Ila, Richard Roberts, and Frank Dellaert. The Bayes tree: An algorithmic foundation for probabilistic robot mapping. In *Algorithmic Foundations of Robotics IX*, pages 157–173. Springer, 2010.
- [75] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. iSAM2: Incremental smoothing and mapping using the Bayes tree. *The International Journal of Robotics Research*, 31(2):216–235, 2012.
- [76] Jeong-Gwan Kang, Won-Seok Choi, Su-Yong An, and Se-Young Oh. Augmented EKF based SLAM method for improving the accuracy of the feature map. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3725–3731. IEEE, 2010.
- [77] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of*

- the IEEE International Conference on Computer Vision*, pages 2938–2946, 2015.
- [78] Christian Kerl, Jürgen Sturm, and Daniel Cremers. Dense visual SLAM for RGB-D cameras. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2100–2106. IEEE, 2013.
- [79] Yuehaw Khoo and Ankur Kapoor. Non-iterative rigid 2D/3D point-set registration using semidefinite programming. *IEEE Transactions on Image Processing*, 25(7):2956–2970, 2016.
- [80] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [81] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.
- [82] Georg Klein and David Murray. Parallel tracking and mapping for small AR workspaces. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–10. IEEE Computer Society, 2007.
- [83] Kurt Konolige, Motilal Agrawal, Robert C Bolles, Cregg Cowan, Martin Fischler, and Brian Gerkey. Outdoor mapping and navigation using stereo vision. In *Experimental Robotics*, pages 179–190. Springer, 2008.
- [84] Henrik Kretzschmar, Cyrill Stachniss, and Giorgio Grisetti. Efficient information-theoretic graph pruning for graph-based SLAM with laser range finders. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 865–871. IEEE, 2011.
- [85] Alp Kucukelbir, Rajesh Ranganath, Andrew Gelman, and David Blei. Automatic variational inference in Stan. In *Advances in Neural Information Processing Systems*, pages 568–576, 2015.
- [86] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g2o: A general framework for graph optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 3607–3613. IEEE, 2011.

- [87] Rainer Kümmerle, Bastian Steder, Christian Dornhege, Alexander Kleiner, Giorgio Grisetti, and Wolfram Burgard. Large scale graph-based SLAM using aerial images as prior information. *Autonomous Robots*, 30(1):25–39, 2011.
- [88] Wai Lau, An-ping Lee, Juhao Wu, Nuan-Ya Huang, and Ming-Chang Chou. Design of a Dogleg bunch compressor with tunable first-order longitudinal dispersion. 2018.
- [89] Pierre L’Ecuyer. Randomized quasi-Monte Carlo: An introduction for practitioners. In *International Conference on Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, pages 29–52. Springer, 2016.
- [90] Tae-jae Lee, Chul-hong Kim, and Dong-il Dan Cho. A monocular vision sensor-based efficient SLAM method for indoor service robots. *IEEE Transactions on Industrial Electronics*, 66(1):318–328, 2018.
- [91] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334, 2015.
- [92] Liang Li, Ming Yang, Chunxiang Wang, and Bing Wang. Gaussian Mixture Model-signature quadratic form distance based point set registration. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 998–1003. IEEE, 2017.
- [93] Liang Li, Ming Yang, Chunxiang Wang, and Bing Wang. Rigid point set registration based on cubature Kalman filter and its application in intelligent vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 19(6):1754–1765, 2017.
- [94] Liang Li, Ming Yang, Chunxiang Wang, and Bing Wang. Cubature split covariance intersection filter-based point set registration. *IEEE Transactions on Image Processing*, 27(8):3942–3953, 2018.
- [95] Liang Li, Ming Yang, Chunxiang Wang, and Bing Wang. Robust point set registration using signature quadratic form distance. *IEEE Transactions on Cybernetics*, 2018.
- [96] Shunkai Li, Fei Xue, Xin Wang, Zike Yan, and Hongbin Zha. Sequential adversarial learning for self-supervised deep visual odometry. In *Proceedings*

- of the *IEEE International Conference on Computer Vision*, pages 2851–2860, 2019.
- [97] Wenfa Li, Dewei Li, Haibin Shao, and Yunwen Xu. An RGBD-SLAM with Bi-directional PnP method and fuzzy frame detection module. In *2019 12th Asian Control Conference (ASCC)*, pages 1530–1535. IEEE, 2019.
- [98] Jongwoo Lim, Jan-Michael Frahm, and Marc Pollefeys. Online environment mapping. In *2011 IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3489–3496. IEEE, 2011.
- [99] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125, 2017.
- [100] Richard Linares and John Crassidis. Sigma point transformation for Gaussian Mixture Distributions. In *AIAA Guidance, Navigation, and Control Conference*, page 4936, 2012.
- [101] Qiang Liu and Alexander Ihler. Variational algorithms for marginal map. *The Journal of Machine Learning Research*, 14(1):3165–3200, 2013.
- [102] Gucan Long, Laurent Kneip, Jose M Alvarez, Hongdong Li, Xiaohu Zhang, and Qifeng Yu. Learning image matching by simply watching video. In *European Conference on Computer Vision*, pages 434–450. Springer, 2016.
- [103] David G Lowe. Method and apparatus for identifying scale invariant features in an image and use of same for locating an object in an image, March 23 2004. US Patent 6,711,293.
- [104] Feng Lu and Evangelos Milios. Globally consistent range scan alignment for environment mapping. *Autonomous Robots*, 4(4):333–349, 1997.
- [105] Malin Lundgren, Lennart Svensson, and Lars Hammarstrand. Variational Bayesian expectation maximization for radar map estimation. *IEEE Transactions on Signal Processing*, 64(6):1391–1404, 2015.
- [106] Jiayi Ma, Ji Zhao, Junjun Jiang, and Huabing Zhou. Non-rigid point set registration with robust transformation estimation under manifold regularization. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

- [107] Jiayi Ma, Ji Zhao, and Alan L Yuille. Non-rigid point set registration by preserving global and local structures. *IEEE Transactions on Image Processing*, 25(1):53–64, 2015.
- [108] Reza Mahjourian, Martin Wicke, and Anelia Angelova. Unsupervised learning of depth and ego-motion from monocular video using 3D geometric constraints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5667–5675, 2018.
- [109] Ruben Mascaró, Lucas Teixeira, Timo Hinzmann, Roland Siegwart, and Margarita Chli. GOMSF: Graph-Optimization based Multi-Sensor Fusion for robust UAV pose estimation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1421–1428. IEEE, 2018.
- [110] John McCormac, Ankur Handa, Andrew Davison, and Stefan Leutenegger. Semanticfusion: Dense 3D semantic mapping with convolutional neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4628–4635. IEEE, 2017.
- [111] Volodymyr Melnykov, Ranjan Maitra, et al. Finite mixture models and model-based clustering. *Statistics Surveys*, 4:80–116, 2010.
- [112] Andrew Miller, Nick Foti, Alexander D’Amour, and Ryan P Adams. Reducing reparameterization gradient variance. In *Advances in Neural Information Processing Systems*, pages 3708–3718, 2017.
- [113] Atanas Mirchev, Baris Kayalibay, Patrick van der Smagt, and Justin Bayer. Approximate Bayesian inference in spatial environments. *arXiv preprint arXiv:1805.07206*, 2018.
- [114] Hongwei Mo, Kai Wang, Haoran Wang, and Weihao Ding. Research on loop closing for SLAM based on RGB-D images. In *Proceedings of 2018 Chinese Intelligent Systems Conference*, pages 739–748. Springer, 2019.
- [115] Mehdi Hedjazi Moghari and Purang Abolmaesumi. Point-based rigid-body registration using an unscented Kalman filter. *IEEE Transactions on Medical Imaging*, 26(12):1708–1728, 2007.
- [116] Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit, et al. FastSLAM: A factored solution to the simultaneous localization and mapping problem. *Aaai/iaai*, 593598, 2002.

- [117] Michael Montemerlo, Sebastian Thrun, Daphne Koller, Ben Wegbreit, et al. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *International Joint Conference on Artificial Intelligence*, pages 1151–1156, 2003.
- [118] Todd K Moon. The expectation-maximization algorithm. *IEEE Signal Processing Magazine*, 13(6):47–60, 1996.
- [119] Etienne Mouragnon, Maxime Lhuillier, Michel Dhome, Fabien Dekeyser, and Patrick Sayd. Real time localization and 3D reconstruction. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 363–370. IEEE, 2006.
- [120] John Mullane, Ba-Ngu Vo, Martin D Adams, and Ba-Tuong Vo. A Random-Finite-Set approach to Bayesian SLAM. *IEEE Transactions on Robotics*, 27(2):268–282, 2011.
- [121] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015.
- [122] Raul Mur-Artal and Juan D Tardós. ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.
- [123] Raúl Mur-Artal and Juan D Tardós. Visual-inertial monocular SLAM with map reuse. *IEEE Robotics and Automation Letters*, 2(2):796–803, 2017.
- [124] Don Murray and James J Little. Using real-time stereo vision for mobile robot navigation. *Autonomous Robots*, 8(2):161–171, 2000.
- [125] Andriy Myronenko and Xubo Song. Point set registration: Coherent point drift. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(12):2262–2275, 2010.
- [126] Andriy Myronenko, Xubo Song, and Miguel A Carreira-Perpinán. Non-rigid point set registration: Coherent point drift. In *Advances in Neural Information Processing Systems*, pages 1009–1016, 2007.
- [127] John C Nash. *Compact numerical methods for computers: linear algebra and function minimisation*. Routledge, 2018.

- [128] Jogendra Nath Kundu, Rahul MV, Aditya Ganeshan, and R Venkatesh Babu. Object pose estimation from monocular image using multi-view keypoint correspondence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018.
- [129] Christopher Nemeth and Paul Fearnhead. Stochastic gradient markov chain monte carlo. *arXiv preprint arXiv:1907.06986*, 2019.
- [130] Thanh Minh Nguyen and QM Jonathan Wu. Multiple kernel point set registration. *IEEE Transactions on Medical Imaging*, 35(6):1381–1394, 2015.
- [131] Juan Nieto, Jose Guivant, and Eduardo Nebot. DenseSLAM: Simultaneous localization and dense mapping. *The International Journal of Robotics Research*, 25(8):711–744, 2006.
- [132] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, volume 1, pages I–I. Ieee, 2004.
- [133] Lina M Paz, Juan D Tardós, and José Neira. Divide and conquer: EKF SLAM in $O(n)$. *IEEE Transactions on Robotics*, 24(5):1107–1120, 2008.
- [134] Marek Pierzchała, Philippe Giguère, and Rasmus Astrup. Mapping forests using an unmanned ground vehicle with 3D LiDAR and graph-SLAM. *Computers and Electronics in Agriculture*, 145:217–225, 2018.
- [135] Taihú Pire, Thomas Fischer, Javier Civera, Pablo De Cristóforis, and Julio Jacobo Berles. Stereo parallel tracking and mapping for robot localization. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1373–1378. IEEE, 2015.
- [136] François Pomerleau, Francis Colas, Roland Siegwart, and Stéphane Magnenat. Comparing ICP variants on real-world data sets. *Autonomous Robots*, 34(3):133–148, 2013.
- [137] WH Press, SA Teukolsky, WT Vetterling, and BP Flannery. Section 14.7. 2. Kullback-Leibler distance. *Numerical Recipes: The Art of Scientific Computing*, 2007.

- [138] Chuang Qian, Hui Liu, Jian Tang, Yuwei Chen, Harri Kaartinen, Antero Kukko, Lingli Zhu, Xinlian Liang, Liang Chen, and Juha Hyypä. An integrated GNSS/INS/LiDAR-SLAM positioning method for highly accurate forest stem mapping. *Remote Sensing*, 9(1):3, 2017.
- [139] Tong Qin, Peiliang Li, and Shaojie Shen. ViNS-Mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.
- [140] Tong Qin and Shaojie Shen. Online temporal calibration for monocular visual-inertial systems. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3662–3669. IEEE, 2018.
- [141] Rajesh Ranganath, Sean Gerrish, and David Blei. Black box variational inference. In *Artificial Intelligence and Statistics*, pages 814–822, 2014.
- [142] Anand Rangarajan, Haili Chui, and James S Duncan. Rigid point feature registration using mutual information. *Medical Image Analysis*, 3(4):425–440, 1999.
- [143] David Ribas, Pere Ridao, Juan Domingo Tardós, and José Neira. Underwater SLAM in a marina environment. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1455–1460. IEEE, 2007.
- [144] Herbert Robbins and Sutton Monroe. A stochastic approximation method. *The Annals of Mathematical Statistics*, pages 400–407, 1951.
- [145] Christian Robert and George Casella. *Monte Carlo statistical methods*. Springer Science & Business Media, 2013.
- [146] Antonio L Rodriguez, Pedro E López-de Teruel, and Alberto Ruiz. GEA optimization for live structureless motion estimation. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 715–718. IEEE, 2011.
- [147] Antonio L Rodríguez, Pedro E López-de Teruel, and Alberto Ruiz. Reduced epipolar cost for accelerated incremental SFM. In *2011 IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3097–3104. IEEE, 2011.

- [148] Geoffrey Roeder, Yuhuai Wu, and David K Duvenaud. Sticking the landing: Simple, lower-variance gradient estimators for variational inference. In *Advances in Neural Information Processing Systems*, pages 6925–6934, 2017.
- [149] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary R Bradski. ORB: An efficient alternative to SIFT or SURF. In *International Conference on Computer Vision (ICCV)*, volume 11, page 2. Citeseer, 2011.
- [150] Francisco R Ruiz, Michalis Titsias RC AUEB, and David Blei. The generalized reparameterization gradient. In *Advances in Neural Information Processing Systems*, pages 460–468, 2016.
- [151] David Saad. Online algorithms and stochastic approximations. *Online Learning*, 5, 1998.
- [152] Renato F Salas-Moreno, Richard A Newcombe, Hauke Strasdat, Paul HJ Kelly, and Andrew J Davison. SLAM++: Simultaneous localisation and mapping at the level of objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1352–1359, 2013.
- [153] Romeil Sandhu, Samuel Dambreville, and Allen Tannenbaum. Point set registration via particle filtering and stochastic dynamics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8):1459–1473, 2009.
- [154] Joao Machado Santos, David Portugal, and Rui P Rocha. An evaluation of 2D SLAM techniques available in robot operating system. In *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 1–6. IEEE, 2013.
- [155] Thomas Schops, Torsten Sattler, and Marc Pollefeys. BAD SLAM: Bundle Adjusted Direct RGB-D SLAM. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 134–144, 2019.
- [156] Sebastian Schuon, Christian Theobalt, James Davis, and Sebastian Thrun. High-quality scanning using time-of-flight depth superresolution. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–7. IEEE, 2008.
- [157] Gang Shao, Lei Wan, and Xiao Di Shen. Hierarchical map building based UKF-SLAM approach for AUV. In *Applied Mechanics and Materials*, volume 437, pages 793–797. Trans Tech Publ, 2013.

- [158] Hagit Shatkay and Leslie Pack Kaelbling. Learning topological maps with weak local odometric information. In *International Joint Conference on Artificial Intelligence*, pages 920–929, 1997.
- [159] Mark Sheehan, Alastair Harrison, and Paul Newman. Continuous vehicle localisation using sparse 3D sensing, kernelised Rényi distance and fast Gauss transforms. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 398–405. IEEE, 2013.
- [160] Gabe Sibley, Larry Matthies, and Gaurav Sukhatme. Sliding window filter with application to planetary landing. *Journal of Field Robotics*, 27(5):587–608, 2010.
- [161] Sandip Sinharay. Assessing convergence of the Markov chain Monte Carlo algorithms: A review. *ETS Research Report Series*, 2003(1):i–52, 2003.
- [162] James S Smith, Bo Wu, and Bogdan M Wilamowski. Neural network training with Levenberg–Marquardt and adaptable weight compression. *IEEE Transactions on Neural Networks and Learning Systems*, 30(2):580–587, 2018.
- [163] James C Spall. Stochastic optimization. In *Handbook of computational statistics*, pages 173–201. Springer, 2012.
- [164] Richard Steffen, Jan-Michael Frahm, and Wolfgang Förstner. Relative bundle adjustment based on trifocal constraints. In *European Conference on Computer Vision*, pages 282–295. Springer, 2010.
- [165] Martha L Stocking. Scale drift in on-line calibration. *ETS Research Report Series*, 1988(1):i–122, 1988.
- [166] Hauke Strasdat, Andrew J Davison, JM Martínez Montiel, and Kurt Konolige. Double window optimisation for constant time visual slam. In *2011 International Conference on Computer Vision*, pages 2352–2359. IEEE, 2011.
- [167] Hauke Strasdat, J Montiel, and Andrew J Davison. Scale drift-aware large scale monocular SLAM. *Robotics: Science and Systems VI*, 2(3):7, 2010.
- [168] Hauke Strasdat, José MM Montiel, and Andrew J Davison. Visual SLAM: why filter? *Image and Vision Computing*, 30(2):65–77, 2012.

- [169] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580. IEEE, 2012.
- [170] Wenbing Tao and Kun Sun. Asymmetrical Gauss mixture models for point sets matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1598–1605, 2014.
- [171] Wenbing Tao and Kun Sun. Robust point sets matching by fusing feature and spatial information using nonuniform gaussian mixture models. *IEEE Transactions on Image Processing*, 24(11):3754–3767, 2015.
- [172] Keisuke Tateno, Federico Tombari, Iro Laina, and Nassir Navab. CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6243–6252, 2017.
- [173] S Teller, E Olson, and J Leonard. Fast iterative optimization of pose graphs with poor initial estimates. In *Proc. of IEEE International Conference on Robotics and Automation*, pages 2262–2269, 2006.
- [174] Peter Teunissen. Nonlinear least squares. 1990.
- [175] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. A probabilistic approach to concurrent mapping and localization for mobile robots. *Autonomous Robots*, 5(3-4):253–271, 1998.
- [176] Sebastian Thrun, Yufeng Liu, Daphne Koller, Andrew Y Ng, Zoubin Ghahramani, and Hugh Durrant-Whyte. Simultaneous localization and mapping with sparse extended information filters. *The International Journal of Robotics Research*, 23(7-8):693–716, 2004.
- [177] Michalis Titsias and Miguel Lázaro-Gredilla. Doubly stochastic variational Bayes for non-conjugate inference. In *International Conference on Machine Learning*, pages 1971–1979, 2014.
- [178] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment—a modern synthesis. In *International Workshop on Vision Algorithms*, pages 298–372. Springer, 1999.

- [179] Yanghai Tsin and Takeo Kanade. A correlation-based approach to robust point set registration. In *European Conference on Computer Vision*, pages 558–569. Springer, 2004.
- [180] Nicholas J Tustison, Suyash P Awate, Gang Song, Tessa S Cook, and James C Gee. Point set registration using Havrda–Charvat–Tsallis entropy measures. *IEEE Transactions on Medical Imaging*, 30(2):451–460, 2010.
- [181] Alexander Vakhitov and Victor Lempitsky. Learnable line segment descriptor for visual SLAM. *IEEE Access*, 7:39923–39934, 2019.
- [182] Nam Vo, Nathan Jacobs, and James Hays. Revisiting IM2GPS in the deep learning era. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2621–2630, 2017.
- [183] Matthew R Walter, Ryan M Eustice, and John J Leonard. Exactly sparse extended information filters for feature-based SLAM. *The International Journal of Robotics Research*, 26(4):335–359, 2007.
- [184] Eric A Wan and Rudolph Van Der Merwe. The unscented Kalman filter for nonlinear estimation. In *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*, pages 153–158. IEEE, 2000.
- [185] Chen Wang, Junsong Yuan, and Lihua Xie. Non-iterative SLAM. In *2017 18th International Conference on Advanced Robotics (ICAR)*, pages 83–90. IEEE, 2017.
- [186] Gang Wang, Zhicheng Wang, Yufei Chen, Weidong Zhao, and Xianhui Liu. Fuzzy correspondences and kernel density estimation for contaminated point set registration. In *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pages 1936–1941. IEEE, 2015.
- [187] Gang Wang, Zhicheng Wang, Yufei Chen, Qiangqiang Zhou, and Weidong Zhao. Context-aware gaussian fields for non-rigid point set registration. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5811–5819, 2016.
- [188] Hongjian Wang, Guixia Fu, Juan Li, Zheping Yan, and Xinqian Bian. An adaptive UKF based SLAM method for unmanned underwater vehicle. *Mathematical Problems in Engineering*, 2013, 2013.

- [189] Hongwei Wang, Hang Yu, Michael Hoy, Justin Dauwels, and Heping Wang. Variational Bayesian dynamic compressive sensing. In *2016 IEEE International Symposium on Information Theory (ISIT)*, pages 1421–1425. IEEE, 2016.
- [190] Xiang Wang and Hong Zhang. A UPF-UKF framework for SLAM. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, pages 1664–1669. IEEE, 2007.
- [191] Jian Wu, Liwei Ma, and Xiaolin Hu. Delving deeper into convolutional neural networks for camera relocalization. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5644–5651. IEEE, 2017.
- [192] Mou Wu, Naixue Xiong, and Liansheng Tan. Adaptive range-based target localization using diffusion Gauss-Newton method in industrial environments. *IEEE Transactions on Industrial Informatics*, 2019.
- [193] Guanglin Xu, Shaoyi Du, and Jianru Xue. Precise 2D point set registration using iterative closest algorithm and correntropy. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 4627–4631. IEEE, 2016.
- [194] Tingxing Yan and Yong Zhang. Mobile robots location and mapping based on corner features. In *2018 5th International Conference on Information Science and Control Engineering (ICISCE)*, pages 833–838. IEEE, 2018.
- [195] Jiaolong Yang, Hongdong Li, Dylan Campbell, and Yunde Jia. Go-ICP: A globally optimal solution to 3D ICP point-set registration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11):2241–2254, 2015.
- [196] Nan Yang, Rui Wang, Jorg Stuckler, and Daniel Cremers. Deep virtual stereo odometry: Leveraging deep depth prediction for monocular direct sparse odometry. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 817–833, 2018.
- [197] Cang Ye and Yimin Zhao. An EKF-SLAM method with filter consistency test for mobile robots using a 3D camera. In *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 188–193. IEEE, 2015.
- [198] Chuho Yi, Seungdo Jeong, and Jungwon Cho. Map representation for robots. *Smart Computing Review*, 2(1):18–27, 2012.

-
- [199] Fei Yu, Qian Sun, Chongyang Lv, Yueyang Ben, and Yanwei Fu. A SLAM algorithm based on adaptive cubature Kalman filter. *Mathematical Problems in Engineering*, 2014, 2014.
 - [200] Hang Yu and Justin Dauwels. Modeling spatio-temporal extreme events using graphical models. *IEEE Transactions on Signal Processing*, 64(5):1101–1116, 2015.
 - [201] Hang Yu and Justin Dauwels. Variational Bayes learning of graphical models with hidden variables. In *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE, 2015.
 - [202] Hang Yu, Wayne Isaac T Uy, and Justin Dauwels. Modeling spatial extremes via ensemble-of-trees of pairwise copulas. *IEEE Transactions on Signal Processing*, 65(3):571–586, 2016.
 - [203] Anderson Y Zhang and Harrison H Zhou. Theoretical and computational guarantees of mean field variational inference for community detection. *arXiv preprint arXiv:1710.11268*, 2017.
 - [204] Jianzhong Zhang and Detong Zhu. A trust region typed dogleg method for nonlinear optimization. *Optimization*, 21(4):543–557, 1990.
 - [205] Liangpei Zhang, Lefei Zhang, and Bo Du. Deep learning for remote sensing data: A technical tutorial on the state of the art. *IEEE Geoscience and Remote Sensing Magazine*, 4(2):22–40, 2016.
 - [206] Wei Zhang, Kazuyoshi Itoh, Jun Tanida, and Yoshiki Ichioka. Parallel distributed processing model with local space-invariant interconnections and its optical architecture. *Applied Optics*, 29(32):4790–4797, 1990.
 - [207] Wenling Zhang, Mingqing Zhu, and Zonghai Chen. An adaptive SLAM algorithm based on strong tracking UKF. *Robot*, 32(2):190–195, 2010.
 - [208] Binqi Zheng, Pengcheng Fu, Baoqing Li, and Xiaobing Yuan. A robust adaptive Unscented Kalman filter for nonlinear estimation with uncertain noise covariance. *Sensors*, 18(3):808, 2018.
 - [209] Binqi Zheng, Pengcheng Fu, Baoqing Li, and Xiaobing Yuan. A robust adaptive unscented Kalman filter for nonlinear estimation with uncertain noise covariance. *Sensors*, 18(3):808, 2018.

-
- [210] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1851–1858, 2017.
 - [211] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3357–3364. IEEE, 2017.
 - [212] Michael Zollhöfer. Commodity RGB-D sensors: Data acquisition. *arXiv preprint arXiv:1902.06835*, 2019.