An Expectation-Maximization Approach to Unsupervised Parameter Learning for Trajectory Estimation

by

Juny David Yoon

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Aerospace Science and Engineering
University of Toronto

Abstract

An Expectation-Maximization Approach to Unsupervised Parameter Learning for Trajectory Estimation

Juny David Yoon
Doctor of Philosophy
Graduate Department of Aerospace Science and Engineering
University of Toronto
2025

Estimating the trajectory of a robot over time is a key component to enabling autonomy in the real world. While there are remaining challenges in difficult edge and corner cases, research and application of state estimation has reached an impressive level of maturity. Large-scale trajectories can be estimated efficiently and accurately with observation data from rich sensor modalities such as cameras and lidars. However, for each different application setting, platform, and mounted sensor, manual effort from an expert state estimation engineer is necessary for successful robot deployment. From this perspective, the research goal of this thesis is to formulate and apply a parameter learning framework that will help automate this deployment process. We seek to train the model parameters in our estimators from data and reduce the burden of manual tuning. To meet this end, we formulate parameter learning as an optimization of the observed data likelihood, which we choose to optimize using the well-known Expectation-Maximization (EM) algorithm. The first major chapter of this thesis develops the necessary tools that we may require for EM, which resulted in a Gaussian variational inference framework that is tractable for large-scale estimation problems. We dedicate the remaining three major chapters to various applications of EM. Two chapters focus on learning measurement noise models, where one of these chapters also considers measurement bias. The last chapter focuses on improving the learning capacity of our models by applying deep learning. We demonstrate experimental results on several real-world datasets where we use EM to train our models without supervision from the groundtruth trajectory.

Acknowledgements

This dissertation would not have been possible without the generous support and guidance of my supervisor, lab mates, and industry collaborators. To my family, thank you for your unconditional love and encouragement to pursue higher levels of education.

Contents

List of Tables

List of Figures

Acronyms

CNN	Convolutional Neural Network
DNN	Deep Neural Network
DOF	Degrees of Freedom
ESGVI	Exactly Sparse Gaussian Variational Inference
EM	Expectation-Maximization
FMCW	Frequency-Modulated Continuous Wave
GP	Gaussian process
GVI	Gaussian Variational Inference
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
MAP	Maximum A Posteriori
PDF	Probability Distribution Function
RANSAC	Random Sample Consensus
ReLU	Rectified Linear Unit
RMSE	Root Mean Squared Error
SLAM	Simultaneous Localization and Mapping
UTIAS	University of Toronto Institute for Aerospace Studies

Notation

a	Symbols in this font are real scalars.
$\mathbf{a}$	Symbols in this font are real column vectors.
$\mathbf{A}$	Symbols in this font are real matrices.
$\hat{\mathbf{a}}$	A unit vector.
$\mathbf{1}$	The identity matrix.
$\mathbf{0}$	The zero matrix.
$\underrightarrow{\mathcal{F}}_a$	A vectrix representing a reference frame in three dimensions.
$\mathbf{p}_a^{cb}$	A vector from point b to point c (denoted by the superscript) and expressed in $\underrightarrow{\mathcal{F}}_a$.
$SO(3)$	The special orthogonal group.
$SE(3)$	The special Euclidean group.
$\mathbb{E}[\cdot]$	The expectation operator.

Chapter 1

Introduction

The accelerating research in autonomous navigation has enabled the application of robots in the real world. Robots operating in our daily lives was once thought to be a novelty, but is now becoming more commonplace. To list a few examples, it is now not out of the ordinary to see robot vacuums in households today. Commercial drones can be purchased with active tracking functionality to autonomously record a subject during outdoor recreation. Waymo, the self-driving subsidiary of Google, offers driver-less taxi services in select locations in the United States (Schwall et al., 2020). These are only a handful of examples of robot application in the real world, with many that already exist, and many more to come.

A key component of autonomous navigation that enables robotics applications is *state estimation*. The *state* of a robot can be defined as its position, orientation, and the change in these quantities over time (Barfoot, 2024). While other components, such as control and path planning, are also essential and directly contribute to the robot's actions in its setting, the estimate of a robot's trajectory is a critical precursor step.

While there are still remaining challenges, state estimation has achieved an impressive level of maturity over the past couple decades. Robust, accurate estimators can be implemented using a combination of different sensor modalities, including those that produce rich data such as cameras, lidars, and radars. In situations where one sensor modality may fail, another sensor can compensate. For example, visual localization using cameras is extremely challenging under severe changes in ambient lighting (Paton et al., 2016), for which we may be able to compensate using a lidar sensor that is robust to ambient lighting in comparison. The same can be said going the other way; lidar sensors can fail to localize in environments with low geometric structure (e.g., a tunnel or barren landscape), for which we may be able to use a camera (given there are enough visual cues), or an inertial measurement unit (IMU) to help dead reckon. The real world is dynamic and unpredictable, which may seem problematic as many of our sensor models are designed assuming that the world is static. However, classic tools in robust estimation, such as Random Sample And Consensus (RANSAC) (Fischler and Bolles, 1981) or M-estimation (Zhang, 1997) offer a surprising level of robustness that facilitates the application of state estimation in many scenarios that may appear difficult at first glance.

From this perspective on the maturity of state estimation, the work presented in this thesis takes a different approach for contributing to state estimation research. We are less concerned about pushing the state of the art on estimation accuracy, e.g., setting the best performance on a benchmark for odometry or localization. Instead, we are interested in how we can improve the automation process of deploying robust, accurate estimators in different application settings. Within the implementation of an estimator, there is a large dependence on the platform, sensors, and application setting that often involves manual design and tuning by an expert engineer. Rather than having an expert engineer meticulously design and tune estimators for each different application, we wish to automate the process of tuning the models from data. For this, we turn to machine learning, but in a way that we maintain many of the classic tools in estimation with which we are familiar.

1.1 Thesis Overview and Novel Contributions

In Chapter 2, we begin with a review on batch state estimation. In particular, we focus on the method known as Maximum A Posteriori (MAP) as it is the most commonly used tool for nonlinear batch inference of a Gaussian approximation of the posterior Probability Density Function (PDF) (Barfoot, 2024). This review sets the stage for the following chapter, in which we present an alternative formulation for batch state estimation that improves upon MAP. There are no novel contributions in this background chapter.

Chapter 3 focuses on the (second author) contributions made to an estimation framework, Exactly Sparse Gaussian Variational Inference (ESGVI) (Barfoot et al., 2020). ESGVI is a nonlinear batch state estimation framework that starts from a variational objective (Bishop, 2006), and provides a family of scalable estimators by exploiting the factorization of the joint likelihood between the observed measurements (data) and state. We demonstrate that our choice of optimizing a variational objective is better than MAP as it optimizes a closer Gaussian fit to the true Bayesian posterior. This is because MAP is simply an optimization of a point estimate of the posterior, which we show is an approximation of our ESGVI method when we optimize the variational objective for only the mean (and not the covariance) of the approximate posterior.

The novel contributions of Chapter 3 are:

1. A computationally tractable approach to Gaussian Variational Inference (GVI) for large-scale estimation problems via exploitation of the sparsity formed from the factorization of the joint likelihood.

2. An implementation of sparse GVI that uses Gaussian cubature to avoid the need for computing analytical derivatives.

3. A conservative, cheaper (compute) approximation of Gaussian Variational Inference (GVI) that is applicable under mild nonlinearities and/or when the posterior is concentrated.

4. Various experiments in both simulation and on real data demonstrating an improvement in performance over MAP.

Critically, our work on ESGVI outlines the necessary tools we require for parameter learning using the well-known Expectation-Maximization (EM) algorithm. This is because the GVI loss is exactly the Evidence Lower Bound (ELBO) that we optimize in EM (Barfoot et al., 2020). We choose EM as our method of parameter learning for its potential to harmoniously combine probabilistic state estimation with machine learning under a single data likelihood objective. We dedicate the remaining chapters of this thesis to applying our ESGVI and EM parameter learning framework to various estimation problems.

Starting with Chapter 4, we learn a measurement covariance model that can vary over time in a trajectory. We apply our ESGVI and EM parameter learning framework by placing an Inverse-Wishart (IW) prior on the measurement covariance and estimating it (the covariance) as part of the state. Our approach results in an adapting measurement covariance that is robust to measurement outliers. This work was done in part as a shared (second author) contribution (Wong et al., 2020b).

The novel contributions of Chapter 4 are:

1. A methodology for estimating measurement covariance by using an IW prior in a EM framework.

2. Experimental results on a lidar localization dataset that demonstrates learning a varying measurement covariance without the groundtruth trajectory. We also show that the resulting covariance model is robust to measurement outliers during both training and testing.

Chapter 5 focuses on learning measurement bias and noise as feature-dependent regression models. Due to nonidealities in the real world, our sensors can produce biased measurements that we may have to calibrate ourselves. Similar to our motivation in Chapter 4, the uncertainty of the measurements can differ from the manufacturer specification and may vary depending on the operation environment. Feature-dependent regression models are a viable solution for applications where these quantities are predictable from a training dataset. While the idea of using feature-dependent models is not novel, we present our EM training scheme as a way to train these models without the groundtruth trajectory. We demonstrate our approach on odometry experiments with a Frequency-Modulated Continuous Wave (FMCW) lidar sensor. The estimators we use in this chapter appeared in two associated publications, one of which is a second-author contribution (Wu et al., 2023; Yoon et al., 2023). In this thesis, we improve upon the work from these publications by training for measurement bias and noise without groundtruth supervision.

The novel contributions of Chapter 5 are:

1. A methodology for training feature-dependent regression models for measurement bias and covariance using ESGVI and EM.
2. A lightweight correspondence-free odometry method using Doppler measurements from a FMCW lidar and gyroscope measurements from an IMU.
3. Experiments using the proposed method for learning Doppler measurement bias and variance, demonstrating the ability to train the regression models without supervision from the groundtruth trajectory.

Chapter 6 focuses on extending our application of parameter learning with Deep Neural Networks (DNNs). We first look into how we can model a measurement model in its entirety using a DNN. We demonstrate application of the derivative-free approach for ESGVI (see Section 3.2.5), which allows us to efficiently train and use a neural network measurement model without computing derivatives of the model with respect to our vehicle state. Instead of derivatives, we use multiple forward passes with sigmapoints, which complements the parallel work-flow of DNNs well. We then present an alternative way of using a DNN to tractably handle rich sensor data. Instead of modelling the measurement model directly, which can be costly in compute due to the density of measurements, we can train a Convolutional Neural Network (CNN) front-end that processes the rich data and outputs sparse features that can be used with a probabilistic estimation back-end (e.g., ESGVI). We demonstrate how our approach for learning deep features is applicable to both lidar (Yoon et al., 2021) and radar (Burnett et al., 2021) odometry.

The novel contributions of Chapter 6 are:

1. Learning a DNN measurement model using EM and ESGVI without having to compute the derivative (Jacobian) of the model with respect to the state.
2. Experiments on a real-robot dataset for localization and learning a range-bearing measurement model without the groundtruth trajectory.
3. Learning a DNN using EM and ESGVI that is specifically tailored to rich sensor data via a CNN architecture that outputs sparse keypoints, descriptors, and uncertainty.
4. Experiments on lidar odometry, again demonstrating that our parameter learning framework can train the network parameters without the groundtruth trajectory.

Finally, a summary of the contributions of this thesis and a discussion of future work are presented in Chapter 7.

1.2 List of Associated Publications

For Chapter 3:

- Barfoot, T. D., Forbes, J. R., and Yoon, D. J. (2020). Exactly sparse gaussian variational inference with application to derivative-free batch nonlinear state estimation. *International Journal of Robotics Research (IJRR) (second author contribution)*

For Chapter 4:

- Wong, J. N., Yoon, D. J., Schoellig, A. P., and Barfoot, T. D. (2020b). Variational inference with parameter learning applied to vehicle trajectory estimation. *IEEE Robotics and Automation Letters (RAL)*, 5(4):5291–5298 *(second author contribution)*

For Chapter 5:

- Wu, Y., Yoon, D. J., Burnett, K., Kammel, S., Chen, Y., Vhavle, H., and Barfoot, T. D. (2023). Picking up speed: Continuous-time lidar-only odometry using doppler velocity measurements. *IEEE Robotics and Automation Letters (RAL)*, 8(1):264–271 *(second author contribution)*

- Yoon, D. J., Burnett, K., Laconte, J., Chen, Y., Vhavle, H., Kammel, S., Reuther, J., and Barfoot, T. D. (2023). Need for speed: Fast correspondence-free lidar-inertial odometry using doppler velocity. In *International Conference on Intelligent Robots and Systems (IROS)*

For Chapter 6:

- Yoon, D. J., Zhang, H., Gridseth, M., Thomas, H., and Barfoot, T. D. (2021). Unsupervised learning of lidar features for use in a probabilistic trajectory estimator. *IEEE Robotics and Automation Letters (RAL)*, 6(2):2130–2138

- Burnett, K., Yoon, D. J., Schoellig, A. P., and Barfoot, T. D. (2021). Radar odometry combining probabilistic estimation and unsupervised feature learning. In *Robotics: Science and Systems (RSS) (equal contribution between Burnett, K. and Yoon, D.)*

Chapter 2

Background on Batch State Estimation

This chapter presents a background primer on batch state estimation, covering topics relevant to the latter chapters of this thesis. Much of the content in this section are sourced from and can be found in more detail in the textbook by Barfoot (2024). There are no novel contributions in this chapter.

2.1 Gaussian Probability

In this thesis, we choose to model uncertainty using a Gaussian PDF. The Gaussian PDF of a random variable, $\mathbf{x} \in \mathbb{R}^N$, is

$$p(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^N |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})\right), \tag{2.1}$$

where $|\cdot|$ is the matrix determinant, $\boldsymbol{\mu} \in \mathbb{R}^N$ is the mean, and $\boldsymbol{\Sigma} \in \mathbb{R}^{N \times N}$ is the symmetric, positive-definite covariance matrix.

Gaussian distributions have a convenient marginalization property, where the marginal distribution of a joint Gaussian is simply another Gaussian PDF with the mean and covariance corresponding to the blocks of that marginal. As an example, say we have a joint distribution

$$p(\mathbf{x}_1, \mathbf{x}_2) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix}\right). \tag{2.2}$$

The marginal distributions are

$$p(\mathbf{x}_1) = \int p(\mathbf{x}_1, \mathbf{x}_2) d\mathbf{x}_2 = \mathcal{N}(\boldsymbol{\mu}_1, \mathbf{\Sigma}_{11}), \tag{2.3a}$$

$$p(\mathbf{x}_2) = \int p(\mathbf{x}_1, \mathbf{x}_2) d\mathbf{x}_1 = \mathcal{N}(\boldsymbol{\mu}_2, \mathbf{\Sigma}_{22}). \tag{2.3b}$$

Often in state estimation we will not immediately have access to the mean and covariance, but have the quantities in *information form*. For a Gaussian $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{\Sigma})$, the information form consists of the *information vector*, $\mathbf{\Sigma}^{-1}\boldsymbol{\mu}$, and the *information matrix*, $\mathbf{\Sigma}^{-1}$. A benefit of working with the information form is that the information matrix (inverse covariance) will have sparsity that the covariance matrix does not have.

We can still conveniently marginalize our distribution when it is in information form. Given the following joint distribution,

$$\mathbf{\Sigma}^{-1}\boldsymbol{\mu} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}, \quad \mathbf{\Sigma}^{-1} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix}, \tag{2.4}$$

the information marginals are

$$\mathbf{\Sigma}_{11}^{-1}\boldsymbol{\mu}_1 = \mathbf{b}_1 - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{b}_2, \quad \mathbf{\Sigma}_{11}^{-1} = \mathbf{A}_{11} - \mathbf{A}_{12}\mathbf{A}_{22}^{-1}\mathbf{A}_{21}, \tag{2.5a}$$

$$\mathbf{\Sigma}_{22}^{-1}\boldsymbol{\mu}_2 = \mathbf{b}_2 - \mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{b}_1, \quad \mathbf{\Sigma}_{22}^{-1} = \mathbf{A}_{22} - \mathbf{A}_{21}\mathbf{A}_{11}^{-1}\mathbf{A}_{12}. \tag{2.5b}$$

We will find this marginalization form useful if we wish to modify our batch estimation formulation into a sliding-window filter for online application.

2.2 Maximum A Posteriori

We begin our derivation of batch state estimation with Bayes' theorem:

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{y})}, \tag{2.6}$$

where $\mathbf{x} = \{\mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_N\}$ is our discrete trajectory and $\mathbf{y} = \{\mathbf{y}_0, \mathbf{y}_1, \ldots, \mathbf{y}_M\}$ is a stacked quantity of all measurements. The quantity $p(\mathbf{x}|\mathbf{y})$ is the posterior probability density and the output from our estimation problem in which we are interested. Bayes' theorem shows that the posterior is a product of a generative model of the measurements given the trajectory, $p(\mathbf{y}|\mathbf{x})$, and a prior density over our trajectory, $p(\mathbf{x})$.

In practice, we generally cannot evaluate Bayes' theorem in a computationally feasible way (e.g., nonidealities that lead to a posterior that is not Gaussian). Instead we can take a Maximum A Posteriori (MAP) approach and optimize (2.6) for a Gaussian approximation of the posterior by noting that the denominator $p(\mathbf{y})$ does not depend on $\mathbf{x}$,

$$\boldsymbol{\mu}^* = \arg\max_{\mathbf{x}} p(\mathbf{y}|\mathbf{x})p(\mathbf{x}) = \arg\max_{\mathbf{x}} p(\mathbf{x}, \mathbf{y}), \tag{2.7}$$

where $\boldsymbol{\mu}^*$ is the mean of our optimized approximate posterior. This MAP approach is concerned with finding the most likely state as a point estimate, which is the mode[1] of the true Bayesian posterior distribution. We then obtain an estimate of the posterior covariance as a byproduct through the *Laplace approximation* (Bishop, 2006). We make note of this to contrast with the estimation framework that we introduce in Chapter 3, which optimizes a different objective for both the mean and covariance.

Often in state estimation for robotics, the joint likelihood, $p(\mathbf{x}, \mathbf{y})$, factors in a way such that there is sparsity that we can exploit for implementing large-scale trajectory problems efficiently. For convenience we work with the negative log-likelihood, which we can express as $\phi(\mathbf{x}, \mathbf{y}) = -\ln p(\mathbf{x}, \mathbf{y})$. In general we can assume that our joint likelihood factors, allowing us to rewrite our MAP objective as a cost function

$$J = \phi(\mathbf{x}, \mathbf{y}) = \sum_{k=1}^{K} \phi_k(\mathbf{x}_k, \mathbf{y}_k), \tag{2.8}$$

where $\phi_k(\cdot, \cdot)$ is the kth (negative log) factor expression, $\mathbf{x}_k$ is a subset of variables in $\mathbf{x}$ associated with the kth factor, and $\mathbf{y}_k$ is a subset of the data in $\mathbf{y}$ associated with the kth factor. We can optimize the cost function J iteratively for nonlinear problems. We perturb our current best estimate of the mean,

$$\mathbf{x} = \bar{\mathbf{x}} + \delta\mathbf{x}, \tag{2.9}$$

where $\bar{\mathbf{x}}$ is the nominal value and $\delta\mathbf{x}$ is a small perturbation vector. We can then optimize J for $\delta\mathbf{x}^*$ at each iteration. Taking the derivative of J with respect to $\delta\mathbf{x}$ and setting it to zero for an extremum will result in a linear system

$$\mathbf{A}\,\delta\mathbf{x}^* = \mathbf{b}. \tag{2.10}$$

Critically, there will be sparsity in the matrix on left-hand side of the linear system, $\mathbf{A}$, corresponding to the factorization of the joint likelihood. The matrix $\mathbf{A}$ is also the approximation of the inverse posterior covariance (i.e., the Laplace approximation). This

[1]Since the posterior is non-Gaussian, the mean and mode are not the same.

sparsity in the inverse covariance includes familiar patterns such as the block-tridiagonal sparsity in smoothing problems and the ‘arrowhead’ matrix in Simultaneous Localization and Mapping (SLAM). Using a sparse solver, we can efficiently compute the latest update to our mean:

$$\bar{\mathbf{x}}^{(i)} = \bar{\mathbf{x}}^{(i-1)} + \delta\mathbf{x}^*, \tag{2.11}$$

where we are showing the update to the mean at iteration i using our estimate from the previous iteration, $i-1$. We repeat this iterative process until convergence of J.

2.3 Estimation on SE(3)

We cannot in general use a vector space to describe the three-dimensional orientation of an object. There are several different ways to parameterize rotations and each approach has its advantages and disadvantages, which appear in the form of singularities and constraints. In this thesis, we choose to parameterize rotations (and poses) as *matrix Lie groups*, as outlined in the textbook by Barfoot (2024). We provide a brief summary of the *left perturbation* approach for expressing Gaussian uncertainty and handling linearization for optimization.

We use the *special Euclidean group*, $SE(3)$, to represent poses (i.e., translation and rotation). This is formally defined as

$$SE(3) = \left\{ \mathbf{T} = \begin{bmatrix} \mathbf{C} & \mathbf{r} \\ \mathbf{0}^T & 1 \end{bmatrix} \in \mathbb{R}^{4\times 4} \,\middle|\, \mathbf{C} \in SO(3), \mathbf{r} \in \mathbb{R}^3 \right\}, \tag{2.12}$$

where we define the rotation, $\mathbf{C}$, as an element of the *special orthogonal group*, $SO(3)$.

Using a left perturbation scheme, we can express Gaussian uncertainty on a pose estimate using

$$\mathbf{T} = \exp(\boldsymbol{\epsilon}^\wedge)\bar{\mathbf{T}}, \tag{2.13}$$

where $\bar{\mathbf{T}} \in SE(3)$ is the nominal pose estimate, $\boldsymbol{\epsilon} \in \mathbb{R}^6 \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$ is a small Gaussian random variable in vectorspace, $\exp(\cdot)$ is the exponential map, and the $(\cdot)^\wedge$ operator is defined as

$$\boldsymbol{\epsilon}^\wedge = \begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\phi} \end{bmatrix}^\wedge = \begin{bmatrix} \boldsymbol{\phi}^\wedge & \boldsymbol{\rho} \\ \mathbf{0}^T & 0 \end{bmatrix}, \quad \boldsymbol{\phi}^\wedge = \begin{bmatrix} \phi_1 \\ \phi_2 \\ \phi_3 \end{bmatrix}^\wedge = \begin{bmatrix} 0 & -\phi_3 & \phi_2 \\ \phi_3 & 0 & -\phi_1 \\ -\phi_2 & \phi_1 & 0 \end{bmatrix}. \tag{2.14}$$

Note that we are overloading the $(\cdot)^\wedge$ operator here for inputs that are $\mathbb{R}^3$ and $\mathbb{R}^6$.

We can also use the same perturbation scheme to take derivatives of our models with respect to a $SE(3)$ pose variable. The relevant example for this thesis is the following error function,

$$\mathbf{e}(\mathbf{T}) = \mathbf{p} - \mathbf{T}\mathbf{q}, \tag{2.15}$$

where $\mathbf{p}$ and $\mathbf{q}$ are homogeneous points, e.g.,

$$\mathbf{p} = \begin{bmatrix} p_1 & p_2 & p_3 & 1 \end{bmatrix}^T, \quad p_1, p_2, p_3 \in \mathbb{R}. \tag{2.16}$$

Using our left perturbaton scheme, we linearize the model as follows:

$$\begin{aligned} \mathbf{e}(\mathbf{T}) &\approx \mathbf{p} - \exp(\boldsymbol{\epsilon}^\wedge)\bar{\mathbf{T}}\mathbf{q} && (2.17) \\ &\approx \mathbf{p} - (\mathbf{1} + \boldsymbol{\epsilon}^\wedge)\bar{\mathbf{T}}\mathbf{q} && (2.18) \\ &= \underbrace{\mathbf{p} - \bar{\mathbf{T}}\mathbf{q}}_{\bar{\mathbf{e}}} - \left(\bar{\mathbf{T}}\mathbf{q}\right)^{\odot} \boldsymbol{\epsilon}. && (2.19) \end{aligned}$$

We applied the definition

$$\mathbf{p}^{\odot} = \begin{bmatrix} \boldsymbol{\varepsilon} \\ \eta \end{bmatrix}^{\odot} = \begin{bmatrix} \eta\mathbf{1} & -\boldsymbol{\varepsilon}^\wedge \\ \mathbf{0}^T & \mathbf{0}^T \end{bmatrix}. \tag{2.20}$$

Notice how our error function was originally expressed with respect to a $SE(3)$ pose variable, but after linearization, it is now a linear function with respect to a vectorspace perturbation, $\boldsymbol{\epsilon}$. We can linearize any model with $SE(3)$ pose variables in this way, resulting in a linear model with respect to a vectorspace perturbation.

It is now clear how we can apply our perturbation scheme to optimize a MAP cost function that is expressed in terms of $SE(3)$ pose variables. Using our chosen perturbation and linearizaton scheme, solving for the optimal (vectorspace) perturbation will be no different from what we showed in the previous section. After solving for the optimal perturbation, updates to the mean estimate at the latest iteration i can be done in a constraint-sensitive way for each pose variable in the state:

$$\bar{\mathbf{T}}^{(i)} = \exp((\boldsymbol{\epsilon}^*)^\wedge)\bar{\mathbf{T}}^{(i-1)}. \tag{2.21}$$

Chapter 3

Exactly Sparse Gaussian Variational Inference

In nonlinear batch state estimation, the true Bayesian posterior will not be a Gaussian PDF. In order to obtain a solution that is computationally tractable, we compromise by optimizing a point estimate that maximizes the posterior (i.e., the mode) using Maximum A Posteriori (MAP). In practice, MAP produces a reasonably accurate approximation of the true Bayesian posterior in the presence of mild nonlinearities and/or when the posterior is concentrated. However, there is potential for improved performance in comparison to MAP by additionally taking into consideration the covariance of the Gaussian in our optimization objective.

In this chapter, we propose formulating the estimation problem as Gaussian Variational Inference (GVI), which optimizes a Gaussian approximation that is 'closest' to the true posterior in terms of the Kullback-Leibler (KL) divergence. Similar to MAP, we demonstrate how to fit the best Gaussian to the posterior efficiently by exploiting factorization of the joint likelihood of the state and data. Our proposed method stores the inverse covariance matrix, which is exactly sparse (e.g., block-tridiagonal for classic state estimation) for many robotics problems. We show that only blocks of the (dense) covariance matrix that are required to be calculated correspond to the non-zero blocks of the inverse covariance matrix, which can be efficiently calculated in the general GVI problem. Exactly Sparse Gaussian Variational Inference (ESGVI) operates iteratively, where analytical derivatives can be used to optimize the GVI objective. We also show how Gaussian cubature can instead be applied as an alternative to analytical derivatives to produce an efficient derivative-free batch formulation.

In summary, the main contribution of this chapter is to show how GVI can be made tractable for large-scale trajectories and improve upon the performance of MAP in challenging estimation problems. We demonstrate the technique on controlled simulation problems and a batch nonlinear SLAM problem with an experimental dataset.

The work we present in this chapter is a second-author contribution to a journal publication in the International Journal of Robotics Research (Barfoot et al., 2020). Secondary contributions were made to the theory and methodology, which we present in Section 3.2 for completeness. The primary contribution as an author of this work was the implementation and analysis of the experiments, which we present in Sections 3.4, 3.5, and 3.6. Critically, Section 3.3 presents a sketch for parameter learning with the ESGVI estimation framework using EM. This idea is the foundation for this thesis, which we build upon and apply in the later chapters to solve practical problems in the real world.

3.1 Related Work

For Gaussian estimation, the well-known Kalman Filter (KF) is the optimal solution that recursively propagates the Gaussian state estimate for linear systems. If we wish to incorporate all measurements in time, the Rauch-Tung-Striebel (RTS) smoother (Rauch et al., 1965) applies forward and backward passes to efficiently estimate the entire state. In practice, our models tend to be nonlinear, for which we can apply more advanced methods. Särkkä (2013) presents an overview of recursive inference methods for both linear and nonlinear models.

However, the work we present in this chapter focuses on a more general problem setup with batch Gaussian inference. Canonical problems in robotics are batch trajectory estimation, pose-graph relaxation (Bourmaud, 2016), and Bundle Adjustment (BA) (Brown, 1958) / SLAM (Durrant-Whyte and Bailey, 2006). Other problems we can apply batch inference include control/planning (Dong et al., 2016; Mukadam et al., 2018), calibration (Pradeep et al., 2014), and three-dimensional modelling problems (Li et al., 2011). What distinguishes these problems from the recursive/smoothing problem discussed above is that the factorization of the joint likelihood between the state and observed data is not limited to a linear chain (i.e., the inverse covariance sparsity is not block-tridiagonal). Despite not having a fixed factorization and sparsity pattern, we can still exploit the existing problem-specific sparsity for a computationally efficient solution. The predominate method for doing so is Maximum A Posteriori (MAP) estimation.

In MAP estimation, we optimize a Gaussian fit by assigning the most likely state in the true Bayesian posterior as the mean of our approximation, with a corresponding covariance referred to as the Laplace approximation (Bishop, 2006). The MAP estimate will produce the exact Gaussian posterior for linear systems, but that is no longer true when our models are nonlinear. The primary goal of the work in this chapter is to revisit the batch Gaussian inference problem in search of improvements over this popular method.

Our approach for improving upon MAP is to formulate our batch inference problem using variational Bayes (Bishop, 2006), which optimizes for the closest Gaussian approximation to the true Bayesian posterior using the KL divergence between the two (Kullback and Leibler, 1951). In other words, we will be optimizing for both the mean and covariance of our Gaussian approximation, which differs from MAP where we only optimize for the mean and compute a Laplace-style covariance post hoc. The challenge is to be able to do this efficiently for problems with a large state size. Naively estimating the covariance for large problems is not computationally feasible since in general it will be a large, dense matrix. We will demonstrate how we can apply full GVI by exploiting the same problem-specific sparsity that we usually do in the MAP approach.

This is made possible in our formulation of GVI because the sub-blocks of the covariance that we require are precisely the ones corresponding to the non-zero sub-blocks of the inverse covariance (which is typically highly sparse). Thankfully, we are able to extract the required sub-blocks of the covariance efficiently. For example, in smoothing problems with a block-tridiagonal inverse covariance, the complexity of the solver for the mean is linear in the trajectory length. We can additionally calculate the corresponding covariance sub-blocks (i.e., the three main block diagonals of the covariance) (Meurant, 1992; Barfoot, 2024) efficiently by reusing the calculations for the mean solve, maintaining the linear complexity of the problem. It turns out we are still able to compute the required sub-blocks efficiently in the general case. We use a method first proposed by Takahashi et al. (1973) in the context of circuit theory, which was later used by Broussolle (1978) in a state estimation context. Erisman and Tinney (1975) provide a proof of the closure of the Takahashi et al. method and also discuss algorithmic complexity. More recently, Triggs et al. (2000) and Kaess and Dellaert (2009) present methods to calculate specific blocks of the covariance matrix efficiently for computer vision and robotics applications, although they do not discuss computing the complete set of covariance sub-blocks that correspond to the non-zero sub-blocks of the inverse covariance.

The idea of GVI is not new, and has been applied in other existing works. Opper and Archambeau (2009) discuss a similar GVI approach in machine learning, where they applied the method to Gaussian process regression problems (Rasmussen and Williams, 2006). Kokkala et al. (2014, 2016), Ala-Luhtala et al. (2015), García-Fernández et al. (2015), Gašperin and Juričić (2011), and Schön et al. (2011a) discuss a very similar approach to our GVI scheme in the context of nonlinear smoothers and filters. They show

how to exploit factorization of the joint likelihood (specific to smoothing problems), and discuss how to apply sigmapoints (Kokkala et al., 2014, 2016; Gašperin and Juričić, 2011) or particles (Schön et al., 2011a) to avoid the need to compute derivatives. Our method extends these works by (i) generalizing to any large-scale batch GVI problems where the likelihood factors (not restricted to smoothing problems with block-tridiagonal inverse covariance), (ii) devising a Newton-style iterative solver for both mean and covariance, (iii) explicitly showing how to exploit problem-specific structure in the case of a factored likelihood to make the technique efficient, and (iv) demonstrating the approach on problems of interest in robotics. Similar to some of these works, we also demonstrate how to apply Gaussian cubature in our approach to avoid the need to calculate derivatives.

3.2 Gaussian Variational Inference

3.2.1 Loss Functional

In variational inference, our objective is to minimize the KL divergence (Kullback and Leibler, 1951) between the true Bayesian posterior, $p(\mathbf{x}|\mathbf{y})$, and an approximation of the posterior, $q(\mathbf{x})$. As we are applying Gaussian variational inference, our approximation will be a multivariate Gaussian, $q(\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. The KL divergence we choose to use for our loss is

$$\mathrm{KL}(q||p) = -\int_{-\infty}^{\infty} q(\mathbf{x}) \ln\left(\frac{p(\mathbf{x}|\mathbf{y})}{q(\mathbf{x})}\right) d\mathbf{x} = \mathbb{E}_q\left[\ln q(\mathbf{x}) - \ln p(\mathbf{x}|\mathbf{y})\right], \tag{3.1}$$

where $\mathbf{x}$ is our state, $\mathbf{y}$ represents our measurements, and $\mathbb{E}[\cdot]$ is the expectation operator. We choose the divergence expressed in this order, rather than the alternative $\mathrm{KL}(p||q)$, to have the expectation over our Gaussian estimate, $q(\mathbf{x})$, rather than the unknown true posterior. This key practical difference will allow us to evaluate the expectation and devise an efficient iterative scheme for $q(\mathbf{x})$ that best approximates the posterior.

The divergence objective can be rewritten in the following way,

$$\mathrm{KL}(q||p) = \mathbb{E}_q[-\ln p(\mathbf{x},\mathbf{y})] \underbrace{- \frac{1}{2}\ln\left((2\pi e)^N |\boldsymbol{\Sigma}|\right)}_{\text{entropy}} + \underbrace{\ln p(\mathbf{y})}_{\text{constant}}, \tag{3.2}$$

where we used the expression for the *entropy*, $-\int q(\mathbf{x}) \ln q(\mathbf{x}) d\mathbf{x}$, for a Gaussian. The final term is a constant (i.e., it does not depend on $q(\mathbf{x})$), allowing us to define the following loss functional that we seek to minimize with respect to $q(\mathbf{x})$:

$$V(q) = \mathbb{E}_q[\phi(\mathbf{x}, \mathbf{y})] + \frac{1}{2}\ln\left(|\boldsymbol{\Sigma}^{-1}|\right), \tag{3.3}$$

where $\phi(\mathbf{x}, \mathbf{y}) = -\ln p(\mathbf{x}, \mathbf{y})$. We choose to express our loss using $\boldsymbol{\Sigma}^{-1}$ (inverse covariance matrix), rather than $\boldsymbol{\Sigma}$ (covariance matrix), since the former has sparsity that we can exploit. Interpreting our loss functional, we can see that the first term encourages the solution to match the data while the second penalizes it for being too certain. We also note that our loss functional is the negative of the ELBO, which is important for our approach to parameter learning that we will discuss later.

3.2.2 Optimization Scheme

Our goal is to optimize our loss functional, $V(q)$, with respect to our approximate posterior, $q(\mathbf{x})$, i.e., the mean, $\boldsymbol{\mu}$, and the inverse covariance, $\boldsymbol{\Sigma}^{-1}$. Starting with a Taylor series expansion of $V(q)$,

$$V\left(q^{(i+1)}\right) \approx V\left(q^{(i)}\right) + \left(\left.\frac{\partial V(q)}{\partial \boldsymbol{\mu}^T}\right|_{q^{(i)}}\right)^T \delta\boldsymbol{\mu} + \frac{1}{2}\delta\boldsymbol{\mu}^T\left(\left.\frac{\partial^2 V(q)}{\partial \boldsymbol{\mu}^T \partial \boldsymbol{\mu}}\right|_{q^{(i)}}\right)\delta\boldsymbol{\mu} + \operatorname{tr}\left(\left.\frac{\partial V(q)}{\partial \boldsymbol{\Sigma}^{-1}}\right|_{q^{(i)}} \delta\boldsymbol{\Sigma}^{-1}\right), \tag{3.4}$$

where $\delta\boldsymbol{\mu} = \boldsymbol{\mu}^{(i+1)} - \boldsymbol{\mu}^{(i)}$ and $\delta\boldsymbol{\Sigma}^{-1} = \left(\boldsymbol{\Sigma}^{-1}\right)^{(i+1)} - \left(\boldsymbol{\Sigma}^{-1}\right)^{(i)}$ with i the iteration index of our scheme. This expansion is second order in $\delta\boldsymbol{\mu}$, but only first order in $\delta\boldsymbol{\Sigma}^{-1}$. We need to update $\delta\boldsymbol{\mu}$ and $\delta\boldsymbol{\Sigma}^{-1}$ to make $V(q)$ get smaller, i.e., $V\left(q^{(i+1)}\right) - V\left(q^{(i)}\right) \le 0$.

The derivatives of our loss functional with respect to our Gaussian parameters are given by (Opper and Archambeau, 2009)

$$\frac{\partial V(q)}{\partial \boldsymbol{\mu}^T} = \boldsymbol{\Sigma}^{-1}\mathbb{E}_q[(\mathbf{x} - \boldsymbol{\mu})\phi(\mathbf{x}, \mathbf{y})], \tag{3.5a}$$

$$\frac{\partial^2 V(q)}{\partial \boldsymbol{\mu}^T \partial \boldsymbol{\mu}} = \boldsymbol{\Sigma}^{-1}\mathbb{E}_q[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T\phi(\mathbf{x}, \mathbf{y})]\boldsymbol{\Sigma}^{-1} - \boldsymbol{\Sigma}^{-1}\,\mathbb{E}_q[\phi(\mathbf{x}, \mathbf{y})], \tag{3.5b}$$

$$\frac{\partial V(q)}{\partial \boldsymbol{\Sigma}^{-1}} = -\frac{1}{2}\mathbb{E}_q[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T\phi(\mathbf{x}, \mathbf{y})] + \frac{1}{2}\boldsymbol{\Sigma}\,\mathbb{E}_q[\phi(\mathbf{x}, \mathbf{y})] + \frac{1}{2}\boldsymbol{\Sigma}. \tag{3.5c}$$

We note that it is not (in general) possible to isolate for $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}^{-1}$ in closed form, which is why we choose to formulate an iterative, Newton-style optimizer.

Comparing (3.5b) and (3.5c), we can get the relationship

$$\frac{\partial^2 V(q)}{\partial \boldsymbol{\mu}^T \partial \boldsymbol{\mu}} = \boldsymbol{\Sigma}^{-1} - 2\boldsymbol{\Sigma}^{-1}\frac{\partial V(q)}{\partial \boldsymbol{\Sigma}^{-1}}\boldsymbol{\Sigma}^{-1}. \tag{3.6}$$

We can set the derivative, $\frac{\partial V(q)}{\partial \boldsymbol{\Sigma}^{-1}}$, to zero for an extremum to get

$$\boldsymbol{\Sigma}^{-1^{(i+1)}} = \left. \frac{\partial^2 V(q)}{\partial \boldsymbol{\mu}^T \partial \boldsymbol{\mu}} \right|_{q^{(i)}}, \tag{3.7}$$

where we use the indices $(i+1)$ and (i) to indicate our iterative update. Inserting (3.6) on the right side, we can rearrange and get our update for the inverse covariance,

$$\delta \boldsymbol{\Sigma}^{-1} = -2(\boldsymbol{\Sigma}^{-1})^{(i)} \left. \frac{\partial V(q)}{\partial \boldsymbol{\Sigma}^{-1}} \right|_{q^{(i)}} (\boldsymbol{\Sigma}^{-1})^{(i)}. \tag{3.8}$$

For the mean, $\boldsymbol{\mu}$, we refer back to our Taylor series approximation in (3.4). Since this approximation is locally quadratic in $\delta\boldsymbol{\mu}$, we can take the derivative with respect to $\delta\boldsymbol{\mu}$ and set this to zero to formulate a Newton-style update (Nocedal and Wright, 2006):

$$\underbrace{\left(\left. \frac{\partial^2 V(q)}{\partial \boldsymbol{\mu}^T \partial \boldsymbol{\mu}} \right|_{q^{(i)}} \right)}_{(\boldsymbol{\Sigma}^{-1})^{(i+1)}} \delta\boldsymbol{\mu} = - \left(\left. \frac{\partial V(q)}{\partial \boldsymbol{\mu}^T} \right|_{q^{(i)}} \right), \tag{3.9}$$

where, similar to the MAP approach, we have $\boldsymbol{\Sigma}^{-1}$ on the left-hand side.

Inserting our update scheme for $\delta\boldsymbol{\mu}$ and $\delta\boldsymbol{\Sigma}^{-1}$ into (3.4), we have

$$V\left(q^{(i+1)}\right) - V\left(q^{(i)}\right) \approx -\frac{1}{2} \underbrace{\delta\boldsymbol{\mu}^T \left(\boldsymbol{\Sigma}^{-1}\right)^{(i+1)} \delta\boldsymbol{\mu}}_{\substack{\geq 0 \\ \text{with equality iff } \delta\boldsymbol{\mu} = \mathbf{0}}} -\frac{1}{2} \underbrace{\operatorname{tr}\left(\boldsymbol{\Sigma}^{(i)}\, \delta\boldsymbol{\Sigma}^{-1}\, \boldsymbol{\Sigma}^{(i)}\, \delta\boldsymbol{\Sigma}^{-1}\right)}_{\substack{\geq 0 \\ \text{with equality iff } \delta\boldsymbol{\Sigma}^{-1} = \mathbf{0}}} \leq 0, \tag{3.10}$$

which shows that our update will always make our loss, $V(q)$, smaller as long as $\delta\boldsymbol{\mu}$ and $\delta\boldsymbol{\Sigma}^{-1}$ are not both zero. This will be true when the derivatives with respect to $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}^{-1}$ are not both zero, which only occurs at a local minimum of $V(q)$. This is a local convergence guarantee based on our Taylor series expansion in (3.4).

We can reformulate our update scheme using *Stein's lemma* (Stein, 1981):

$$\mathbb{E}_q[(\mathbf{x} - \boldsymbol{\mu}) f(\mathbf{x})] \equiv \boldsymbol{\Sigma}\, \mathbb{E}_q \left[\frac{\partial f(\mathbf{x})}{\partial \mathbf{x}^T} \right], \tag{3.11}$$

where $q(\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is a Gaussian random variable and $f(\cdot)$ is any nonlinear differentiable function. Assuming $f(\cdot)$ is twice differentiable, a double application of Stein's lemma gives us

$$\mathbb{E}_q[(\mathbf{x}-\boldsymbol{\mu})(\mathbf{x}-\boldsymbol{\mu})^T f(\mathbf{x})] \equiv \boldsymbol{\Sigma}\,\mathbb{E}_q\left[\frac{\partial^2 f(\mathbf{x})}{\partial \mathbf{x}^T \partial \mathbf{x}}\right]\boldsymbol{\Sigma} + \boldsymbol{\Sigma}\,\mathbb{E}_q[f(\mathbf{x})]. \tag{3.12}$$

We can apply (3.11) and (3.12) to (3.7) and (3.9) to rewrite our update equations as

$$\left(\boldsymbol{\Sigma}^{-1}\right)^{(i+1)} = \mathbb{E}_{q^{(i)}}\left[\frac{\partial^2}{\partial \mathbf{x}^T \partial \mathbf{x}}\phi(\mathbf{x},\mathbf{y})\right], \tag{3.13a}$$

$$\left(\boldsymbol{\Sigma}^{-1}\right)^{(i+1)} \delta\boldsymbol{\mu} = -\mathbb{E}_{q^{(i)}}\left[\frac{\partial}{\partial \mathbf{x}^T}\phi(\mathbf{x},\mathbf{y})\right], \tag{3.13b}$$

$$\boldsymbol{\mu}^{(i+1)} = \boldsymbol{\mu}^{(i)} + \delta\boldsymbol{\mu}. \tag{3.13c}$$

Ala-Luhtala et al. (2015) also apply Stein's lemma in this way in the context of Gaussian variational smoothers. Since only the first and second derivatives of $\phi(\mathbf{x},\mathbf{y})$ are required, we can drop any constant terms (i.e., the normalization constant of $p(\mathbf{x},\mathbf{y})$). Also note how these equations are identical to those of the MAP approach if we only evaluate the expectations at the mean of $q(\mathbf{x})$. We can view MAP with the Laplace covariance approximation as one possible approximation of our GVI approach. Consequently, our GVI approach will be equivalent to the discrete-time RTS smoother in the linear case (i.e., when the true posterior is Gaussian) (Barfoot, 2024).

3.2.3 Exploiting Sparsity

Upon first glance of our loss functional and iterative update scheme in (3.13), it is not clear how these updates can be calculated efficiently. We require the calculation of these three expectations:

$$\underbrace{\mathbb{E}_q[\phi(\mathbf{x},\mathbf{y})]}_{\text{scalar}},\quad \underbrace{\mathbb{E}_q\left[\frac{\partial}{\partial \mathbf{x}^T}\phi(\mathbf{x},\mathbf{y})\right]}_{\text{column}},\quad \underbrace{\mathbb{E}_q\left[\frac{\partial^2}{\partial \mathbf{x}^T \partial \mathbf{x}}\phi(\mathbf{x},\mathbf{y})\right]}_{\text{matrix}}. \tag{3.14}$$

Note that we have dropped the iteration index for convenience. The expectations are with respect to the full Gaussian estimate, $q(\mathbf{x})$, which will be too expensive to be computationally feasible for large-scale trajectories as the covariance, $\boldsymbol{\Sigma}$, will (in general) be dense.

Often in state estimation for robotics, the joint likelihood factors in a way such that there is sparsity that we can exploit for implementing large-scale trajectory problems efficiently. This sparsity consequently appears as the sparsity in the inverse covariance,

$\boldsymbol{\Sigma}^{-1}$, and includes familiar patterns such as the block-tridiagonal sparsity in smoothing problems and the 'arrowhead' matrix in SLAM. Recalling that we prefer to work with the negative log-likelihood, $\phi(\mathbf{x}, \mathbf{y}) = -\ln p(\mathbf{x}, \mathbf{y})$, in general we will assume that our joint likelihood factors in the following way:

$$\phi(\mathbf{x}, \mathbf{y}) = \sum_{k=1}^{K} \phi_k(\mathbf{x}_k, \mathbf{y}_k), \tag{3.15}$$

where $\phi_k(\cdot, \cdot)$ is the kth (negative log) factor expression, $\mathbf{x}_k$ is a subset of variables in $\mathbf{x}$ associated with the kth factor, and $\mathbf{y}_k$ is a subset of the data in $\mathbf{y}$ associated with the kth factor.

Consider the first (scalar) expectation in (3.14). Inserting the factored likelihood,

$$\mathbb{E}_q[\phi(\mathbf{x}, \mathbf{y})] = \mathbb{E}_q\left[\sum_{k=1}^{K} \phi_k(\mathbf{x}_k, \mathbf{y}_k)\right] = \sum_{k=1}^{K} \mathbb{E}_q[\phi_k(\mathbf{x}_k, \mathbf{y}_k)] = \sum_{k=1}^{K} \mathbb{E}_{q_k}[\phi_k(\mathbf{x}_k, \mathbf{y}_k)], \tag{3.16}$$

where the last step shows (without approximation) that the expectation simplifies from being over $q = q(\mathbf{x})$, the full Gaussian estimate, to being over $q_k = q_k(\mathbf{x}_k)$, the *marginal* of the estimate for just the variables in each factor.

The other two expectations in (3.14) can also be simplified in a similar way. Let $\mathbf{P}_k$ be a selection matrix such that it extracts $\mathbf{x}_k$ from $\mathbf{x}$:

$$\mathbf{x}_k = \mathbf{P}_k \mathbf{x}. \tag{3.17}$$

Inserting the factored expression into the second (column) expectation, we have

$$\begin{aligned} \mathbb{E}_q\left[\frac{\partial}{\partial \mathbf{x}^T}\phi(\mathbf{x}, \mathbf{y})\right] &= \mathbb{E}_q\left[\frac{\partial}{\partial \mathbf{x}^T}\sum_{k=1}^{K} \phi_k(\mathbf{x}_k, \mathbf{y}_k)\right] = \sum_{k=1}^{K} \mathbb{E}_q\left[\frac{\partial}{\partial \mathbf{x}^T}\phi_k(\mathbf{x}_k, \mathbf{y}_k)\right] \\ &= \sum_{k=1}^{K} \mathbf{P}_k^T \, \mathbb{E}_q\left[\frac{\partial}{\partial \mathbf{x}_k^T}\phi_k(\mathbf{x}_k, \mathbf{y}_k)\right] = \sum_{k=1}^{K} \mathbf{P}_k^T \, \mathbb{E}_{q_k}\left[\frac{\partial}{\partial \mathbf{x}_k^T}\phi_k(\mathbf{x}_k, \mathbf{y}_k)\right]. \end{aligned} \tag{3.18}$$

For factor k, we are able to simplify the derivative from being with respect to $\mathbf{x}$, to being with respect to $\mathbf{x}_k$. We use the selection matrix (as a dilation matrix) to map the derivative back into the appropriate rows of the overall result. We see that the expectation again simplifies to being with respect to $q_k = q_k(\mathbf{x}_k)$, the marginal of the estimate for just the variables in factor k. For the last (matrix) expectation, we can follow similar steps to show:

$$\begin{aligned}\mathbb{E}_q\left[\frac{\partial^2}{\partial \mathbf{x}^T \partial \mathbf{x}} \phi(\mathbf{x},\mathbf{y})\right] &= \mathbb{E}_q\left[\frac{\partial^2}{\partial \mathbf{x}^T \partial \mathbf{x}} \sum_{k=1}^{K} \phi_k(\mathbf{x}_k,\mathbf{y}_k)\right] = \sum_{k=1}^{K} \mathbb{E}_q\left[\frac{\partial^2}{\partial \mathbf{x}^T \partial \mathbf{x}} \phi_k(\mathbf{x}_k,\mathbf{y}_k)\right] \\ &= \sum_{k=1}^{K} \mathbf{P}_k^T\, \mathbb{E}_q\left[\frac{\partial^2}{\partial \mathbf{x}_k^T \partial \mathbf{x}_k} \phi_k(\mathbf{x}_k,\mathbf{y}_k)\right] \mathbf{P}_k = \sum_{k=1}^{K} \mathbf{P}_k^T\, \mathbb{E}_{q_k}\left[\frac{\partial^2}{\partial \mathbf{x}_k^T \partial \mathbf{x}_k} \phi_k(\mathbf{x}_k,\mathbf{y}_k)\right] \mathbf{P}_k. \end{aligned} \tag{3.19}$$

Now it is clear that we do not require the full Gaussian estimate, $q(\mathbf{x})$, which would require a large amount of compute and storage in large-scale trajectories due to the dense covariance, $\boldsymbol{\Sigma}$. We only require the marginals of $q(\mathbf{x})$ that correspond to each factor. We can also see that from (3.13a) and (3.19) that $\boldsymbol{\Sigma}^{-1}$ is exactly sparse (with its pattern depending on the nature of the factors) and that the sparsity pattern will be constant in our iterative update scheme.

Some of these remarks may seem familiar to those used to working with a MAP approach to batch state estimation (e.g., the sparsity pattern of $\boldsymbol{\Sigma}^{-1}$ exists and is constant across iterations). But now we are performing GVI that iterates over a full Gaussian PDF (i.e., mean and covariance) not just a point estimate (i.e., mean only). What remains is how we can efficiently compute the sub-blocks of the covariance that we need, which we discuss in the following subsection.

3.2.4 Partial Computation of the Covariance

At each iteration of our GVI approach, we need to solve a system of linear equations for the update in the mean:

$$\boldsymbol{\Sigma}^{-1}\,\delta\boldsymbol{\mu} = \mathbf{r}, \tag{3.20}$$

where $\mathbf{r}$ is the right-hand side in (3.13b). We start by applying a sparse lower-diagonal-upper decomposition,

$$\boldsymbol{\Sigma}^{-1} = \mathbf{L}\mathbf{D}\mathbf{L}^T, \tag{3.21}$$

where $\mathbf{D}$ is diagonal and $\mathbf{L}$ is a sparse, lower-triangular matrix with ones on the main diagonal. The sparsity of this decomposition depends on the factorization of the joint likelihood (i.e., the nature of the prior and measurement factors). We can then solve for the mean update similarly to what is done in MAP (e.g., sparse forward and backward substitution).

For the covariance, $\boldsymbol{\Sigma}$, we only need to compute the sub-blocks that correspond to the non-zero sub-blocks of $\boldsymbol{\Sigma}^{-1}$. We will apply the method of Takahashi et al. (1973), which we summarize here for completeness. First, we notice that

$$\mathbf{L}\mathbf{D}\mathbf{L}^T\boldsymbol{\Sigma} = \mathbf{1}, \tag{3.22}$$

where $\mathbf{1}$ is the identity matrix. Premultiplying by the inverse of $\mathbf{LD}$, we get

$$\mathbf{L}^T\boldsymbol{\Sigma} = \mathbf{D}^{-1}\mathbf{L}^{-1}, \tag{3.23}$$

where $\mathbf{L}^{-1}$ will in general no longer be sparse. Taking the transpose and adding $\boldsymbol{\Sigma} - \boldsymbol{\Sigma}\mathbf{L}$ to both sides we have (Takahashi et al., 1973)

$$\boldsymbol{\Sigma} = \mathbf{L}^{-T}\mathbf{D}^{-1} + \boldsymbol{\Sigma}\left(\mathbf{1} - \mathbf{L}\right). \tag{3.24}$$

Since $\boldsymbol{\Sigma}$ is symmetric, we only need to (at most) compute the main diagonal and the lower-half blocks, which can be done through a backward substitution pass. To see this we expand the lower-half blocks as follows:

$$\begin{bmatrix} \ddots & & & \\ \cdots & \boldsymbol{\Sigma}_{K-2,K-2} & & \\ \cdots & \boldsymbol{\Sigma}_{K-1,K-2} & \boldsymbol{\Sigma}_{K-1,K-1} & \\ \cdots & \boldsymbol{\Sigma}_{K,K-2} & \boldsymbol{\Sigma}_{K,K-1} & \boldsymbol{\Sigma}_{K,K} \end{bmatrix} = \begin{bmatrix} \ddots & & & \\ & \mathbf{D}^{-1}_{K-2,K-2} & & \\ & & \mathbf{D}^{-1}_{K-1,K-1} & \\ & & & \mathbf{D}^{-1}_{K,K} \end{bmatrix}$$
$$- \begin{bmatrix} \ddots & \vdots & \vdots & \vdots \\ \cdots & \boldsymbol{\Sigma}_{K-2,K-2} & \boldsymbol{\Sigma}_{K-2,K-1} & \boldsymbol{\Sigma}_{K-2,K} \\ \cdots & \boldsymbol{\Sigma}_{K-1,K-2} & \boldsymbol{\Sigma}_{K-1,K-1} & \boldsymbol{\Sigma}_{K-1,K} \\ \cdots & \boldsymbol{\Sigma}_{K,K-2} & \boldsymbol{\Sigma}_{K,K-1} & \boldsymbol{\Sigma}_{K,K} \end{bmatrix} \begin{bmatrix} \ddots & & & \\ \cdots & \mathbf{0} & & \\ \cdots & \mathbf{L}_{K-1,K-2} & \mathbf{0} & \\ \cdots & \mathbf{L}_{K,K-2} & \mathbf{L}_{K,K-1} & \mathbf{0} \end{bmatrix}, \tag{3.25}$$

where we only show the blocks necessary for the calculation of the lower-half of $\boldsymbol{\Sigma}$. We drop $\mathbf{L}^{-T}$ since it only affects the upper-half blocks of $\boldsymbol{\Sigma}$ and is therefore unnecessary. Temporarily ignoring the need to exploit sparsity, we see that we can calculate the lowerbasic sparsity constraint (note fill in at $(5, 3)$ in $\mathbf{L}$)

$$\boldsymbol{\Sigma}^{-1} = \begin{bmatrix} * & & * & & * & \\ & * & & & & \\ * & & * & & & \\ & & & * & & \\ * & & & & * & \\ & & & & & * \end{bmatrix}$$

$$\mathbf{L} = \begin{bmatrix} * & & & & & \\ & * & & & & \\ * & & * & & & \\ & & & * & & \\ * & & + & & * & \\ & & & & & * \end{bmatrix}$$

trajectory example (6 robot poses)

$$\boldsymbol{\Sigma}^{-1} = \begin{bmatrix} * & * & & & & \\ * & * & * & & & \\ & * & * & * & & \\ & & * & * & * & \\ & & & * & * & * \\ & & & & * & * \end{bmatrix}$$

$$\mathbf{L} = \begin{bmatrix} * & & & & & \\ * & * & & & & \\ & * & * & & & \\ & & * & * & & \\ & & & * & * & \\ & & & & * & * \end{bmatrix}$$

SLAM example (3 poses, 3 landmarks)

$$\boldsymbol{\Sigma}^{-1} = \left[\begin{array}{ccc|ccc} * & * & & * & * & * \\ * & * & * & * & * & * \\ & * & * & * & * & * \\ \hline * & * & * & * & & \\ * & * & * & & * & \\ * & * & * & & & * \end{array}\right]$$

$$\mathbf{L} = \left[\begin{array}{ccc|ccc} * & & & & & \\ * & * & & & & \\ & * & * & & & \\ \hline * & * & * & * & & \\ * & * & * & + & * & \\ * & * & * & + & + & * \end{array}\right]$$

Figure 3.1: Example sparsity patterns of $\boldsymbol{\Sigma}^{-1}$ and the corresponding sparsity of the lower-triangular factor $\mathbf{L}$. The set of zero entries (whitespace) of the lower-half of $\mathbf{L}$ is a subset of the zero entries of the lower-half of $\boldsymbol{\Sigma}^{-1}$. There are some extra non-zero entries of $\mathbf{L}$, shown as $+$, that arise from completing the 'four corners of a box' as shown in the first example.

half blocks of $\boldsymbol{\Sigma}$ through backward substitution:

$$\boldsymbol{\Sigma}_{K,K} = \mathbf{D}_{K,K}^{-1}, \tag{3.26a}$$

$$\boldsymbol{\Sigma}_{K,K-1} = -\boldsymbol{\Sigma}_{K,K}\mathbf{L}_{K,K-1}, \tag{3.26b}$$

$$\boldsymbol{\Sigma}_{K-1,K-1} = \mathbf{D}_{K-1,K-1}^{-1} - \boldsymbol{\Sigma}_{K-1,K}\mathbf{L}_{K,K-1}, \tag{3.26c}$$

$$\vdots$$

$$\boldsymbol{\Sigma}_{j,k} = \delta(j,k)\,\mathbf{D}_{j,k}^{-1} - \sum_{\ell=k+1}^{K} \boldsymbol{\Sigma}_{j,\ell}\mathbf{L}_{\ell,k}, \qquad (j \geq k) \tag{3.26d}$$

where $\delta(\cdot,\cdot)$ is the Kronecker delta function.

Now recall that we do not want to compute all blocks of the covariance as that will be too expensive. We will make use of an observation that (in general) blocks that are zero in $\mathbf{L}$ will also be zero in $\boldsymbol{\Sigma}^{-1}$, but not the other way around. This is because the sparsity of the lower-half of $\mathbf{L}$ matches the sparsity of the lower-half of $\boldsymbol{\Sigma}^{-1}$, except that $\mathbf{L}$ can have a few more non-zero entries to ensure the sparsity of $\boldsymbol{\Sigma}^{-1}$ when multiplied together. Figure 3.1 shows example sparsity patterns for $\mathbf{\Sigma}^{-1}$ and the corresponding sparsity pattern of $\mathbf{L}$. Specifically, if $\mathbf{L}_{k,i} \neq \mathbf{0}$ and $\mathbf{L}_{j,i} \neq \mathbf{0}$, then we must have $\mathbf{L}_{j,k} \neq \mathbf{0}$ (Erisman and Tinney, 1975). This can be visualized as completing the 'four corners of a box', as shown in the first column of Figure 3.1.

Finally, we follow the explanation of Erisman and Tinney (1975) to understand why we do not need to calculate all of the blocks of $\mathbf{\Sigma}$. We want to compute all the blocks of the lower-half of $\mathbf{\Sigma}$ corresponding to the non-zero blocks of $\mathbf{L}$. In equation (3.26d), we see that if $\mathbf{L}_{p,k}$ is non-zero, then we need $\mathbf{\Sigma}_{j,p}$ to compute the non-zero block $\mathbf{\Sigma}_{j,k}$. But if $\mathbf{\Sigma}_{j,k}$ is non-zero, $\mathbf{L}_{j,k}$ must also be non-zero. Then using our 'four corners of a box' rule, $\mathbf{L}_{j,p}$ must be non-zero and so we will have $\mathbf{\Sigma}_{j,p}$ and $\mathbf{\Sigma}_{p,j} = \mathbf{\Sigma}_{j,p}^T$ on our list of blocks to compute already. This shows the calculation of the desired blocks is closed under the scheme defined by (3.26d), which in turn implies that there will always exist an efficient algorithm to calculate the blocks of $\mathbf{\Sigma}$ corresponding to the non-zero blocks of $\mathbf{\Sigma}^{-1}$, plus a few more according to the 'four corners of a box' rule.

We have shown that in general, the calculation of the required blocks of $\mathbf{\Sigma}$ (corresponding to the non-zero block of $\mathbf{\Sigma}^{-1}$) can be piggybacked efficiently onto the solution of (3.13b). The bottleneck for computational complexity is the original lower-diagonal-upper decomposition, which is also required for MAP. Therefore our ESGVI approach has the same order of computational cost (as a function of the state size) as MAP for a given problem, but will have a higher coefficient due to the extra burden of using the marginals to compute expectations.

3.2.5 Marginal Sampling and Derivative-Free Optimization

In Section 3.2.3, we demonstrated that we only need to calculate the marginal expectations at the factor level, which can then be reassembled back into the larger expectations of (3.14). These expectations are

$$\underbrace{\mathbb{E}_{q_k}[\phi_k(\mathbf{x}_k, \mathbf{y}_k)]}_{\text{scalar}}, \quad \underbrace{\mathbb{E}_{q_k}\left[\frac{\partial}{\partial \mathbf{x}_k^T}\phi_k(\mathbf{x}_k, \mathbf{y}_k)\right]}_{\text{column}}, \quad \underbrace{\mathbb{E}_{q_k}\left[\frac{\partial^2}{\partial \mathbf{x}_k^T \partial \mathbf{x}_k}\phi_k(\mathbf{x}_k, \mathbf{y}_k)\right]}_{\text{matrix}}. \tag{3.27}$$

We note that we have not made any assumptions on the specific form of the factors, $\phi_k(\cdot, \cdot)$, including the differentiability of the factors (despite the first and second derivatives in the equations). If the factors are twice differentiable, we can proceed as is and use a Gaussian cubature approximation for the expectations. Alternatively, motivated by existing sampling-based filters, such as the unscented Kalman filter (Julier and Uhlmann, 1996), the cubature Kalman filter (Arasaratnam and Haykin, 2009), and the Gauss-Hermite Kalman filter (Ito and Xiong, 2000)(Wu et al., 2006), we present an alternative method to compute the terms in (3.27) that is derivative-free.

To avoid the need to compute derivatives of $\phi_k(\cdot,\cdot)$, we can apply Stein's lemma in the opposite direction from our previous use. Using (3.11) we have

$$\mathbb{E}_{q_k}\left[\frac{\partial}{\partial \mathbf{x}_k^T}\phi_k(\mathbf{x}_k,\mathbf{y}_k)\right] = \boldsymbol{\Sigma}_{kk}^{-1}\mathbb{E}_{q_k}[(\mathbf{x}_k-\boldsymbol{\mu}_k)\phi_k(\mathbf{x}_k,\mathbf{y}_k)], \tag{3.28}$$

and using (3.12) we have

$$\mathbb{E}_{q_k}\left[\frac{\partial^2}{\partial \mathbf{x}_k^T\partial \mathbf{x}_k}\phi_k(\mathbf{x}_k,\mathbf{y}_k)\right] = \boldsymbol{\Sigma}_{kk}^{-1}\mathbb{E}_{q_k}[(\mathbf{x}_k-\boldsymbol{\mu}_k)(\mathbf{x}_k-\boldsymbol{\mu}_k)^T\phi_k(\mathbf{x}_k,\mathbf{y}_k)]\boldsymbol{\Sigma}_{kk}^{-1}-\boldsymbol{\Sigma}_{kk}^{-1}\mathbb{E}_{q_k}[\phi_k(\mathbf{x}_k,\mathbf{y}_k)]. \tag{3.29}$$

Therefore an alternative way to compute the three expectations in (3.27) without explicit computation of derivatives involves first computing

$$\underbrace{\mathbb{E}_{q_k}[\phi_k(\mathbf{x}_k,\mathbf{y}_k)]}_{\text{scalar}}, \quad \underbrace{\mathbb{E}_{q_k}\left[(\mathbf{x}_k-\boldsymbol{\mu}_k)\phi_k(\mathbf{x}_k,\mathbf{y}_k)\right]}_{\text{column}}, \quad \underbrace{\mathbb{E}_{q_k}\left[(\mathbf{x}_k-\boldsymbol{\mu}_k)(\mathbf{x}_k-\boldsymbol{\mu}_k)^T\phi_k(\mathbf{x}_k,\mathbf{y}_k)\right]}_{\text{matrix}}, \tag{3.30}$$

then computing (3.28) and (3.29) using the results of (3.30). Note how this reverse application of Stein's lemma has not affected the sparsity that we are exploiting for efficiency since we are applying it at the marginal level.

Now we discuss how we can approximate the three expectations given in (3.30) in an efficient, yet accurate way. As integrals, computing the expectations in (3.30) is generally not possible analytically, so we seek a numerical approximation. There are many ways of approximating the integrals of this form, the most popular type being multi-dimensional *Gaussian quadrature*, commonly referred to as Gaussian cubature or simply *cubature* (Cools, 1997)(Sarmavuori and Särkkä, 2012)(Kokkala et al., 2016)(Särkkä et al., 2016)(Särkkä, 2013). Using cubature, each of the expectations is approximated as

(Kokkala et al., 2016)(Särkkä et al., 2016)(Särkkä, 2013)

$$\mathbb{E}_{q_k}[\phi_k(\mathbf{x}_k, \mathbf{y}_k)] \approx \sum_{\ell=1}^{L} w_{k,\ell}\, \phi_k(\mathbf{x}_{k,\ell}, \mathbf{y}_k), \tag{3.31a}$$

$$\mathbb{E}_{q_k}[(\mathbf{x}_k - \boldsymbol{\mu}_k)\phi_k(\mathbf{x}_k, \mathbf{y}_k)] \approx \sum_{\ell=1}^{L} w_{k,\ell}\, (\mathbf{x}_{k,\ell} - \boldsymbol{\mu}_k)\phi_k(\mathbf{x}_{k,\ell}, \mathbf{y}_k), \tag{3.31b}$$

$$\mathbb{E}_{q_k}[(\mathbf{x}_k - \boldsymbol{\mu}_k)(\mathbf{x}_k - \boldsymbol{\mu}_k)^T\phi_k(\mathbf{x}_k, \mathbf{y}_k)] \approx \sum_{\ell=1}^{L} w_{k,\ell}\, (\mathbf{x}_{k,\ell} - \boldsymbol{\mu}_k)(\mathbf{x}_{k,\ell} - \boldsymbol{\mu}_k)^T\phi_k(\mathbf{x}_{k,\ell}, \mathbf{y}_k), \tag{3.31c}$$

where $w_{k,\ell}$ are weights, $\mathbf{x}_{k,\ell} = \boldsymbol{\mu}_k + \sqrt{\boldsymbol{\Sigma}_{kk}}\boldsymbol{\xi}_{k,\ell}$ are sigmapoints, and $\boldsymbol{\xi}_{k,\ell}$ are unit sigmapoints. Both the weights and unit sigmapoints are specific to the cubature method. Some popular examples are the unscented transform (Julier and Uhlmann, 1996)(Särkkä et al., 2016)(Särkkä, 2013), the spherical-cubature rule (Arasaratnam and Haykin, 2009)(Kokkala et al., 2016)(Särkkä, 2013), and Gauss-Hermite cubature (Ito and Xiong, 2000)(Wu et al., 2006)(Särkkä, 2013). Of note is how the nonlinearity of the integrand will affect how accurately each cubature method can approximate it. Särkkä (2013) states that, given an integrand composed of a linear combination of monomials of the form $x_1^{d_1}, x_2^{d_2}, \ldots, x_{N_k}^{d_{N_k}}$, the Mth order Gauss-Hermite cubature rule is exact when $d_i \leq 2M - 1$. In other words, more nonlinearity requires more computation effort (more sigmapoints). For example, an Mth-order Gauss-Hermite cubature approximation will require M^{N_k} sigmapoints, which could be infeasible in practice when N_k is large. Fortunately, the approximations given in (3.31) are at the factor level (i.e., at the level of $\mathbf{x}_k$, not $\mathbf{x}$), where N_k is a manageable size in most robotics problems. For this reason, we use Gauss-Hermite cubature in our numerical work presented in Sections 3.4, 3.5, and 3.6, resulting in accurate approximations of the expectations that are reasonably efficient.

3.2.6 Alternate Loss Functional

In this subsection, we present an alternative loss functional that will require less computational effort and offer more numerical stability over the main ESGVI approach. We consider the special case where the negative-log-likelihood takes the form

$$\phi(\mathbf{x}, \mathbf{y}) = \frac{1}{2}\mathbf{e}(\mathbf{x}, \mathbf{y})^T\mathbf{W}^{-1}\mathbf{e}(\mathbf{x}, \mathbf{y}). \tag{3.32}$$

Substituting this into the loss functional, we have

$$V(q) = \frac{1}{2}\mathbb{E}_q\left[\mathbf{e}(\mathbf{x},\mathbf{y})^T\mathbf{W}^{-1}\mathbf{e}(\mathbf{x},\mathbf{y})\right] + \frac{1}{2}\ln(|\boldsymbol{\Sigma}^{-1}|). \tag{3.33}$$

Owing to the convexity of the quadratic expression, $\mathbf{e}^T\mathbf{W}^{-1}\mathbf{e}$, we can apply *Jensen's inequality* (Jensen, 1906) directly to write

$$\mathbb{E}_q[\mathbf{e}(\mathbf{x},\mathbf{y})]^T\mathbf{W}^{-1}\mathbb{E}_q[\mathbf{e}(\mathbf{x},\mathbf{y})] \leq \mathbb{E}_q\left[\mathbf{e}(\mathbf{x},\mathbf{y})^T\mathbf{W}^{-1}\mathbf{e}(\mathbf{x},\mathbf{y})\right]. \tag{3.34}$$

The *Jensen gap* is the (positive) difference between the right and left sides of this inequality and tends to be larger the more nonlinear $\mathbf{e}(\mathbf{x},\mathbf{y})$ is and less concentrated $q(\mathbf{x})$ is. Motivated by this relationship, we can define a new loss functional as

$$V'(q) = \frac{1}{2}\mathbb{E}_q[\mathbf{e}(\mathbf{x},\mathbf{y})]^T\mathbf{W}^{-1}\mathbb{E}_q[\mathbf{e}(\mathbf{x},\mathbf{y})] + \frac{1}{2}\ln(|\boldsymbol{\Sigma}^{-1}|), \tag{3.35}$$

which can be thought of as a conservative approximation of $V(q)$ for mild nonlinearities and/or concentrated posteriors.

$V'(q)$ can be minimized by iteratively updating $q(\mathbf{x})$ while exploiting the problem sparsity from a factored likelihood. Combining Stein's lemma with (3.5a), (3.5b), and (3.5c), we have the useful identities

$$\frac{\partial}{\partial\boldsymbol{\mu}^T}\mathbb{E}_q[f(\mathbf{x})] \equiv \mathbb{E}_q\left[\frac{\partial f(\mathbf{x})}{\partial\mathbf{x}^T}\right], \tag{3.36a}$$

$$\frac{\partial^2}{\partial\boldsymbol{\mu}^T\partial\boldsymbol{\mu}}\mathbb{E}_q[f(\mathbf{x})] \equiv \mathbb{E}_q\left[\frac{\partial^2 f(\mathbf{x})}{\partial\mathbf{x}^T\partial\mathbf{x}}\right] \equiv -2\boldsymbol{\Sigma}^{-1}\left(\frac{\partial}{\partial\boldsymbol{\Sigma}^{-1}}\mathbb{E}_q[f(\mathbf{x})]\right)\boldsymbol{\Sigma}^{-1}. \tag{3.36b}$$

We can then directly approximate the expected error as

$$\begin{aligned}\mathbb{E}_{q^{(i+1)}}[\mathbf{e}(\mathbf{x},\mathbf{y})] &\approx \mathbb{E}_{q^{(i)}}[\mathbf{e}(\mathbf{x},\mathbf{y})] + \frac{\partial}{\partial\boldsymbol{\mu}}\mathbb{E}_{q^{(i)}}[\mathbf{e}(\mathbf{x},\mathbf{y})]\underbrace{\left(\boldsymbol{\mu}^{(i+1)} - \boldsymbol{\mu}^{(i)}\right)}_{\delta\boldsymbol{\mu}} \\ &= \underbrace{\mathbb{E}_{q^{(i)}}[\mathbf{e}(\mathbf{x},\mathbf{y})]}_{\bar{\mathbf{e}}^{(i)}} + \underbrace{\mathbb{E}_{q^{(i)}}\left[\frac{\partial}{\partial\mathbf{x}}\mathbf{e}(\mathbf{x},\mathbf{y})\right]}_{\bar{\mathbf{E}}^{(i)}}\delta\boldsymbol{\mu} = \bar{\mathbf{e}}^{(i)} + \bar{\mathbf{E}}^{(i)}\,\delta\boldsymbol{\mu},\end{aligned} \tag{3.37}$$

where we have employed (3.36a).

We can then approximate the loss functional as

$$V'(q) \approx \frac{1}{2}\left(\bar{\mathbf{e}}^{(i)} + \bar{\mathbf{E}}^{(i)}\,\delta\boldsymbol{\mu}\right)^T\mathbf{W}^{-1}\left(\bar{\mathbf{e}}^{(i)} + \bar{\mathbf{E}}^{(i)}\,\delta\boldsymbol{\mu}\right) + \frac{1}{2}\ln(|\boldsymbol{\Sigma}^{-1}|), \tag{3.38}$$

which is now exactly quadratic in $\delta\boldsymbol{\mu}$. Since we have implicitly approximated the Hessian, our approximations lead to a Gauss-Newton estimator. Taking the first and second derivatives with respect to $\delta\boldsymbol{\mu}$, we have

$$\frac{\partial V'(q)}{\partial\,\delta\boldsymbol{\mu}^T} = \bar{\mathbf{E}}^{(i)^T}\mathbf{W}^{-1}\left(\bar{\mathbf{e}}^{(i)} + \bar{\mathbf{E}}^{(i)}\,\delta\boldsymbol{\mu}\right), \tag{3.39}$$

$$\frac{\partial^2 V'(q)}{\partial\,\delta\boldsymbol{\mu}^T\partial\,\delta\boldsymbol{\mu}} = \bar{\mathbf{E}}^{(i)^T}\mathbf{W}^{-1}\bar{\mathbf{E}}^{(i)}. \tag{3.40}$$

For the derivative with respect to $\boldsymbol{\Sigma}^{-1}$, we have

$$\frac{\partial V'(q)}{\partial \boldsymbol{\Sigma}^{-1}} \approx -\frac{1}{2}\boldsymbol{\Sigma}\,\bar{\mathbf{E}}^{(i)^T}\mathbf{W}^{-1}\bar{\mathbf{E}}^{(i)}\,\boldsymbol{\Sigma} + \frac{1}{2}\boldsymbol{\Sigma}, \tag{3.41}$$

where the approximation enforces the relationship in (3.36b), which does not hold exactly anymore due to the altered nature of $V'(q)$. Setting this to zero for a critical point we have

$$(\boldsymbol{\Sigma}^{-1})^{(i+1)} = \bar{\mathbf{E}}^{(i)^T}\mathbf{W}^{-1}\bar{\mathbf{E}}^{(i)}, \tag{3.42}$$

where we have created an iterative update analogous to that in the main ESGVI approach.

For the mean, we set (3.39) to zero and then for the optimal update we have

$$\underbrace{\bar{\mathbf{E}}^{(i)^T}\mathbf{W}^{-1}\bar{\mathbf{E}}^{(i)}}_{(\boldsymbol{\Sigma}^{-1})^{(i+1)}}\,\delta\boldsymbol{\mu} = -\bar{\mathbf{E}}^{(i)^T}\mathbf{W}^{-1}\bar{\mathbf{e}}^{(i)}. \tag{3.43}$$

Solving for $\delta\boldsymbol{\mu}$ provides a Gauss-Newton update, which we will refer to as ESGVI Gauss-Newton (ESGVI-GN). This is identical to how Gauss-Newton is normally carried out, but now we calculate $\bar{\mathbf{e}}$ and $\bar{\mathbf{E}}$ not just at a single point but rather as an expectation over our Gaussian posterior estimate.

We make a number of remarks about the approach:

1. The sparsity of the inverse covariance matrix, $\boldsymbol{\Sigma}^{-1}$, will be identical to the full ESGVI approach. We can demonstrate this by showing

$$\phi(\mathbf{x},\mathbf{y}) = \sum_{k=1}^{K}\phi_k(\mathbf{x}_k,\mathbf{y}_k) = \frac{1}{2}\sum_{k=1}^{K}\mathbf{e}_k(\mathbf{x}_k,\mathbf{y}_k)^T\mathbf{W}_k^{-1}\mathbf{e}_k(\mathbf{x}_k,\mathbf{y}_k) = \frac{1}{2}\mathbf{e}(\mathbf{x},\mathbf{y})^T\mathbf{W}^{-1}\mathbf{e}(\mathbf{x},\mathbf{y}), \tag{3.44}$$

where

$$\mathbf{e}(\mathbf{x},\mathbf{y}) = \begin{bmatrix}\mathbf{e}_1(\mathbf{x}_1,\mathbf{y}_1)\\ \vdots \\ \mathbf{e}_K(\mathbf{x}_K,\mathbf{y}_K)\end{bmatrix}, \quad \mathbf{W} = \operatorname{diag}(\mathbf{W}_1,\ldots,\mathbf{W}_K). \tag{3.45}$$

Then we have

$$
\begin{aligned}
\boldsymbol{\Sigma}^{-1} &= \mathbb{E}_q\left[\frac{\partial}{\partial \mathbf{x}}\mathbf{e}(\mathbf{x},\mathbf{y})\right]^T \mathbf{W}^{-1}\mathbb{E}_q\left[\frac{\partial}{\partial \mathbf{x}}\mathbf{e}(\mathbf{x},\mathbf{y})\right] \\
&= \sum_{k=1}^{K} \mathbf{P}_k^T \mathbb{E}_{q_k}\left[\frac{\partial}{\partial \mathbf{x}_k}\mathbf{e}_k(\mathbf{x}_k,\mathbf{y}_k)\right]^T \mathbf{W}_k^{-1}\mathbb{E}_{q_k}\left[\frac{\partial}{\partial \mathbf{x}_k}\mathbf{e}_k(\mathbf{x}_k,\mathbf{y}_k)\right]\mathbf{P}_k, \quad (3.46)
\end{aligned}
$$

which will have zeros wherever an error term does not depend on the variables. We also see that the expectations can be reduced to being over the marginal, $q_k(\mathbf{x}_k)$, meaning we still only require the blocks of $\boldsymbol{\Sigma}$ corresponding to the non-zero blocks of $\boldsymbol{\Sigma}^{-1}$.

2. We can still use Stein's lemma to avoid the need to compute any derivatives:

$$
\mathbb{E}_{q_k}\left[\frac{\partial}{\partial \mathbf{x}_k}\mathbf{e}_k(\mathbf{x}_k,\mathbf{y}_k)\right] = \mathbb{E}_{q_k}\left[\mathbf{e}_k(\mathbf{x}_k,\mathbf{y}_k)(\mathbf{x}_k-\boldsymbol{\mu}_k)^T\right]\boldsymbol{\Sigma}_{kk}^{-1}. \quad (3.47)
$$

This is sometimes referred to as a statistical Jacobian and this usage is very similar to the filtering and smoothing approaches described by Särkkä (2013), because cubature can be applied at the measurement model level rather than the factor level. Since we are iteratively recomputing the statistical Jacobian about our posterior estimate, this is most similar to Sibley et al. (2006) and García-Fernández et al. (2015), although some details are different, such as the fact that we started from our loss functional, $V'(q)$.

3. The number of cubature points required to calculate $\mathbb{E}_{q_k}\left[\mathbf{e}_k(\mathbf{x}_k,\mathbf{y}_k)(\mathbf{x}_k-\boldsymbol{\mu}_k)^T\right]$ will be lower than our full ESGVI approach since the order of the expression in the integrand is half that of $\mathbb{E}_{q_k}\left[(\mathbf{x}_k-\boldsymbol{\mu}_k)(\mathbf{x}_k-\boldsymbol{\mu}_k)^T\phi_k(\mathbf{x}_k,\mathbf{y}_k)\right]$. Since the number of cubature points goes up as M^{N_k}, cutting M in half is significant.

4. It is known that minimizing $\mathrm{KL}(q||p)$, the objective of $V(q)$, can result in a Gaussian that is too confident (i.e., inverse covariance is too large) (Bishop, 2006; Ala-Luhtala et al., 2015). From Jensen's inequality, we can see that $V'(q)$ will result in a more

conservative inverse covariance. For an arbitrary non-zero vector, $\mathbf{a}$, we have

$$\begin{aligned} 0 \;<\; & \underbrace{\mathbf{a}^T \mathbb{E}_q\left[\frac{\partial \mathbf{e}(\mathbf{x},\mathbf{y})}{\partial \mathbf{x}}\right]^T \mathbf{W}^{-1} \mathbb{E}_q\left[\frac{\partial \mathbf{e}(\mathbf{x},\mathbf{y})}{\partial \mathbf{x}}\right] \mathbf{a}}_{\boldsymbol{\Sigma}^{-1} \text{ from } V'(q)} \\ \overset{\text{Jensen}}{\leq} \; & \mathbf{a}^T \mathbb{E}_q\left[\frac{\partial \mathbf{e}(\mathbf{x},\mathbf{y})}{\partial \mathbf{x}}^T \mathbf{W}^{-1} \frac{\partial \mathbf{e}(\mathbf{x},\mathbf{y})}{\partial \mathbf{x}}\right] \mathbf{a} \quad \overset{\text{Gauss-Newton}}{\approx} \quad \mathbf{a}^T \underbrace{\mathbb{E}_q\left[\frac{\partial^2 \phi(\mathbf{x},\mathbf{y})}{\partial \mathbf{x}^T \partial \mathbf{x}}\right]}_{\boldsymbol{\Sigma}^{-1} \text{ from } V(q)} \mathbf{a}, \end{aligned} \tag{3.48}$$

which ensures we have a positive definite inverse covariance that is conservative compared to the full ESGVI approach.

Due to the extra approximations made in ESGVI-GN compared to ESGVI, it remains to be seen whether it improves over MAP approaches. However, as ESGVI-GN provides a batch option that does not require any derivatives, it can be used as a less expensive preprocessor for the derivative-free version of full ESGVI that will always be positive definite (for numerical stability).

3.3 Parameter Estimation

We use this section to provide a sketch of how parameters can be learned using our ESGVI framework and EM. This parameter learning framework is critical to the rest of this thesis, as the remaining chapters will focus on applications of this idea.

We begin with a maximum-likelihood objective for the observed data, $\mathbf{y}$,

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} p(\mathbf{y}|\boldsymbol{\theta}), \tag{3.49}$$

where $\boldsymbol{\theta}$ represents the parameters we want to learn. Switching to the negative log-likelihood, introducing the latent state, $\mathbf{x}$, and applying the usual EM decomposition (Neal and Hinton, 1998; Ghahramani and Roweis, 1999; Bishop, 2006) results in

$$\mathscr{L} = -\ln p(\mathbf{y}|\boldsymbol{\theta}) = \underbrace{\int_{-\infty}^{\infty} q(\mathbf{x}) \ln\left(\frac{p(\mathbf{x}|\mathbf{y},\boldsymbol{\theta})}{q(\mathbf{x})}\right) d\mathbf{x}}_{\leq 0} \underbrace{- \int_{-\infty}^{\infty} q(\mathbf{x}) \ln\left(\frac{p(\mathbf{x},\mathbf{y}|\boldsymbol{\theta})}{q(\mathbf{x})}\right) d\mathbf{x}}_{\text{upper bound}},$$

where, as in the earlier sections, $q(\mathbf{x})$ is our Gaussian posterior estimate.

The first term is a KL divergence between the approximate posterior and the true posterior, $p(\mathbf{x}|\mathbf{y})$, which is unknown. However, we can work with the second, upper bound term that is the (negative) ELBO. We already discussed in Section 3.2.1 that the ELBO is equivalent to our ESGVI loss:

$$V(q|\boldsymbol{\theta}) = \mathbb{E}_q[\phi(\mathbf{x}, \mathbf{y}|\boldsymbol{\theta})] + \frac{1}{2}\ln(|\boldsymbol{\Sigma}^{-1}|), \tag{3.50}$$

which we now write with our unknown parameters, $\boldsymbol{\theta}$. We now can proceed iteratively in two steps: the expectation step (e-step) and the maximization step (m-step). Note that we are working with the negative log-likelihood so we are technically applying a minimization, but the acronym stays the same.

The e-step, is already accomplished by ESGVI. We simply hold $\boldsymbol{\theta}$ fixed and run the inference to convergence to solve for $q(\mathbf{x})$, our Gaussian approximation to the posterior. In the m-step, we hold $q(\mathbf{x})$ fixed and find the value of $\boldsymbol{\theta}$ that minimizes the loss functional. By alternating between the e- and m-steps, we can (locally) solve for the best value of the parameters to minimize the original data likelihood objective, $\mathscr{L}$.

As we have done in the main part of the chapter, we assume the joint likelihood of the state and measurements (given the parameters) factors so that

$$\phi(\mathbf{x}, \mathbf{y}|\boldsymbol{\theta}) = \sum_{k=1}^{K} \phi_k(\mathbf{x}_k, \mathbf{y}_k|\boldsymbol{\theta}), \tag{3.51}$$

where for generality we have each factor being affected by the entire parameter set, $\boldsymbol{\theta}$, but in practice it could be a subset. Taking the derivative of the loss functional with respect to $\boldsymbol{\theta}$, we have

$$\frac{\partial V(q|\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{\partial}{\partial \boldsymbol{\theta}}\mathbb{E}_q[\phi(\mathbf{x}, \mathbf{y}|\boldsymbol{\theta})] = \frac{\partial}{\partial \boldsymbol{\theta}}\mathbb{E}_q\left[\sum_{k=1}^{K} \phi_k(\mathbf{x}_k, \mathbf{y}_k|\boldsymbol{\theta})\right] = \sum_{k=1}^{K} \mathbb{E}_{q_k}\left[\frac{\partial}{\partial \boldsymbol{\theta}}\phi_k(\mathbf{x}_k, \mathbf{y}_k|\boldsymbol{\theta})\right], \tag{3.52}$$

where in the rightmost expression the expectation simplifies to being over the marginal, $q_k(\mathbf{x}_k)$, rather than the full Gaussian, $q(\mathbf{x})$. As with the main ESGVI approach, this means that we only need the blocks of the covariance, $\boldsymbol{\Sigma}$, corresponding to the non-zero blocks of $\boldsymbol{\Sigma}^{-1}$, which we are already calculating as part of the e-step. Furthermore, we can easily evaluate the marginal expectations using sigmapoints.

To make this more tangible, consider the example of

$$\phi(\mathbf{x}, \mathbf{y}|\mathbf{W}) = \frac{1}{2}\sum_{k=1}^{K}\Big(\mathbf{e}_k(\mathbf{x}_k, \mathbf{y}_k)^T \mathbf{W}\, \mathbf{e}_k(\mathbf{x}_k, \mathbf{y}_k) - \ln(|\mathbf{W}|)\Big), \tag{3.53}$$

where the unknown parameter is $\mathbf{W}$, the inverse measurement covariance matrix. Then taking the derivative with respect to $\mathbf{W}$ we have

$$\frac{\partial V(q|\mathbf{W})}{\partial \mathbf{W}} = \frac{1}{2}\sum_{k=1}^{K} \mathbb{E}_{q_k}\left[\mathbf{e}_k(\mathbf{x}_k, \mathbf{y}_k)\mathbf{e}_k(\mathbf{x}_k, \mathbf{y}_k)^T\right] - \frac{K}{2}\mathbf{W}^{-1}. \tag{3.54}$$

Setting this to zero for a minimum we have

$$\mathbf{W}^{-1} = \frac{1}{K}\sum_{k=1}^{K} \mathbb{E}_{q_k}\left[\mathbf{e}_k(\mathbf{x}_k, \mathbf{y}_k)\mathbf{e}_k(\mathbf{x}_k, \mathbf{y}_k)^T\right], \tag{3.55}$$

where we can use sigmapoints to evaluate the marginal expectations. Reiterating, we never require the full covariance matrix, $\mathbf{\Sigma}$, implying that our exactly sparse framework extends to parameter estimation.

3.4 Experiment 1: Stereo One-Dimensional Simulation

Our first experiment is a simulation of a simple one-dimensional, nonlinear estimation problem motivated by the type of inverse-distance nonlinearity in a stereo camera model. Since this problem is one-dimensional, our aim in this experiment is to show that our proposed iterative scheme converges to the minimum of our cost function and that we offer an improvement over the usual MAP approach. We will demonstrate the ability to exploit sparsity in the next two experiments.

(Barfoot, 2024) uses this same experiment, with the same parameter settings, as an example to demonstrate how the point estimate of MAP approaches the mode of the true (non-Gaussian) posterior. We assume our true state is drawn from a Gaussian prior:

$$x \sim \mathcal{N}(\mu_p, \sigma_p^2). \tag{3.56}$$

We then generate a measurement according to

$$y = \frac{fb}{x} + n, \qquad n \sim \mathcal{N}(0, \sigma_r^2), \tag{3.57}$$

where n is measurement noise. The numerical values used in our trials were

$$\mu_p = 20 \text{ [m]}, \quad \sigma_p^2 = 9 \text{ [m}^2\text{]}, f = 400 \text{ [pixel]}, \quad b = 0.1 \text{ [m]}, \quad \sigma_r^2 = 0.09 \text{ [pixel}^2\text{]}.$$

The two factors are defined as

$$\phi = \frac{1}{2}\frac{(x-\mu_p)^2}{\sigma_p^2}, \qquad \psi = \frac{1}{2}\frac{\left(y-\frac{fb}{x}\right)^2}{\sigma_r^2}, \tag{3.58}$$

so that $-\ln p(\mathbf{x}, \mathbf{y}) = \phi + \psi + \text{constant}$. Our loss functional is

$$V(q) = \mathbb{E}_q[\phi] + \mathbb{E}_q[\psi] + \frac{1}{2}\ln(\sigma^{-2}), \tag{3.59}$$

where $q = \mathcal{N}(\mu, \sigma^2)$ is our Gaussian estimate of the posterior.

We ran $100,000$ trials of a proper Bayesian experiment, where each trial consisted of drawing the latent state from the prior, then producing a noisy measurement given that state. We avoided edge cases (e.g., negative distance) by only accepting a draw of the latent state if it was within 4 standard deviations of the mean, resulting in 6 out of the $100,000$ experiments to not be accepted and the state redrawn. We then ran several versions of our algorithm summarized in Table 3.2. Everything else to do with the experiment was the same for all algorithms, allowing a fair comparison. Figure 3.2 shows the statistical results of our $100,000$ trials as boxplots. The columns correspond to the different versions of our algorithm, while the rows are different performance metrics. The first column (analytical Hessian and Jacobian with a single quadrature point at the mean) is equivalent to a standard MAP approach. We can see that our new algorithms do require a few more iterations (first row) to converge than MAP, i.e., it takes more computation to arrive at a better approximation to the posterior. We can also see that the new algorithms do find a lower final value of the loss functional, $V(q)$, which is what we asked them to minimize (second row).

We also wanted to see if the new algorithms were less biased and more consistent than MAP, and so we calculated the average error (third row), average squared error (fourth row), and squared Mahalanobis / Normalized Estimation Squared Error (NEES) (fifth row). To be fair, we did not ask the estimator to minimize these quantities, but our hypothesis has been that we should also do better on these metrics by minimizing $V(q)$. Looking at the third row, all the GVI variants are less biased than MAP by two orders of magnitude or more. Our MAP error of -30.6 cm is consistent with the result reported by Barfoot (2024). The best algorithm reported there, the Iterated Sigmapoint Kalman Filter (ISPKF) (Sibley et al., 2006), had a bias of -3.84 cm. Here, our best algorithm has a bias of 0.3 cm. Squared error (fourth row) is also slightly improved compared to MAP.

The average squared Mahalanobis / NEES error should be close to 1 for a one-

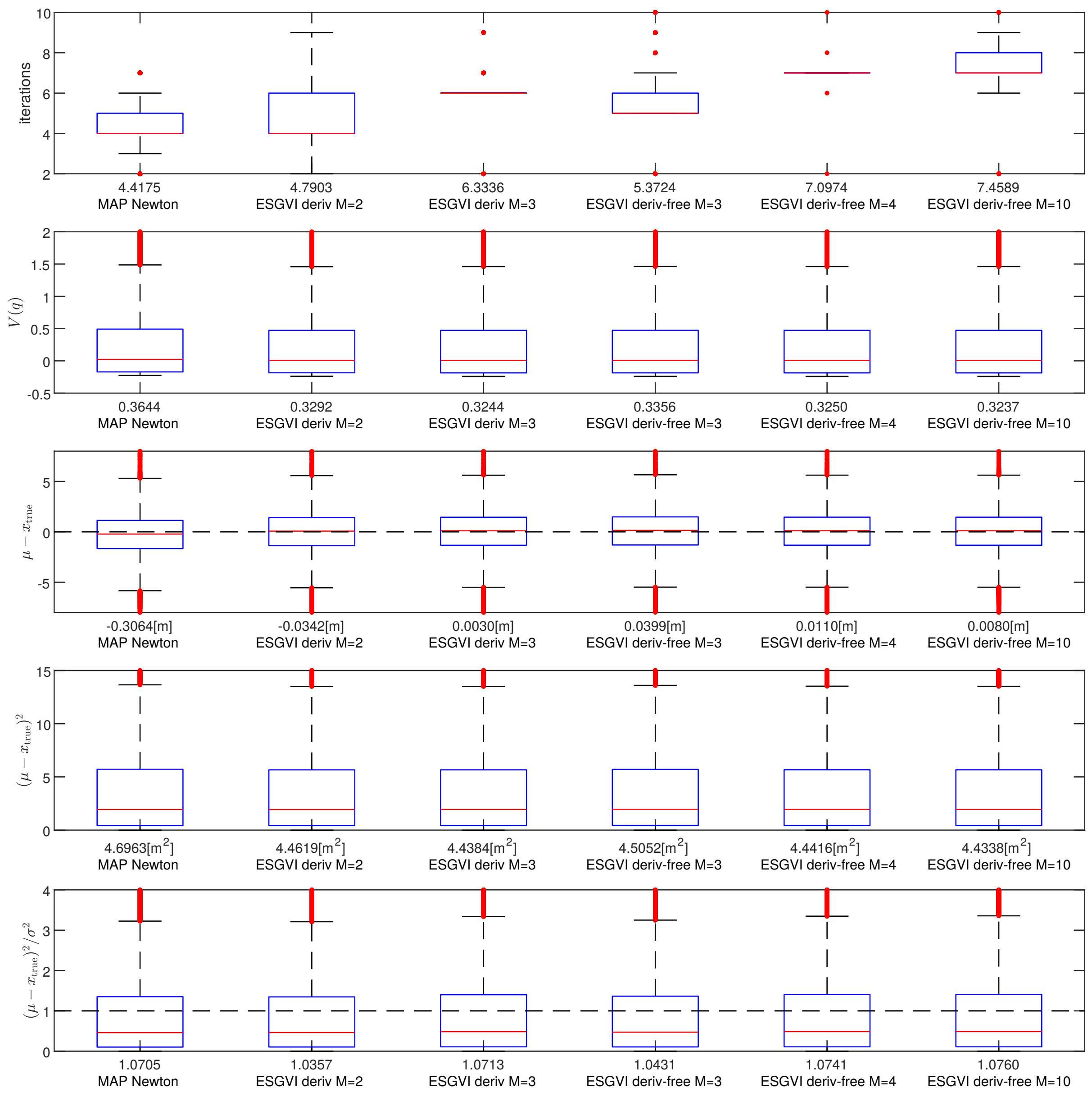


Figure 3.2: (Experiment 1) Statistical results of 100,000 trials of the one-dimensional stereo camera simulation shown as standard boxplots. The different rows show different performance metrics for the different variants of our algorithm (columns). Table 3.2 provides details of the different algorithms tested. The number below an algorithm label is its mean performance on that metric. Further details are discussed in the text.

dimensional problem. Our results here are mixed, with some of our approaches doing better than MAP and some not. It seems that our choice of KL$(q||p)$ rather than KL$(p||q)$ results in a slightly overconfident covariance. Bishop (2006) and Ala-Luhtala et al. (2015) show similar situations for the same choice of KL$(q||p)$. It may be possible to overcome this by changing the relative weighting between the two main terms in $V(q)$ through the

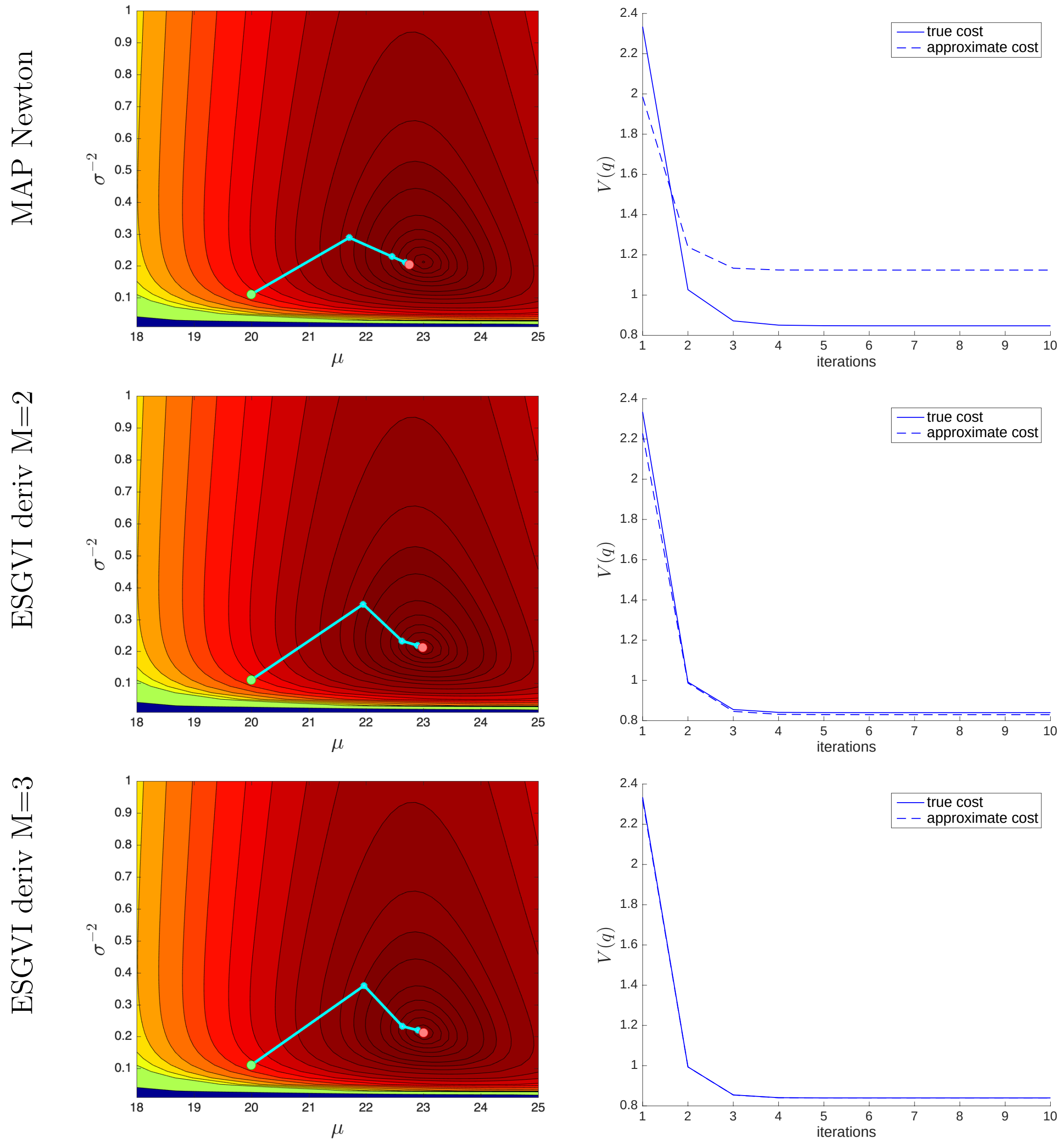


Figure 3.3: (Experiment 1) One trial of the one-dimensional simulation showing the convergence history for three different algorithms in each row. The left column shows a contour map of the loss functional, $V(q)$, with the steps the optimizer took starting from the prior (green dot) to its converged value (red dot). The right column shows the loss at each iteration. Since each algorithm makes different approximations to the loss, we show the loss that each algorithm used to make decisions and the true loss at each step.

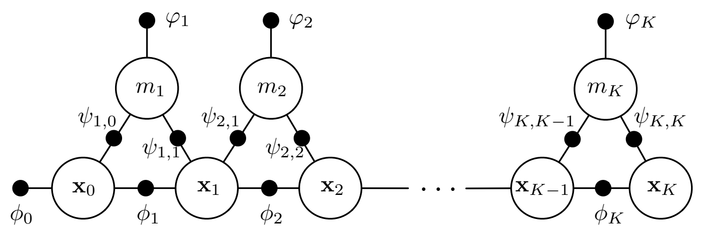


Figure 3.4: (Experiment 2) Factor graph for the stereo K-dimensional simulation. White circles represent variables to be estimated, which are robot positions and landmark positions. Small black dots represent factors in the joint likelihood of the state and data.

use of a hyperparameter that is tuned for a particular situation.

Figure 3.3 shows the convergence of a single trial of the $100,000$ that we ran. Only a subset of the algorithms (rows) are shown in the interest of space. The left column is a contour plot of $V(q)$ and the path the optimizer took to arrive at its minimum (red dot), starting from the prior (green dot). The right column shows a plot of $V(q)$ at each iteration. We show the true value of the loss and the approximation of the loss that the algorithm had access to during its iterations (each algorithm used a different number of quadrature points, M). Note that the true value of the loss is calculated with a larger number of quadrature points. We see that the MAP approach clearly does not converge at the minimum of $V(q)$ due to its approximation of the required expectations. The other algorithms converge close to the true minimum in a similar number of iterations.

3.5 Experiment 2: Stereo K-Dimensional Simulation

In this simulated experiment, we introduce time and allow our simulated robot to move along the x-axis, using the same nonlinear stereo camera model as the previous experiment. Our aim is to show that we can exploit the sparse structure of the problem in higher dimensions, while still benefiting from our variational approach. We apply a prior both on the robot motion and on the landmark positions in this SLAM problem. While a prior on the landmark positions is not common in practice, we do so in order to conduct a proper Bayesian comparison of the algorithms. The *factor graph* for the problem is shown in Figure 3.4.

Our state is

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_0 \\ \vdots \\ \mathbf{x}_K \\ m_1 \\ \vdots \\ m_K \end{bmatrix}, \quad \mathbf{x}_k = \begin{bmatrix} p_k \\ v_k \end{bmatrix}, \tag{3.60}$$

where p_k is a robot position, v_k a robot speed, and m_k a landmark position. The problem is highly structured as each landmark is seen exactly twice from two consecutive robot positions.

For the (linear) prior factors, we have

$$\phi_k = \begin{cases} \frac{1}{2}(\mathbf{x}_0 - \check{\mathbf{x}}_0)^T \check{\mathbf{P}}^{-1}(\mathbf{x}_0 - \check{\mathbf{x}}_0) & k = 0 \\ \frac{1}{2}(\mathbf{x}_k - \mathbf{A}\mathbf{x}_{k-1})^T \mathbf{Q}^{-1}(\mathbf{x}_k - \mathbf{A}\mathbf{x}_{k-1}) & k > 0 \end{cases}, \tag{3.61a}$$

$$\varphi_k = \frac{1}{2}\frac{(m_k - \mu_{m,k})^2}{\sigma_m^2}, \tag{3.61b}$$

with

$$\check{\mathbf{P}} = \mathrm{diag}(\sigma_p^2, \sigma_v^2), \quad \mathbf{A} = \begin{bmatrix} 1 & T \\ 0 & 1 \end{bmatrix}, \quad \mathbf{Q} = \begin{bmatrix} \frac{1}{3}T^3 Q_C & \frac{1}{2}T^2 Q_C \\ \frac{1}{2}T^2 Q_C & T Q_C \end{bmatrix}, \tag{3.62}$$

where T is the discrete-time sampling period, Q_C is a power spectral density, and σ_p^2, σ_v^2, σ_m^2 are variances. The robot state prior encourages constant velocity (Barfoot, 2024), and the landmark prior is a Gaussian centered at the true landmark location, $\mu_{m,k}$.

For the (nonlinear) measurement factors, we have

$$\psi_{\ell,k} = \frac{1}{2}\frac{\left(y_{\ell,k} - \frac{fb}{m_\ell - p_k}\right)^2}{\sigma_r^2}, \tag{3.63}$$

where f and b are the same camera parameters of the previous experiment, $y_{\ell,k}$ is the disparity measurement of the ℓth landmark from the kth position, and σ_r^2 is the measurement noise variance.

The negative log-likelihood of the state and data is then

$$-\ln p(\mathbf{x}, \mathbf{y}) = \sum_{k=0}^{K} \phi_k + \sum_{k=1}^{K} \varphi_k + \sum_{k=1}^{K} (\psi_{k,k-1} + \psi_{k,k}) + \text{constant}. \tag{3.64}$$

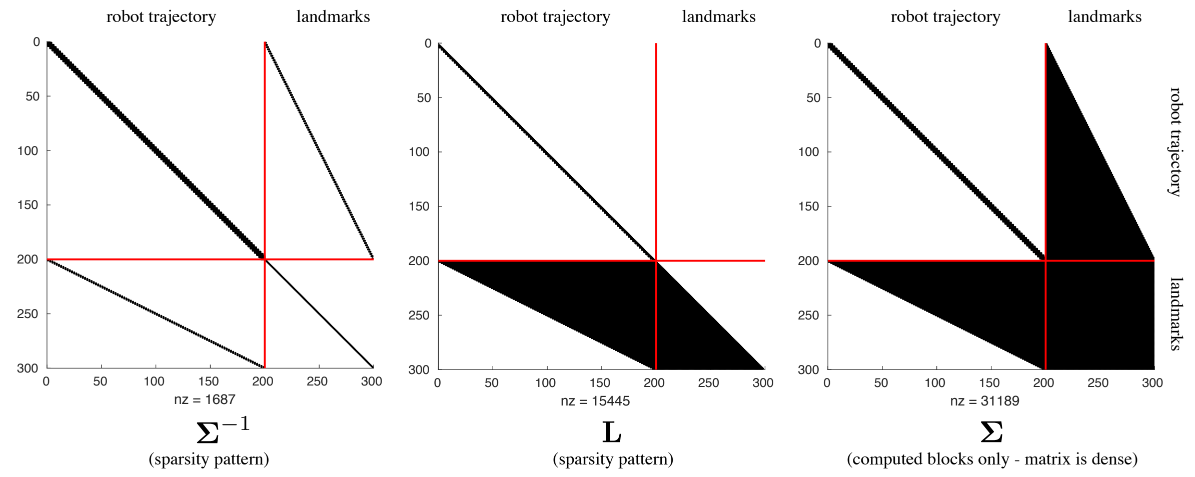


Figure 3.5: (Experiment 2) Sparsity patterns for the stereo K-dimensional simulation experiment. The red partition lines separate the robot state variables from the landmark variables. The inverse covariance, $\boldsymbol{\Sigma}^{-1}$, is highly sparse due to the factor graph pattern in Figure 3.4, where only 1,687/89,401 = 1.9% of entries are nonzero. After performing a lower-diagonal-upper decomposition, the lower factor, $\mathbf{L}$, becomes more filled due to the 'four corners of a box' rule, where 15,445/89,401 = 17.3% of entries are non-zero. Finally, we see that only a fraction of the entries of $\boldsymbol{\Sigma}$ are required even though this matrix is actually dense. Since $\boldsymbol{\Sigma}$ is symmetric, we only need to calculate 17.3% of it as well.

We set the maximum number of timesteps to be $K = 99$, resulting in a total state dimension of 299. Figure 3.5 shows the sparsity patterns of $\boldsymbol{\Sigma}^{-1}$, $\mathbf{L}$, and the blocks of $\boldsymbol{\Sigma}$ that get computed by the method of Takahashi et al. (1973).

We ran $10,000$ trials of this simulation, where for each trial, we drew the latent robot trajectory and landmark states from the Bayesian prior, then simulated the noisy nonlinear measurements. We estimated the full state using four different algorithms from Table 3.2: 'MAP Newton', 'ESGVI deriv M=2', 'ESGVI deriv M=3', and 'ESGVI deriv-free M=4'. Figure 3.6 shows the statistical results of the $10,000$ trials.

The results show that all the algorithms converge well in a small number of iterations (usually 4). Increasing the number of cubature points for the derivative-based methods does result in reducing the overall value of the loss functional, $V(q)$. The 'ESGVI deriv-free M=4' method performs on par with 'ESGVI deriv M=3' method, but requires no analytical derivatives of the factors.

Similar to the one-dimensional simulation, the bias in the estimate (see middle row of Figure 3.6) is reduced in the GVI approaches compared to the MAP approach. This result is important since it can be achieved in a tractable way for large-scale problems and even without analytical derivatives. The GVI methods also do slightly better than MAP on the mean-squared error (fourth row of Figure 3.6) as well as squared Mahalanobis

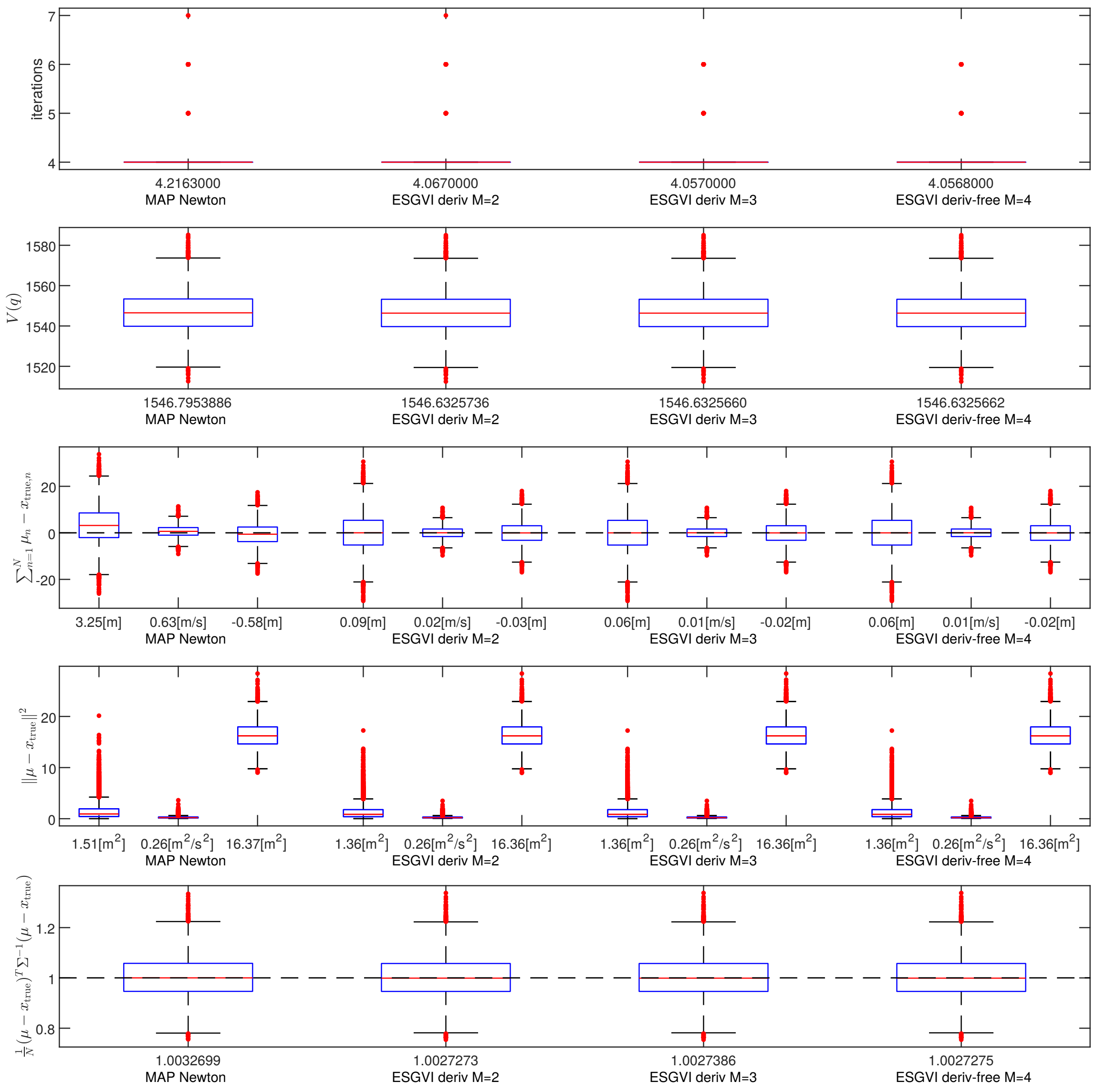


Figure 3.6: (Experiment 2) Statistical results of 10, 000 trials of the K-dimensional stereo camera simulation shown as standard boxplots. The different rows show different performance metrics for the different variants of our algorithm (columns). Table 3.2 provides details of the different algorithms tested. The number below an algorithm label is its mean performance on that metric. Further details are discussed in the text.

distance / NEES (fifth row of Figure 3.6), but the improvements are smaller.

3.6 Experiment 3: Robot Dataset

Here, we consider a batch SLAM problem with a robot driving around and building a map of landmarks using a laser rangefinder. This dataset has previously been used by

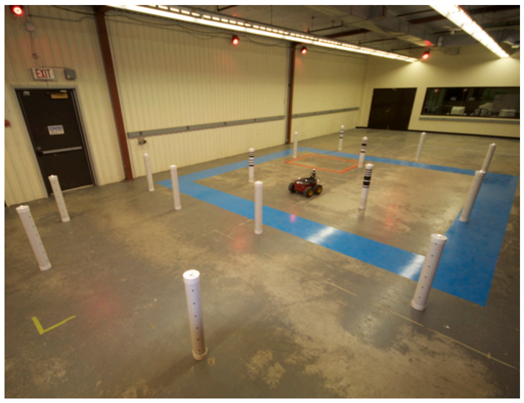
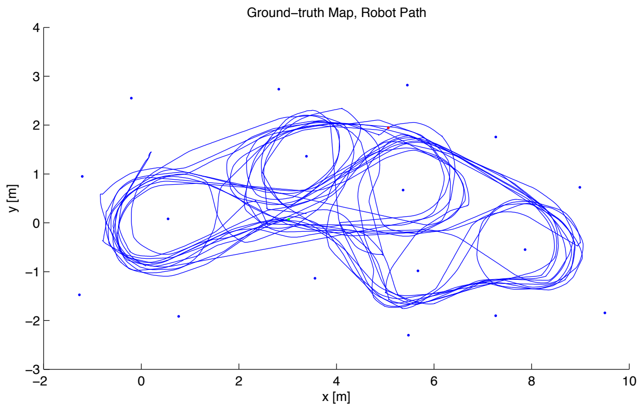



Figure 3.7: (Experiment 3) Setup: (left) a mobile robot navigates a map of landmarks, receiving bearing measurements to some landmarks and wheel odometry. (right) the ground-truth path of the robot and landmark map as measured by the Vicon system.

Barfoot et al. (2014) to test SLAM algorithms. Figure 3.7 shows the experiment setup. The groundtruth for both the robot trajectory and landmark positions is provided by a Vicon motion capture system. The whole dataset is 12, 000 timesteps long, which we divide into six subsequences of 2, 000 timestamps. We report statistical performance as an average over these six subsequences. We assume that the data association (i.e., the correspondence between measurements and landmarks) is known in this experiment to focus on testing the state estimation part of the problem.

Our state is

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_0 \\ \vdots \\ \mathbf{x}_K \\ \mathbf{m}_1 \\ \vdots \\ \mathbf{m}_L \end{bmatrix}, \quad \mathbf{x}_k = \begin{bmatrix} x_k \\ y_k \\ \theta_k \\ \dot{x}_k \\ \dot{y}_k \\ \dot{\theta}_k \end{bmatrix}, \quad \mathbf{m}_\ell = \begin{bmatrix} x_\ell \\ y_\ell \end{bmatrix}, \tag{3.65}$$

where $\mathbf{x}_k$ is a robot state and $\mathbf{m}_\ell$ a landmark position. For each of our six subsequences we have $K = 2,000$ and $L = 17$.

Figure 3.8 shows the factor graph for this experiment and Figure 3.9 shows the corresponding sparsity patterns. The (linear) prior on the robot states is

$$\phi_k = \begin{cases} \frac{1}{2}(\mathbf{x}_0 - \check{\mathbf{x}}_0)^T \check{\mathbf{P}}^{-1}(\mathbf{x}_0 - \check{\mathbf{x}}_0) & k = 0 \\ \frac{1}{2}(\mathbf{x}_k - \mathbf{A}\mathbf{x}_{k-1})^T \mathbf{Q}^{-1}(\mathbf{x}_k - \mathbf{A}\mathbf{x}_{k-1}) & k > 0 \end{cases}, \tag{3.66}$$

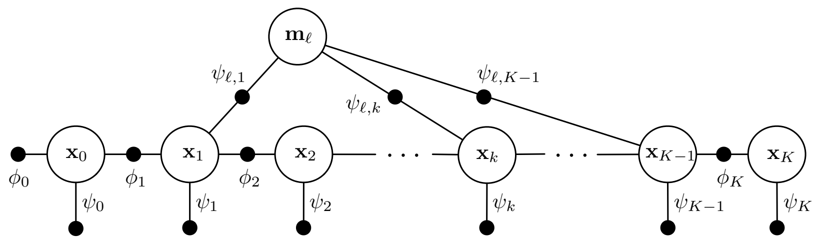


Figure 3.8: (Experiment 3) Factor graph for our robot dataset. White circles indicate state variables and small black circles indicate factors.

with

$$\begin{aligned}\check{\mathbf{P}} =& \text{diag}(\sigma_x^2, \sigma_y^2, \sigma_\theta^2, \sigma_{\dot{x}}^2, \sigma_{\dot{y}}^2, \sigma_{\dot{\theta}}^2), \quad \mathbf{A} = \begin{bmatrix} \mathbf{1} & T\mathbf{1} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}, \quad \mathbf{Q} = \begin{bmatrix} \frac{1}{3}T^3\mathbf{Q}_C & \frac{1}{2}T^2\mathbf{Q}_C \\ \frac{1}{2}T^2\mathbf{Q}_C & T\mathbf{Q}_C \end{bmatrix}, \\ \mathbf{Q}_C =& \text{diag}(Q_{C,1}, Q_{C,2}, Q_{C,3}),\end{aligned} \tag{3.67}$$

where T is the discrete-time sampling period, $Q_{C,i}$ are power spectral densities, and σ_x^2, σ_y^2,σ_θ^2, $\sigma_{\dot{x}}^2$, $\sigma_{\dot{y}}^2$, $\sigma_{\dot{\theta}}^2$ are variances. The robot state prior encourages constant velocity (Barfoot, 2024). Unlike the previous experiment, we do not have a prior on the landmark positions. The landmark prior was necessary for conducting a proper Bayesian evaluation in the previous experiment, but here we have a standard SLAM problem.

The (nonlinear) odometry factors are

$$\psi_k = \frac{1}{2}\left(\mathbf{v}_k - \mathbf{C}_k\mathbf{x}_k\right)^T \mathbf{S}^{-1}\left(\mathbf{v}_k - \mathbf{C}_k\mathbf{x}_k\right), \tag{3.68}$$

where

$$\mathbf{v}_k = \begin{bmatrix} u_k \\ v_k \\ \omega_k \end{bmatrix}, \quad \mathbf{C}_k = \begin{bmatrix} 0 & 0 & 0 & \cos\theta_k & \sin\theta_k & 0 \\ 0 & 0 & 0 & -\sin\theta_k & \cos\theta_k & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{S} = \text{diag}\left(\sigma_u^2, \sigma_v^2, \sigma_\omega^2\right). \tag{3.69}$$

The $\mathbf{v}_k$ consists of measured forward, lateral, and rotational speeds in the robot frame, derived from the wheel encoder measurements. We set $v_k = 0$, which enforces the non-holonomy of the wheels as a penalty. The σ_u^2, σ_v^2, and σ_ω^2 are measurement noise variances.

The (nonlinear) bearing measurement factors are

$$\psi_{\ell,k} = \frac{1}{2}\frac{\left(\beta_{\ell,k} - g(\mathbf{m}_\ell, \mathbf{x}_k)\right)^2}{\sigma_r^2}, \tag{3.70}$$

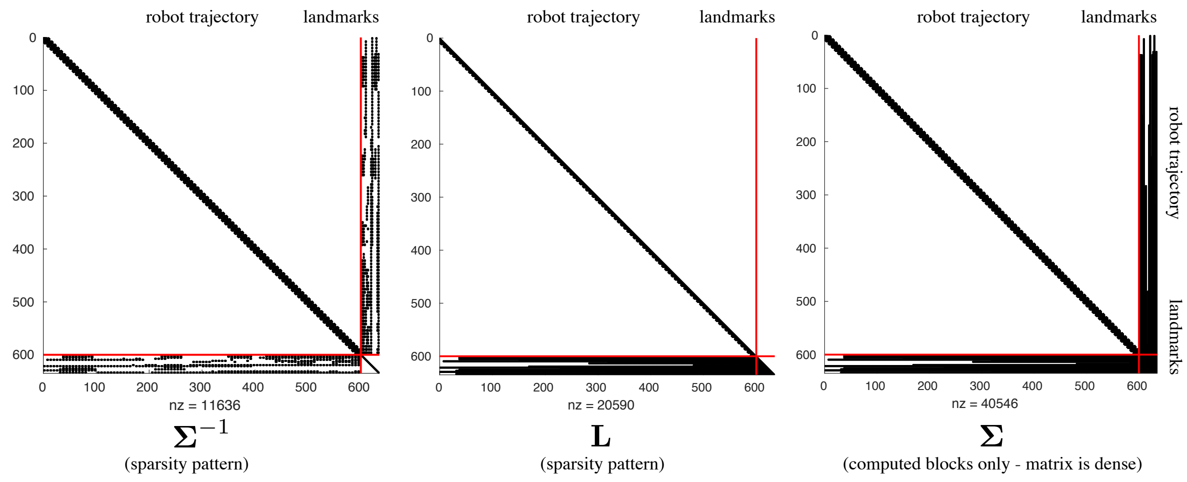


Figure 3.9: (Experiment 3) Sparsity patterns for the first 100 of $2,000$ timestamps of the robot dataset. The red partition lines separate the robot state variables from the landmark variables. The inverse covariance, $\boldsymbol{\Sigma}^{-1}$, is highly sparse due to the factor graph pattern in Figure 3.8, where only 11,636/401,956 = 2.9% of entries are nonzero. After performing a lower-diagonal-upper decomposition, the lower factor, $\mathbf{L}$, becomes more filled in due to the 'four corners of a box' rule, where 20,590/401,956 = 5.1% of entries are non-zero. Finally, we see that only a fraction of the entries of $\boldsymbol{\Sigma}$ are required even though this matrix is actually dense. Since $\boldsymbol{\Sigma}$ is symmetric, we only need to calculate 5.1% of it as well. For the full $2,000$-timestamp trajectory, the sparsity is even more favourable for ESGVI, but the landmark part of the pattern becomes difficult to visualize due to its small size relative to the trajectory.

with

$$g(\mathbf{m}_\ell, \mathbf{x}_k) = \operatorname{atan2}(y_\ell - y_k - d\sin\theta_k, x_\ell - x_k - d\cos\theta_k) - \theta_k, \tag{3.71}$$

where $\beta_{\ell,k}$ is a bearing measurement from the kth robot pose to the ℓth landmark, d is the offset of the laser rangefinder from the robot center in the longitudinal direction, and σ_r^2 is the measurement noise variance. Although the dataset also provides range measurements to the landmarks, we chose to omit these measurements to make the problem more difficult and accentuate the differences between the various algorithms. Our setup is similar to a monocular camera SLAM problem, which is known to be challenging.

Combining all the factors, our joint state/data likelihood is

$$-\ln p(\mathbf{x}, \mathbf{y}) = \sum_{k=0}^{K} \phi_k + \sum_{k=0}^{K} \psi_k + \sum_{k=1}^{K} \sum_{\ell=1}^{L} \psi_{\ell,k} + \text{constant}. \tag{3.72}$$

Note that not all $L = 17$ landmarks are actually seen at each timestep, so we remove those factors for unseen landmarks in the implementation.

We initialized our landmark locations using the bearing-only RANSAC (Fischler and

Bolles, 1981) strategy described by McGarey et al. (2017). We attempted to initialize our robot states using only wheel odometry, but this was too difficult in practice for methods that use the full Hessian (i.e., Newton's method). We instead used wheel odometry to initialize Gauss-Newton, which then provided a better initialization for Newton's method. Specifically, we used MAP Gauss-Newton to initialize MAP Newton and ESGVI-GN to initialize ESGVI. To evaluate our results using the groundtruth, we aligned the resulting landmark map to the groundtruth map since SLAM produces a relative solution. We compute the errors we report after this alignment. We also allowed all of the algorithms to make use of the usual Levenberg-Marquardt approximation of the Hessian and a line search at each iteration to increase robustness.

As an aside, it is worth noting that an alternative to using Gauss-Newton for initialization was demonstrated as an extension to our full ESGVI approach in later work by Goudar et al. (2022), where they propose a modification to the update scheme that guarantees that the update to the inverse posterior covariance will always be positive definite. We encourage readers to consider their work for improved numerical stability. However, using the Gauss-Newton variant with the alternate loss functional, as we have in our experiments, may still be desirable for its cheaper compute.

Similar to the previous experiments, Figure 3.10 provides the statistical results of several variants of our ESGVI algorithms. We see that the number of iterations to converge is higher than the previous experiments, with the ESGVI variants requiring a few more than the corresponding MAP algorithms. Again, we see the ESGVI variants reducing the loss functional, $V(q)$, lower than the MAP methods. The mean error is further away from zero for the ESGVI methods than MAP, but we believe this is due to the small number of trials compared to the previous two experiments. The squared error and squared Mahalanobis distance metrics are improved for the full ESGVI methods compared to the MAP methods and the ESGVI-GN method.

Table 3.1: (Experiment 3) Wall-clock time per iteration for tested algorithms.

algorithm label	wall-clock time per iteration [s]
MAP Newton	4.25
MAP GN	1.09
ESGVI deriv M=3	82.94
ESGVI-GN deriv-free M=3	24.53
ESGVI deriv-free M=4	113.32

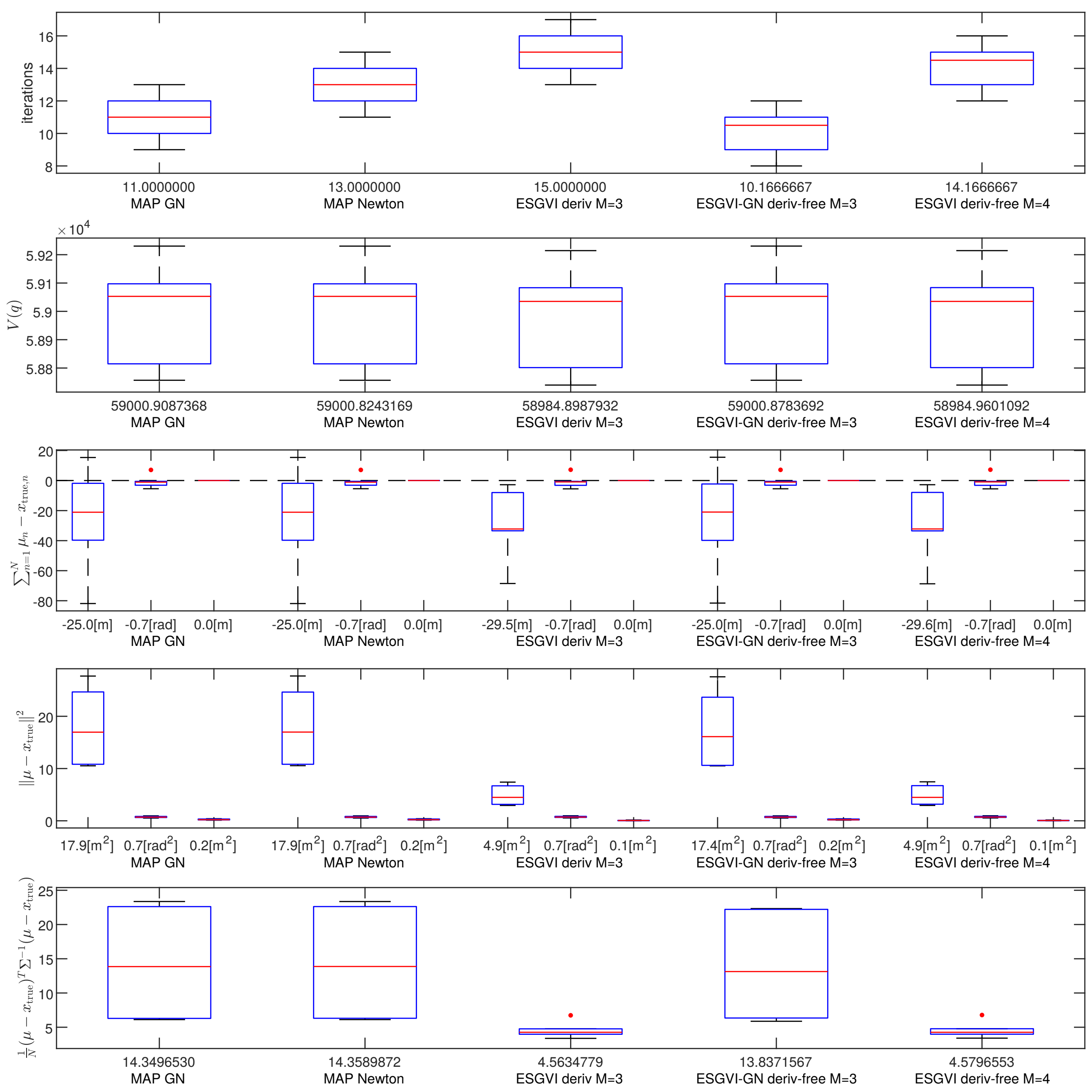


Figure 3.10: (Experiment 3) Statistical results of robot dataset shown as standard box-plots. The different rows show different performance metrics for the different variants of our algorithm (columns). Table 3.2 provides details of the different algorithms tested. The number below an algorithm label is its mean performance on that metric, averaged over all 2, 000 timestamps and six subsequences. Further details are discussed in the text.

Figure 3.11 shows the error plots for one of the six subsequences for the 'MAP GN' and 'ESGVI deriv-free M=4' algorithms. The ESGVI path is visibly better than the MAP in most sections. MAP seems to have underestimated the scale of the whole solution, resulting in worse performance in translation, while performing similarly in heading. Both algorithms are fairly consistent, with ESGVI being more confident. The other five subsequences have similar results.

Figure 3.9 shows the sparsity patterns of $\mathbf{\Sigma}^{-1}$, $\mathbf{L}$, and the blocks of $\mathbf{\Sigma}$ that are computed[1]. We show only the patterns for the first 100 timestamps for clarity, but each subsequence is actually $2,000$ timestamps long. In terms of computational complexity, all of the algorithms for this SLAM problem are $O(L^3 + L^2K)$ per iteration, where L is the number of landmarks and K is the number of timesteps. However, the wall-clock time required by the different algorithms varies significantly due to different numbers of iterations and the accuracy with which the required expectations in (3.27) are computed. Table 3.1 reports how long each algorithm took per iteration. The ESGVI methods come at a cost, but this may be acceptable for batch (i.e., offline) applications. It is also worth noting that we have made little attempt to optimize our implementation. We used a brute-force cubature method requiring M^{N_k} sigmapoints where N_k is the number of state variables involved in a factor. More efficient options could be swapped in to speed up the evaluation of each factor. Parallelization could also be employed at the factor level to evaluate the expectations in (3.27) in parallel across several cores/threads.

Table 3.2: Descriptions of variants of our GVI algorithm tested in our experiments.

algorithm label	method to evaluate expectations in (3.27)	M, # quadrature points (per dimension)
MAP Newton	analytical Jacobian and Hessian	1
MAP GN	analytical Jacobian and approximate Hessian	1
ESGVI deriv M=2	analytical Jacobian and Hessian + quadrature	2
ESGVI deriv M=3	analytical Jacobian and Hessian + quadrature	3
ESGVI deriv-free M=3	Stein's lemma + quadrature	3
ESGVI-GN deriv-free M=3	Stein's lemma + quadrature	3
ESGVI deriv-free M=4	Stein's lemma + quadrature	4
ESGVI deriv-free M=10	Stein's lemma + quadrature	10

3.7 Summary and Conclusions

We presented our Exactly Sparse Gaussian Variational Inference (ESGVI) approach and demonstrated that it is possible to compute a Gaussian that is 'closest' in terms of KL divergence from the true Bayesian posterior. For large-scale estimation problems, we exploited the fact that the joint likelihood of the state and data factors, a property

[1]The covariance matrix is actually dense.

of most common robotics problems, to show that the full (dense) covariance matrix is not required, only the blocks corresponding to the non-zero blocks of the (sparse) inverse covariance matrix. We further showed how to apply cubature methods (e.g., sigmapoints) within our framework resulting in a batch inference scheme that does not require analytical derivatives, yet is applicable to large-scale problems. The methods offer performance improvements (over MAP) that increase as the problem becomes more nonlinear and/or the posterior less concentrated. The methodology and experimental results in this chapter appeared in Barfoot et al. (2020) as a second-author contribution.

In summary, the contributions of this chapter are:

1. A computationally tractable approach to Gaussian Variational Inference (GVI) for large-scale estimation problems via exploitation of the sparsity formed from the factorization of the joint likelihood.

2. An implementation of sparse GVI that uses Gaussian cubature to avoid the need for computing analytical derivatives.

3. A conservative, cheaper (compute) approximation of Gaussian Variational Inference (GVI) that is applicable under mild nonlinearities and/or when the posterior is concentrated.

4. Various experiments in both simulation and on real-data demonstrating an improvement in performance over MAP.

In the following chapters of this thesis, we shift the focus from state estimation to parameter learning, of which we presented a brief sketch in Section 3.3.

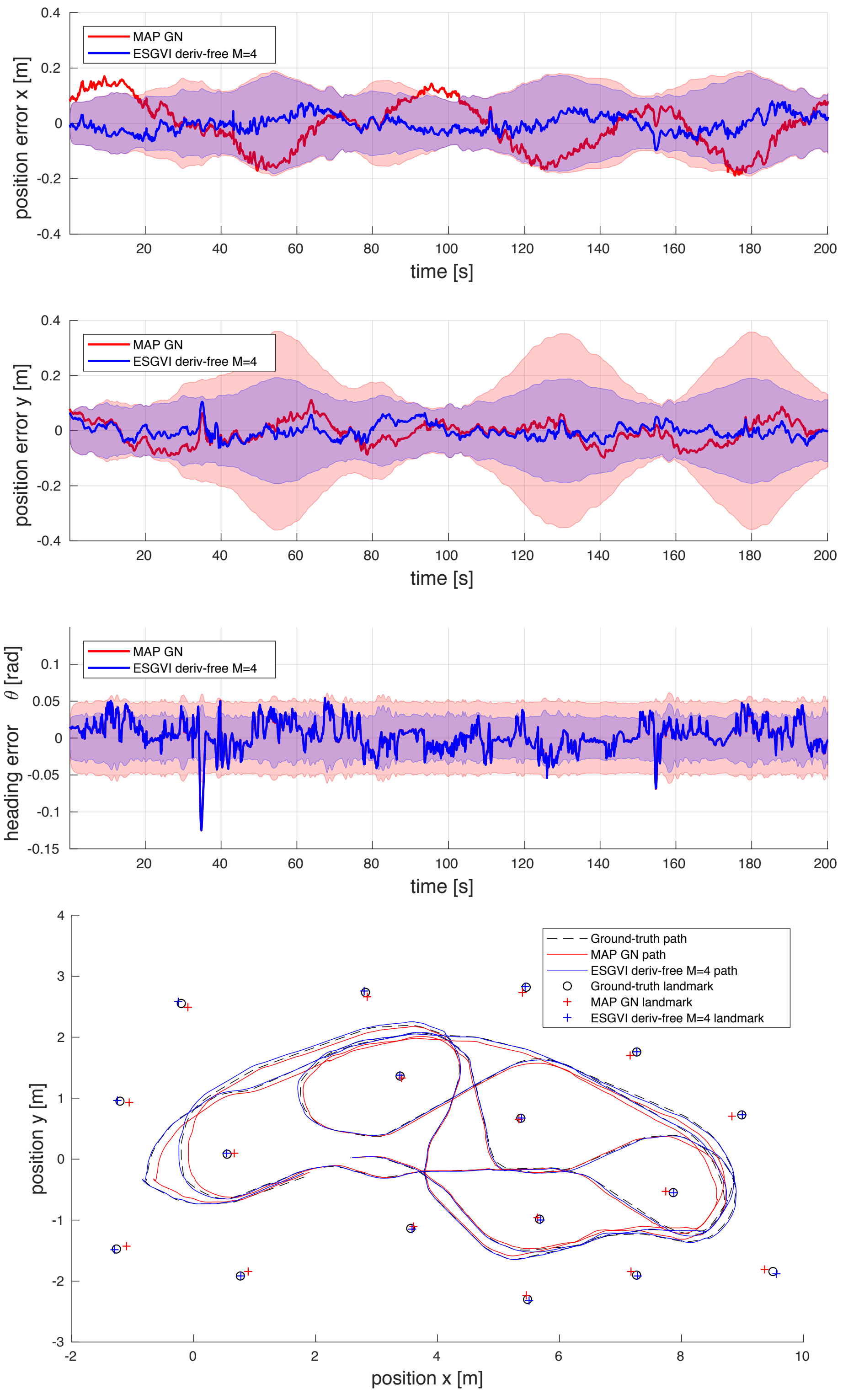


Figure 3.11: (Experiment 3) A comparison of 'MAP GN' and 'ESGVI deriv-free M=4' on one of the six subsequences of 2, 000 timestamps. Above, we see the individual error plots with 3σ covariance envelopes for the x, y, and θ components of the robot state. Below, we have an overhead view of the robot path and landmark map for the two algorithms, as well as groundtruth.

Chapter 4

Learning a Prior on Covariance

For state estimation, we often have a good understanding of the sensors we will use in real-world applications and may have handcrafted sensor models to apply in practice. As discussed in the earlier chapters, we can formulate a probabilistic estimator that relies on an assumption that uncertain quantities are approximately Gaussian and unbiased. However, we may find that the noise characteristics of our sensor are either unknown or differ from the manufacturer specification.

In this chapter, we explore the application of learning covariances for measurement models. We previously provided a sketch for learning a constant (i.e., non-varying) measurement covariance in Section 3.3, but a constant model may not always be suitable in practice. In order to achieve a covariance model that can adapt to varying noise levels, we apply our ESGVI and EM parameter learning framework to estimate the measurement covariance as part of the state, rather than treating it as a parameter. This is made possible in our framework by placing an Inverse-Wishart (IW) prior on the measurement covariance.

In summary, the main contribution of this chapter is the methodology for estimating measurement covariance as part of the state by incorporating an IW prior. We demonstrate our technique using a 36 km experimental dataset for lidar localization.

We present the methodology for our technique in Section 4.2. This is a paradigm shift from the constant covariance example in Section 3.3 as the measurement covariance is now treated as part of the state as an uncertain quantity (i.e., e-step). We instead train for the parameters of the prior on the covariance as part of the m-step. In Section 4.3, we show experimental results on a lidar localization problem and demonstrate a measurement covariance that adapts during sensor operation and is robust to measurement outliers in a fashion similar to a robust cost function.

The contents of this chapter is from a second-author contribution that appeared in a publication to the IEEE Robotics and Automation Letters (Wong et al., 2020b). This collaboration work was an opportunity to apply parameter learning using the ESGVI framework, which we developed in the previous chapter, to solve a practical problem of covariance learning, the research focus of the first author. As a second author, the main contribution was made to the theory and methodology (as a shared contribution), which we developed from our work in Chapter 3. Secondary contributions were made to the experiments, which we also present in this thesis chapter for completeness.

4.1 Related Work

In the simplest case, we may model the measurement noise as a zero-mean Gaussian with a constant covariance matrix. With access to high-quality groundtruth, we can simply calculate the sample covariance using a training dataset and be satisfied with this non-varying noise model throughout the lifetime of the sensor. However, a constant noise model may not be viable if the noise characteristics vary during sensor operation. An early approach for handling this problem was the adaptive Kalman Filter (Mehra, 1970; Stengel, 1994), which is a reactive method that recomputes the sample covariance over a trailing window of the latest sensor data.

Rather than a reactive approach, a feature-dependent predictive approach adapts the measurement covariance based on a training dataset. CELLO, a kernel regression method for the measurement covariance, was presented by Vega-Brown et al. (2013). By introducing handcrafted features that describe the measurements, local kernel estimates of the covariance were learned by weighting measurements based on feature similarity. Landry et al. (2019) adapt CELLO to model the uncertainty of pointcloud registration for Iterative Closest Point (ICP). Later work extended CELLO to unsupervised training using EM (Vega-Brown and Roy, 2013; Peretroukhin et al., 2016). Since CELLO is kernel-based, it is a form of 'lazy learning' and may not be computationally efficient for querying large training datasets. We can instead use a learned approach that generalizes to the training data offline (i.e., 'eager learning'), such as a DNN regression model. Brossard et al. (2020a) regress the noise variances of a vehicle model that penalizes lateral and vertical motion using a trailing window of IMU data as input. Predicting the covariance matrices for richer sensor data has been demonstrated for cameras (Liu et al., 2018; Russell and Reale, 2021) and for ICP using lidars (Torroba et al., 2020; De Maio and Lacroix, 2022).

In this chapter, we present our approach of treating the measurement covariance as part of the state we estimate, which involves using a corresponding prior distribution.

This idea fits into the reactive category for adapting covariances as we will later show that our estimated covariances are a weighted average between our covariance prior and the current measurement error residual. Our method is most similar to the work of Peretroukhin et al. (2016). They also apply an IW prior for learning a measurement covariance and train it using EM. However, our method differs in the formulation of the EM learning objective, where they choose to update the vehicle state as a parameter during the m-step. Our approach considers both the vehicle trajectory and measurement covariance as state variable quantities that are estimated during the e-step, and updates the parameters of the IW prior in the m-step. In the next chapter, we will demonstrate a feature-dependent regression model to predict the measurement variance of Doppler measurements from a FMCW lidar sensor.

4.2 Inverse-Wishart Prior on Covariance

In this section, we present how an Inverse-Wishart (IW) prior can be placed on a measurement covariance in our ESGVI and EM parameter learning framework. We start by redefining our joint likelihood as

$$p(\mathbf{x}, \mathbf{y}, \mathbf{R}) = p(\mathbf{x}, \mathbf{y}|\mathbf{R})p(\mathbf{R}), \tag{4.1}$$

where now we also include the covariances, $\mathbf{R} = \{\mathbf{R}_1, \mathbf{R}_2, \dots \mathbf{R}_K\}$, as random variables, distinguishing it from the $\mathbf{W}^{-1}$ we used in the constant covariance case. We also redefine our posterior estimate to be

$$q'(\mathbf{x}) = q(\mathbf{x})s(\mathbf{R}), \tag{4.2}$$

a product between a Gaussian $q(\mathbf{x})$ and a posterior distribution for the covariances, $s(\mathbf{R})$. Following the usual EM decomposition, our negative log-likelihood objective can be written as

$$\begin{aligned} \mathscr{L} = -\ln p(\mathbf{y}) = & \underbrace{\int\!\!\int q(\mathbf{x})s(\mathbf{R}) \ln\left(\frac{p(\mathbf{x}, \mathbf{R}|\mathbf{y})}{q(\mathbf{x})s(\mathbf{R})}\right) d\mathbf{x}\, d\mathbf{R}}_{\leq 0} \\ & \underbrace{- \int\!\!\int q(\mathbf{x})s(\mathbf{R}) \ln\left(\frac{p(\mathbf{x}, \mathbf{y}|\mathbf{R})p(\mathbf{R})}{q(\mathbf{x})s(\mathbf{R})}\right) d\mathbf{x}\, d\mathbf{R}}_{\text{upper bound}}. \end{aligned} \tag{4.3}$$

We choose to define the posterior over the covariances as

$$s(\mathbf{R}) = \delta(\mathbf{R} - \boldsymbol{\Upsilon}), \tag{4.4}$$

where $\delta(\cdot)$ is the Dirac delta function (interpreted as a PDF) and $\boldsymbol{\Upsilon} = \{\boldsymbol{\Upsilon}_1, \boldsymbol{\Upsilon}_2 \ldots \boldsymbol{\Upsilon}_K\}$ is the set of optimal covariances. The upper bound now simplifies to

$$-\int q(\mathbf{x}) \ln \left(p(\mathbf{x}, \mathbf{y}|\boldsymbol{\Upsilon}) p(\boldsymbol{\Upsilon})\right) d\mathbf{x} + \int q(\mathbf{x}) \ln q(\mathbf{x})\, d\mathbf{x} + \underbrace{\int s(\mathbf{R}) \ln s(\mathbf{R}) d\mathbf{R}}_{\text{indep. of } \boldsymbol{\Upsilon}},$$

where we have abused notation and written $p(\mathbf{R} = \boldsymbol{\Upsilon})$ as $p(\boldsymbol{\Upsilon})$, and similarly will later write $p(\mathbf{R}_k = \boldsymbol{\Upsilon}_k)$ as $p(\boldsymbol{\Upsilon}_k)$. We view our selection of the delta function (in our posterior over the covariances) as a convenient way of showing how we can approximate a Gaussian distribution for the trajectory and a MAP approximation of the covariances in a single variational framework. The last term is the differential entropy of a Dirac delta function, and because it is independent of our variational parameter, $\boldsymbol{\Upsilon}$, we choose to drop it from our loss functional.

We assume $p(\boldsymbol{\Upsilon})$ factors as

$$p(\boldsymbol{\Upsilon}) = \prod_{k=1}^{K} p(\boldsymbol{\Upsilon}_k). \tag{4.5}$$

We apply an IW prior over our covariances by defining

$$p(\boldsymbol{\Upsilon}_k) = \frac{|\boldsymbol{\Psi}|^{\nu/2}}{2^{\frac{\nu d}{2}} \Gamma_d(\frac{\nu}{2})} |\boldsymbol{\Upsilon}_k|^{-\frac{\nu+d+1}{2}} \exp\left(-\frac{1}{2} \operatorname{tr}(\boldsymbol{\Psi} \boldsymbol{\Upsilon}_k^{-1})\right), \tag{4.6}$$

where d is the dimension of $\boldsymbol{\Upsilon}_k$, $\boldsymbol{\Psi} \in \mathbb{R}^{d \times d} > 0$ is the scale matrix, $\nu > d-1$ is the degrees-of-freedom (DOF), and $\Gamma_d(\cdot)$ is the multivariate Gamma function. We choose to estimate the scale matrix parameter $\boldsymbol{\Psi}$ and leave the degrees-of-freedom ν as a metaparameter.

Now we define our factors as

$$-\ln(\boldsymbol{\Upsilon}) = \sum_{k=1}^{K} -\ln p(\boldsymbol{\Upsilon}_k) = \sum_{k=1}^{K} \phi_k^w(\boldsymbol{\Upsilon}_k|\boldsymbol{\Psi}) = \phi^w(\boldsymbol{\Upsilon}|\boldsymbol{\Psi}).$$

Dropping constant terms, the loss functional can finally be written as

$$V(q'|\boldsymbol{\Upsilon}, \boldsymbol{\Psi}) = \sum_{k=1}^{K} \mathbb{E}_{q_k}\left[\phi_k^m(\mathbf{x}_k|\boldsymbol{\Upsilon}_k) + \phi_k^w(\boldsymbol{\Upsilon}_k|\boldsymbol{\Psi})\right] + \frac{1}{2} \ln\left(|\boldsymbol{\Sigma}^{-1}|\right), \tag{4.7}$$

where

$$\phi_k^m(\mathbf{x}_k|\boldsymbol{\Upsilon}_k) = \frac{1}{2}\left(\mathbf{e}_k(\mathbf{x}_k)^T\boldsymbol{\Upsilon}_k^{-1}\mathbf{e}_k(\mathbf{x}_k) - \ln(|\boldsymbol{\Upsilon}_k^{-1}|\right), \tag{4.8}$$

$$\phi_k^w(\boldsymbol{\Upsilon}_k|\boldsymbol{\Psi}) = -\frac{\alpha-1}{2}\ln|\boldsymbol{\Upsilon}_k^{-1}| - \frac{\nu}{2}\ln|\boldsymbol{\Psi}| + \frac{1}{2}\mathrm{tr}(\boldsymbol{\Psi}\boldsymbol{\Upsilon}_k^{-1}), \tag{4.9}$$

with $\alpha = \nu + d + 2$.

In the e-step, we hold $\boldsymbol{\Psi}$ fixed and optimize for $\boldsymbol{\Upsilon}_k$, which we accomplish by taking the derivative of the loss functional as follows:

$$\frac{\partial V}{\partial \boldsymbol{\Upsilon}_k^{-1}} = \frac{1}{2}\mathbb{E}_{q_k}\left[\mathbf{e}_k(\mathbf{x}_k)\mathbf{e}_k(\mathbf{x}_k)^T\right] - \frac{1}{2}\alpha\boldsymbol{\Upsilon}_k + \frac{1}{2}\boldsymbol{\Psi}. \tag{4.10}$$

Setting the derivative to zero,

$$\boldsymbol{\Upsilon}_k = \frac{1}{\alpha}\boldsymbol{\Psi} + \frac{1}{\alpha}\mathbb{E}_{q_k}\left[\mathbf{e}_k(\mathbf{x}_k)\mathbf{e}_k(\mathbf{x}_k)^T\right] = \frac{\alpha-1}{\alpha}\underbrace{\left(\frac{\boldsymbol{\Psi}}{\alpha-1}\right)}_{\text{IW mode}} + \frac{1}{\alpha}\mathbb{E}_{q_k}\left[\mathbf{e}_k(\mathbf{x}_k)\mathbf{e}_k(\mathbf{x}_k)^T\right], \tag{4.11}$$

where we see the optimal $\boldsymbol{\Upsilon}_k$ is a weighted average between the mode of the IW distribution and the optimal constant covariance estimate from (3.55) at a single marginal factor. Since our e-step in ESGVI is already iterative, we can seamlessly extend it by applying (4.11) as iteratively reweighted least squares (IRLS).

In the m-step, we hold $\boldsymbol{\Upsilon}$ fixed and optimize for $\boldsymbol{\Psi}$, which we accomplish by taking the derivative of the loss functional as follows:

$$\frac{\partial V}{\partial \boldsymbol{\Psi}} = \sum_{k=1}^{K}\left(-\frac{\nu}{2}\boldsymbol{\Psi}^{-1} + \frac{1}{2}\mathbf{R}_k^{-1}\right). \tag{4.12}$$

Setting the derivative to zero,

$$\boldsymbol{\Psi}^{-1} = \frac{1}{K\nu}\sum_{k=1}^{K}\boldsymbol{\Upsilon}_k^{-1}. \tag{4.13}$$

Applying (4.11) in the e-step and (4.13) in the m-step, in practice we found that our optimization scheme was ill-posed, resulting in our covariance estimates tending toward the positive-definite boundary (i.e., the zero matrix). As a practical solution, we propose constraining the determinant of $\boldsymbol{\Psi}$ to be a constant β, which constrains the volume of the uncertainty ellipsoid of the corresponding measurements to be fixed. We accomplish

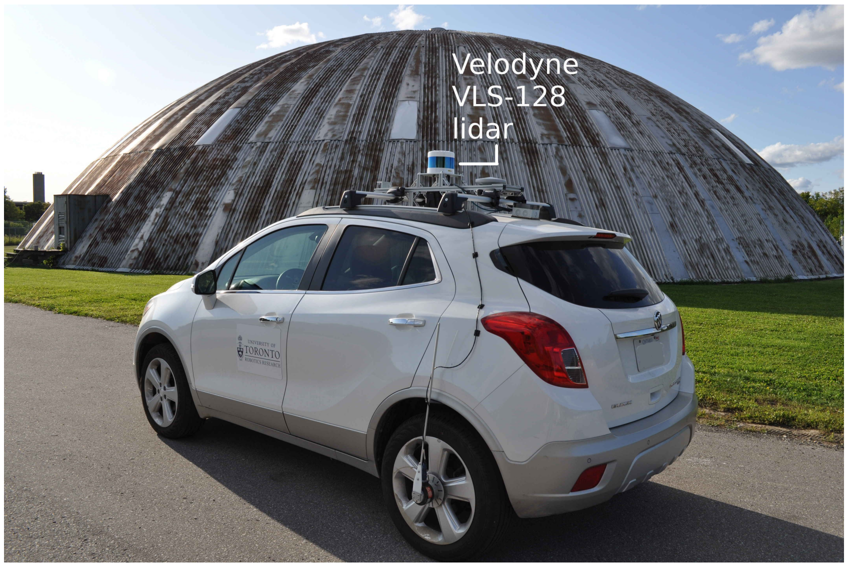


Figure 4.1: The data collection vehicle used for our lidar localization experiments. The vehicle is equipped with a Velodyne VLS-128 lidar and an Applanix POS LV for the groundtruth trajectory.

this by scaling the latest $\mathbf{\Psi}$ update as follows:

$$\mathbf{\Psi}_{\text{constrained}} \leftarrow \left(\beta\,|\mathbf{\Psi}|^{-1}\right)^{\frac{1}{d}}\ \mathbf{\Psi}. \tag{4.14}$$

While this determinant constraint limits the learning capacity of our noise model, in practice we can rely on the noise models of other factors (e.g., the motion prior) to relatively adapt to our selection of β during training.

4.3 Experiment: Lidar Localization

We present our experiments on learning an IW prior for the measurement covariance in this section, which appeared in a publication as a second author contribution (Wong et al., 2020b). Our experiments use the vehicle dataset collected by an earlier publication by Wong et al. (2020a). The dataset consists of 36 km of driving, with Velodyne VLS-128 lidar data and an Applanix POS LV positioning system. An image of the experimental setup is shown in Figure 4.1, which is an earlier setup for the *Boreas* data collection vehicle that we will later use for our experiments in the following chapter.

There are two sources of 6-DOF vehicle pose measurements. The first is from the Applanix POS LV, which we treat as groundtruth. The second is from a lidar localization system, also from Applanix, which localizes the lidar data to a prebuilt high-definition pointcloud map. We will treat the lidar localization estimates as pose *pseudomeasurements*[1] and place a prior on its measurement covariance to demonstrate our parameter learning approach.

We use Route A[2], our 16 km long training set, to learn the parameters of our noise models. For inference, we perform a batch trajectory optimization on Route B[3], our 20 km long test set, using the learned noise model parameters of our motion prior and measurements.

4.3.1 Loss Functional and Factors

Loss Functional

We define our trajectory as $\mathbf{x} = \{\mathbf{T}_k, \boldsymbol{\varpi}_k | \, k = 1, 2, \ldots, K\}$, where $\mathbf{T}_k \in SE(3)$ is the vehicle pose at time t_k and $\boldsymbol{\varpi}_k$ is the corresponding body-centric vehicle velocity. The estimation times, t_k, correspond exactly to the measurement times of the lidar localization measurements, which is approximately a uniform temporal spacing of 100 ms.

The loss functional corresponding to this experiment is

$$V(q'|\boldsymbol{\Upsilon}, \boldsymbol{\Psi}, \mathbf{W}_{gt}, \mathbf{Q}_c) = \mathbb{E}_{q'}[\phi^p(\mathbf{x}|\mathbf{Q}_c) + \phi^m(\mathbf{x}|\mathbf{W}_{gt}) + \phi^m(\mathbf{x}|\boldsymbol{\Upsilon}) + \phi^w(\boldsymbol{\Upsilon}|\boldsymbol{\Psi})] + \frac{1}{2}\ln\left(|\boldsymbol{\Sigma}^{-1}|\right), \tag{4.15}$$

where $\phi^p(\mathbf{x}|\mathbf{Q}_c)$ are motion prior factors, $\phi^m(\mathbf{x}|\mathbf{W}_{gt})$ are the groundtruth pose factors (when available during training), and $\phi^m(\mathbf{x}|\boldsymbol{\Upsilon})$ and $\phi^w(\boldsymbol{\Upsilon}|\boldsymbol{\Psi})$ are the lidar measurement factors with an IW prior over the covariances.

We illustrate our loss functional in Figure 4.2 as a factor graph, where we can train with or without the groundtruth factors shown inside the dashed box. For the sake of conciseness in our notation, we denote $\phi^p(\mathbf{x}_{k-1,k}|\mathbf{Q}_c)$ as $\phi^p_{\mathbf{x}_{k-1,k}|\mathbf{Q}_c}$, $\phi^m(\mathbf{x}_k|\mathbf{W}_{gt})$ as $\phi^m_{\mathbf{x}_k|\mathbf{W}_{gt}}$, $\phi^m(\mathbf{x}_k|\boldsymbol{\Upsilon}_k)$ as $\phi^m_{\mathbf{x}_k|\boldsymbol{\Upsilon}_k}$, and $\phi^w(\boldsymbol{\Upsilon}_k|\boldsymbol{\Psi})$ as $\phi^w_{\boldsymbol{\Upsilon}_k|\boldsymbol{\Psi}}$. We show the details of each factor in the following subsections.

Pose Measurement Factors

In our experiments, we place an IW prior on the measurement covariance for the lidar measurements. We defined this factor in Section 4.2, which we repeat here for conve-

[1]We will simply refer to these localization poses as lidar measurements for the rest of this chapter.
[2] Map available at: `https://tinyurl.com/rrjgxaj`
[3] Map available at: `https://tinyurl.com/r5m78nq`

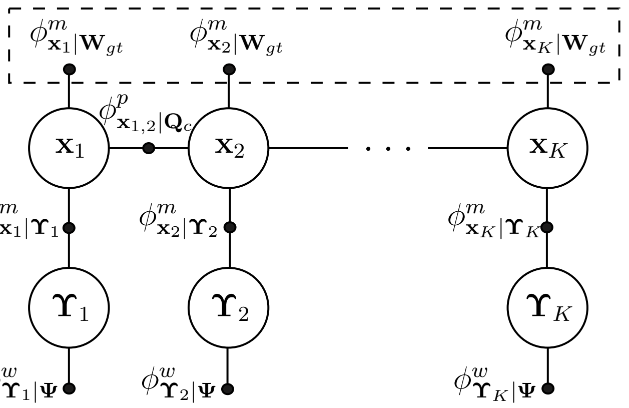

$\phi^m_{\mathbf{x}_1|\mathbf{W}_{gt}}$
$\phi^m_{\mathbf{x}_2|\mathbf{W}_{gt}}$
$\phi^m_{\mathbf{x}_K|\mathbf{W}_{gt}}$
$\phi^p_{\mathbf{x}_{1,2}|\mathbf{Q}_c}$
$\mathbf{x}_1$
$\mathbf{x}_2$
$\mathbf{x}_K$
$\phi^m_{\mathbf{x}_1|\mathbf{\Upsilon}_1}$
$\phi^m_{\mathbf{x}_2|\mathbf{\Upsilon}_2}$
$\phi^m_{\mathbf{x}_K|\mathbf{\Upsilon}_K}$
$\mathbf{\Upsilon}_1$
$\mathbf{\Upsilon}_2$
$\mathbf{\Upsilon}_K$
$\phi^w_{\mathbf{\Upsilon}_1|\mathbf{\Psi}}$
$\phi^w_{\mathbf{\Upsilon}_2|\mathbf{\Psi}}$
$\phi^w_{\mathbf{\Upsilon}_K|\mathbf{\Psi}}$

Figure 4.2: Factor graph for our vehicle estimation problem in Experiment A (see Section 4.3.2). White circles represent random variables to be estimated (vehicle state $\mathbf{x}$ and measurement covariances $\mathbf{\Upsilon}$). Small black dots represent factors in the joint likelihood of the data and the state. Binary motion prior factors, $\phi^p_{\mathbf{x}_{k-1,k}|\mathbf{Q}_c}$, depend on parameter $\mathbf{Q}_c$. Unary groundtruth pose factors (if available), $\phi^m_{\mathbf{x}_k|\mathbf{W}_{gt}}$, depend on parameter $\mathbf{W}_{gt}$. Factors $\phi^m_{\mathbf{x}_k|\mathbf{\Upsilon}_k}$ and $\phi^w_{\mathbf{\Upsilon}_k|\mathbf{\Psi}}$ are for applying an Inverse-Wishart prior over our measurement pose covariances, $\mathbf{\Upsilon}$, and depend on parameter $\mathbf{\Psi}$. We are able to learn parameters $\mathbf{Q}_c$ and $\mathbf{\Psi}$, even without groundtruth factors (factors inside dashed box).

nience:

$$\phi^m(\mathbf{x}|\mathbf{\Upsilon}) = \sum_{k=1}^{K} \phi^m_k(\mathbf{x}_k|\mathbf{\Upsilon}_k) = \sum_{k=1}^{K} \frac{1}{2}\left(\mathbf{e}_{m,k}(\mathbf{x}_k)^T \mathbf{\Upsilon}_k^{-1} \mathbf{e}_{m,k}(\mathbf{x}_k) - \ln(|\mathbf{\Upsilon}_k^{-1}|\right), \tag{4.16}$$

$$\phi^w(\mathbf{\Upsilon}|\mathbf{\Psi}) = \sum_{k=1}^{K} \phi^w_k(\mathbf{\Upsilon}_k|\mathbf{\Psi}) = \sum_{k=1}^{K} \left(-\frac{\alpha-1}{2}\ln|\mathbf{\Upsilon}_k^{-1}| - \frac{\nu}{2}\ln|\mathbf{\Psi}| + \frac{1}{2}\mathrm{tr}(\mathbf{\Psi}\mathbf{\Upsilon}_k^{-1})\right). \tag{4.17}$$

We can now specify the error function as the following:

$$\mathbf{e}_{m,k}(\mathbf{x}_k) = \ln(\mathbf{T}_k \mathbf{T}_{\text{meas},k}^{-1}), \tag{4.18}$$

where $\mathbf{T}_{\text{meas},k}$ is a lidar measurement corresponding to time t_k. We choose to fix the parameters to $\nu = 6$ and $\beta = 1$ and learn the parameters $\mathbf{\Psi}$

We optionally train our parameters by including the groundtruth poses from the Applanix POS LV. We choose to model this factor using a constant measurement covariance, $\mathbf{W}_{gt}$:

$$\phi^m(\mathbf{x}|\mathbf{W}_{gt}) = \sum_{k=1}^{K} \phi^m_k(\mathbf{x}|\mathbf{W}_{gt}) = \sum_{k=1}^{K} \frac{1}{2}\left(\mathbf{e}_{m,k}(\mathbf{x}_k)^T \mathbf{W}_{gt}^{-1} \mathbf{e}_{m,k}(\mathbf{x}_k) - \ln(|\mathbf{W}_{gt}^{-1}|\right), \tag{4.19}$$

where we apply the same error function as in (4.18). Recall that we demonstrated how to learn a constant covariance as part of the m-step in Section 3.3, which we will apply in our experiments for $\mathbf{W}_{gt}$. This factor will only be used during training and will be removed for testing.

Motion Prior Factor

We apply the $SE(3)$ White-Noise-on-Acceleration (WNOA) motion prior by Anderson and Barfoot (2015), which is defined as

$$\begin{aligned} \dot{\mathbf{T}}(t) &= \boldsymbol{\varpi}(t)^\wedge \mathbf{T}(t), \\ \dot{\boldsymbol{\varpi}} &= \mathbf{w}(t), \quad \mathbf{w}(t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{Q}_c \delta(t-t')), \end{aligned} \tag{4.20}$$

where $\mathbf{T}(t) \in SE(3)$ is a continuous-time representation of the pose, $\boldsymbol{\varpi}(t) \in \mathbb{R}^6$ is the body-centric vehicle velocity, $\mathbf{w}(t) \in \mathbb{R}^6$ is a zero-mean, white-noise Gaussian process, and the operator, $\wedge$, transforms an element of $\mathbb{R}^6$ into a member of Lie algebra, $\mathfrak{se}(3)$. Recall that the state at time t_k is $\mathbf{x}_k = \{\mathbf{T}_k, \boldsymbol{\varpi}_k\}$. Similarly, we will write $\mathbf{x}_{k-1,k}$ to be the state at two consecutive times, t_{k-1} and t_k. Written in factor form, our motion prior is

$$\phi^p(\mathbf{x}|\mathbf{Q}_c) = \sum_{k=2}^{K} \phi_k^p(\mathbf{x}_{k-1,k}|\mathbf{Q}_c) = \sum_{k=2}^{K} \frac{1}{2}\left(\mathbf{e}_{p,k}^T \mathbf{Q}_k^{-1} \mathbf{e}_{p,k} + \ln|\mathbf{Q}_k|\right), \tag{4.21}$$

where

$$\mathbf{e}_{p,k} = \begin{bmatrix} \ln(\mathbf{T}_k \mathbf{T}_{k-1}^{-1})^\vee - (t_k - t_{k-1})\boldsymbol{\varpi}_{k-1} \\ \boldsymbol{\mathcal{J}}^{-1}(\ln(\mathbf{T}_k \mathbf{T}_{k-1}^{-1})^\vee)\boldsymbol{\varpi}_k - \boldsymbol{\varpi}_{k-1} \end{bmatrix}, \tag{4.22}$$

and the covariance of the prior, $\mathbf{Q}_k$, is defined as

$$\mathbf{Q}_k = \mathbf{Q}_{\Delta t} \otimes \mathbf{Q}_c, \quad \mathbf{Q}_k^{-1} = \mathbf{Q}_{\Delta t}^{-1} \otimes \mathbf{Q}_c^{-1},$$

$$\mathbf{Q}_{\Delta t} = \begin{bmatrix} \frac{1}{3}\Delta t^3 & \frac{1}{2}\Delta t^2 \\ \frac{1}{2}\Delta t^2 & \Delta t \end{bmatrix}, \quad \mathbf{Q}_{\Delta t}^{-1} = \begin{bmatrix} 12\Delta t^{-3} & -6\Delta t^{-2} \\ -6\Delta t^{-2} & 4\Delta t^{-1} \end{bmatrix},$$

where $\otimes$ is the Kronecker product.

In our experiments, we will additionally learn the parameters of this prior, the power spectral density matrix $\mathbf{Q}_c$, as part of the m-step. To do so, we evaluate the derivative of our loss with respect to $Q_{c_{ij}}$, the (i,j) matrix element of $\mathbf{Q}_c$, which is

$$\frac{\partial V(q|\boldsymbol{\theta})}{\partial Q_{c_{ij}}} = \frac{1}{2}\mathrm{tr}\left(\sum_{k=2}^{K} \mathbb{E}_{q_{k-1,k}}[\mathbf{e}_{p,k}\mathbf{e}_{p,k}^T](\mathbf{Q}_{\Delta t}^{-1} \otimes \mathbf{1}_{ij})\right) - \frac{1}{2}(K-1)\mathrm{dim}(\mathbf{Q}_{\Delta t})Q_{c_{ij}}, \tag{4.23}$$

where $q_{k-1,k}$ is the marginal posterior at two consecutive times, t_{k-1} and t_k. Setting the derivative to zero, the optimal estimate of our parameter is then

$$Q_{c_{ij}} = \frac{\mathrm{tr}\left(\sum_{k=2}^{K} \mathbb{E}_{q_{k-1,k}}[\mathbf{e}_{p,k}\mathbf{e}_{p,k}^T](\mathbf{Q}_{\Delta t}^{-1} \otimes \mathbf{1}_{ij})\right)}{\mathrm{dim}(\mathbf{Q}_{\Delta t})(K-1)}. \tag{4.24}$$

4.3.2 Experiment A: Training With and Without Groundtruth

In Experiment A, we only use the lidar measurements to train our model parameters (i.e., training without groundtruth). As a benchmark, we also train another set of model parameters where we additionally include groundtruth poses in our training (i.e., training with incomplete groundtruth). This is different from earlier work by Wong et al. (2020a), where the training method required groundtruth of the entire state (training with complete groundtruth), which for our problem setup is the vehicle pose and velocity.

Figure 4.3 shows the error plots where we have trained without groundtruth for our estimated x, y, and z positions, along with their 3σ covariance envelopes. We can see that the errors consistently remain within the covariance envelopes. However, we note that our estimator appears to be underconfident. We believe that this is a result of our decision to constrain $|\boldsymbol{\Psi}| = \beta = 1$ in order for our training method to work in practice. This decision is analogous to fixing the volume of the covariance ellipsoid to be constant. In doing so, we relied on the learned covariance of the motion prior to adjust relative to the measurement covariances. The posterior mean is unaffected by this choice, but not the posterior covariance.

Table 4.1 shows the translational Root Mean Squared Error (RMSE) from both training methods on all test sequences. For comparison, we also include the results from earlier work by Wong et al. (2020a) (first column). This publication trained the same motion prior on the same dataset using complete groundtruth (both pose and velocity) and did not learn the measurement covariances for the lidar localization, instead using the noise levels predicted by the Applanix lidar localization system. While all three methods perform on par with each other, the key benefit is that our newly proposed method does not require any groundtruth during training. However, we note that our lidar measurements are quite accurate relative to the groundtruth, which we believe plays a factor in our ability to achieve comparable performance when trained unsupervised.

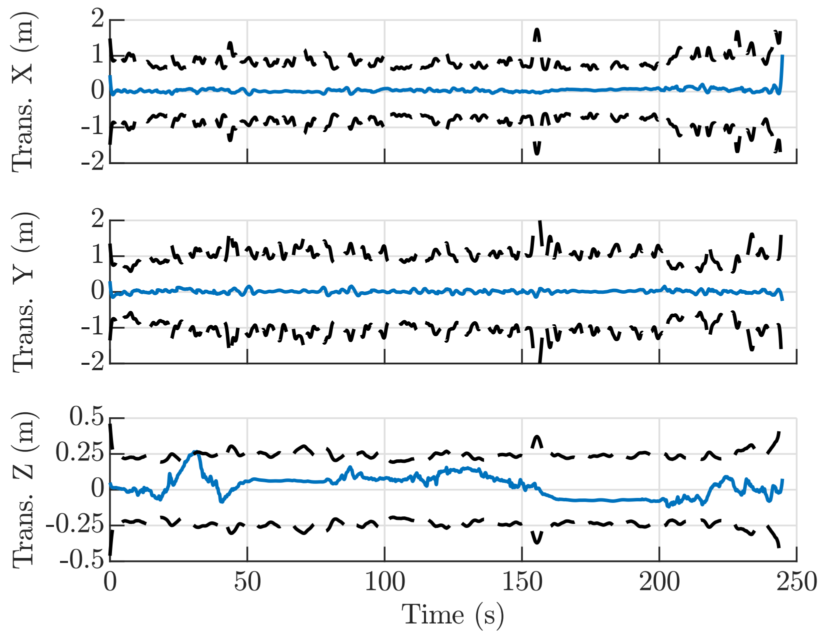


Figure 4.3: Experiment A: Error plots (blue lines) along with the 3σ covariance envelopes (black dashed lines) when parameters are trained without groundtruth. The resulting trajectory error from our estimator consistently remains within the covariance envelopes.

To further validate our method and show that we can train with noisier measurements and without groundtruth, we perform a sensitivity test by artificially adding additional noise to the measurements. We keep the statistics of the additional noise unknown to the training process. We use the following $SE(3)$ perturbation scheme to inject noise into the translational portion of our pose measurements:

$$\mathbf{T}_{\text{noisy}} = \exp(\boldsymbol{\epsilon}^\wedge)\mathbf{T}_{\text{meas}}, \tag{4.25}$$

where

$$\boldsymbol{\epsilon} = \begin{bmatrix} \boldsymbol{\epsilon}_{1:3} \\ \mathbf{0} \end{bmatrix}, \quad \boldsymbol{\epsilon}_{1:3} \sim \mathcal{N}\left(\mathbf{0}, \sigma^2\mathbf{I}\right). \tag{4.26}$$

We vary σ from 0.25 m to 1 m, injecting the same amount of noise into the test measurements and training measurements.

Table 4.2 shows the results of our sensitivity test. Our test errors change with increasing noise on measurements in both our training and test set. While measurement error increases significantly (up to over 1.6 m), we are still able to achieve translational errors below 0.5 m on our estimated trajectory. This experiment demonstrates that we are still able to learn reasonable parameters of our system without any groundtruth and

Table 4.1: Experiment A: Comparison of translational errors on test set between training with complete groundtruth, with incomplete groundtruth, and without groundtruth (GT). We note that the first column did not learn the measurement covariances.

Seq no.	Trained with complete GT (Wong et al., 2020a) (m)	Trained with incomplete GT (m)	Trained without GT (m)
0	0.0690	0.0720	0.0717
1	0.0888	0.1003	0.0925
2	0.4071	0.4148	0.4106
3	0.1947	0.1908	0.1847
4	0.2868	0.2866	0.2820
5	0.5703	0.5592	0.5549
6	0.3292	0.3014	0.2965
7	0.2207	0.2248	0.2230
8	0.1115	0.1151	0.1199
9	0.0979	0.1026	0.0997
AVG	0.2376	0.2368	0.2335

Table 4.2: Experiment A: Analysis of how increasing noise on measurements affects the parameter learning method. Even with measurement errors of over 1.6 m the errors on the estimated trajectory are under 0.5 m.

Measurement noise (m)	Estimated trajectory errors (m)
0.2407	0.2335
0.5010	0.2909
0.8653	0.3289
1.2481	0.3936
1.6383	0.4566

noisier measurements (relative to our original experiment).

4.3.3 Experiment B: Training and Testing With Measurement Outliers

In Experiment B, we show that our formulation of an IW prior on our measurement covariances results in outlier rejection. We artificially introduce outliers in our training and test set. With 5% probability, we apply the following perturbation to the our pose measurements:

$$\mathbf{T}_{\text{outlier}} = \exp(\boldsymbol{\epsilon}^{\wedge})\mathbf{T}_{\text{meas}}, \tag{4.27}$$

with

$$\boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 & \epsilon_2 & \epsilon_3 & \epsilon_4 & \epsilon_5 & \epsilon_6 \end{bmatrix}^T, \quad \epsilon_i \sim \mathcal{U}(-200, 200). \tag{4.28}$$

Figure 4.4 shows a qualitative illustration of the measurement outliers on sequence 3 of our test set.

We now compare the performance between the cases where we have treated the measurement covariance, $\mathbf{W}$, as a static parameter to be learned (using the same data likelihood objective, see Section 3.3), and where we have treated the measurement covariance at each time as a random variable and learn the parameter, $\boldsymbol{\Psi}$, of the IW prior.

The left portion of Table 4.3 shows the resulting translational errors on our test trajectory for this experiment (Experiment B). Without the IW prior, the estimation framework fails to reject outliers and results in an average translation error above 5 m. In comparison, using the IW prior results in an average error of only 0.2365 m, which is a significant improvement and similar to our previous experiment where we did not have any outliers (0.2335 m in Table 4.1). We expect that similar performance would be achievable if an existing outlier rejection scheme, such as M-estimation, was applied alongside the static $\mathbf{W}$. However, the benefit we highlight with our proposed method is the ability to learn the measurement covariance (without groundtruth) and achieve outlier rejection using a unified approach, rather than learning a static covariance (which can be challenging in the presence of outliers) and applying outlier rejection post hoc.

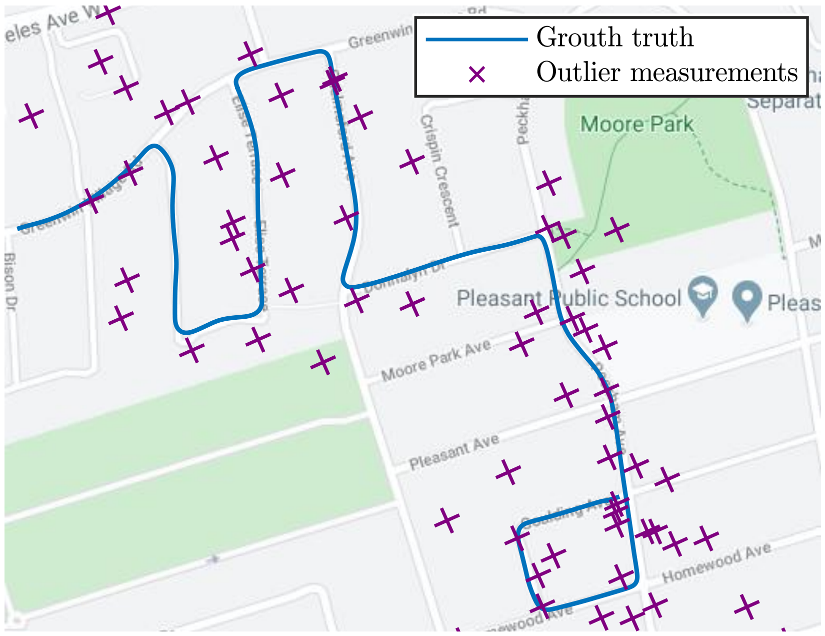


Figure 4.4: Experiments B & C: Measurement outliers overlaid with the groundtruth trajectory on sequence 3 of the test set. Background image from Google Maps.

Table 4.3: Experiments B & C: Translational errors using a static measurement covariance compared to using an IW prior when we have outliers in our test set. In Experiment B, we train with outliers and in Experiment C, we train without outliers.

	Experiment B		Experiment C	
Seq no.	Static **W** (m)	IW prior (m)	Static **W** (m)	IW prior (m)
0	6.1976	0.0773	7.3504	0.0731
1	5.8371	0.0979	6.0754	0.0948
2	5.3652	0.4125	5.5771	0.4096
3	5.1217	0.1860	5.8157	0.1873
4	5.5186	0.2807	5.5503	0.2826
5	5.4780	0.5563	6.3057	0.5554
6	6.3936	0.3004	7.1858	0.2995
7	5.6898	0.2274	6.0332	0.2256
8	6.3717	0.1233	9.3079	0.1224
9	6.8032	0.1036	8.1237	0.1046
AVG	5.8776	0.2365	6.7325	0.2355

4.3.4 Experiment C: Training Without and Testing With Measurement Outliers

In Experiment B, we included outlier measurements in both the training and test set and saw that the IW prior allows us to achieve comparable errors to the case with no outliers. To see if this still holds even when we do not see any outliers in training, in Experiment C we now train without any outliers, but test with outliers. The results of this experiment are shown as the right portion of Table 4.3.

We see that the resulting translational errors are again very high when we simply learn a static measurement covariance, but that we can still achieve reasonably low errors when learning the parameters of our IW prior. By incorporating the IW prior instead of learning a static measurement covariance, we decrease error from above 6 m to 0.2355 m. Compared to the error of 0.2335 m when there are no outliers in our test set (Table 4.1), we see an increase in error of only 0.002 m with the IW prior, which we believe is negligible. This experiment shows that we can indeed still benefit from the outlier rejection scheme that comes with using an IW prior, even when there are no outliers in our training set.

4.4 Summary and Conclusions

We presented our ESGVI and EM parameter learning framework that incorporates an IW prior to learn a varying measurement covariance. We estimate the measurement covariance as part of the state (e-step) and learn the parameters of the corresponding IW prior as part of the m-step. We presented experimental results on a lidar localization dataset and demonstrated training without the groundtruth trajectory. The methodology and experimental results in this chapter appeared in Wong et al. (2020b) as a second-author contribution.

In summary, the contributions of this chapter are:

1. A methodology for estimating measurement covariance by using an IW prior in a EM framework.

2. Experimental results on a lidar localization dataset that demonstrates learning a varying measurement covariance without the groundtruth trajectory. We also show that the resulting covariance model is robust to measurement outliers during both training and testing.

In the next chapter, we will revisit the measurement covariance problem using a feature-dependent model and also demonstrate how we can also model measurement bias.

Chapter 5

Learning Feature-Dependent Models

Due to nonidealities in the real world, our sensors can produce biased measurements with respect to our sensor models that we may have to calibrate ourselves. The uncertainty of the measurements can also differ from the manufacturer's specification and may vary depending on the operation environment. In the previous chapter, we presented a reactive approach for estimating the measurement covariance as part of the state. In this chapter, we explore training feature-dependent regression models for both measurement bias and covariance, which is a viable solution in applications where these quantities are predictable from a training dataset.

When tuning a sensor for bias and covariance, one could carefully devise a calibration experiment with high-quality groundtruth in a controlled setting. However, experiments in a controlled setting may not accurately reflect the data in a practical setting. The quality of the data may even change over time (possibly due to hardware degradation). This suggests that we should instead be able to readily train our models from practical data collection experiments such that the training data will be in-distribution and up-to-date. Our method ideally should not be reliant on high-quality groundtruth as a training signal since it may be difficult to obtain in an uncontrolled setting.

We propose to train sensor models using the likelihood of the observed data from the sensors themselves. We can accomplish this using our ESGVI and EM parameter learning framework. By choosing to optimize the data likelihood, we may avoid the requirement of a groundtruth training signal, which we demonstrate in a lidar odometry experiment using real-world data.

In summary, the main contribution of this chapter is the methodology for learning feature-dependent regression models for measurement bias and covariance. While the usage of feature-dependent models is not new, the novelty of our work is the application of training without the groundtruth trajectory using EM.

Section 5.2 presents our methodology for learning feature-dependent regression models for the measurement bias and covariance. For the e-step, we can apply a suitable approximation for ESGVI, as covered in Chapter 3. We present the specifics of the m-step in this section, which can differ if the learnable parameters have a linear or nonlinear relationship in our regression models. In Section 5.3 and Section 5.4, we apply our methodology to learning the bias and covariance for Doppler velocity measurements from a FMCW lidar for application to lidar odometry. We present experimental results on two datasets we collected: the Boreas dataset and the Aeva HQ dataset. Since we are learning a measurement bias, we cannot reliably train for the bias and covariance models using EM with the Doppler measurements alone[1]. Instead, we can run EM using a slower *ICP-based* odometry method (i.e., effectively using the pointcloud alignment as a supervision signal) and apply the learned models to a faster *Correspondence-Free (CF)-based* odometry method after training.

The work on the ICP-based odometry method using Doppler measurements was published as a second-author contribution to the IEEE Robotics and Automation Letters (Wu et al., 2023). As a second author, contributions were mainly made to the methodology and analysis of the experiments. Doppler bias and noise models were not applied or learned in this publication.

The methodology of the fast CF-based odometry method was presented as a first-author contribution at the International Conference on Intelligent Robots and Systems (Yoon et al., 2023), which we presented as a fast and low-cost alternative to ICP. We demonstrated supervised training for Doppler bias in this publication using the groundtruth trajectory, and we did not learn the Doppler measurement noise.

The work in this chapter extends upon the work of the existing two publications by (i) also incorporating a learned noise model for the Doppler measurements, (ii) demonstrating unsupervised training using EM, and (iii) applying the CF-based method to multiple FMCW lidar sensors in simulation and on real data.

5.1 Related Work

In this section, we provide a literature review on measurement biases and FMCW lidar odometry. For a literature review on measurement covariances, please see the previous chapter.

[1]We further discuss this later in the chapter

5.1.1 Measurement Biases

It is no surprise that unaccounted biases will degrade state estimation performance. In some situations, however, the exact source of the bias may not be well understood or difficult to model. In such cases, it may be favourable to model a systematic bias as a correction to the output of the estimator (Hidalgo-Carrió et al., 2017; Peretroukhin and Kelly, 2017; Tang et al., 2018). Hidalgo-Carrió et al. (2017) model the residual error for wheel odometry using Gaussian process (GP) regression. A similar approach was later applied to correct for the bias in lidar odometry by Tang et al. (2018). Instead of GP regression, Peretroukhin and Kelly (2017) learn a correction for stereo visual odometry using a CNN.

We can apply a more direct approach if the source of the bias can be narrowed down to a residual error within a well-understood model. One such way is online estimation of the bias as part of the state (Barfoot, 2024, §5, p. 183). This approach is often applied for handling biases in IMU data since it can be modelled as a slowly changing quantity that evolves as a random walk (Barfoot, 2024). Instead of a random walk, a more sophisticated process model for IMU bias can be learned using a deep network for better robustness to prolonged visual tracking failure (Buchanan et al., 2022).

If the dependencies of the biases are known and do not change over time (or change relatively slowly over time), a bias model can instead be calibrated offline. For example, several works investigated bias compensation for lidar range measurements that is dependent on measurement incidence angle and/or distance (Laconte et al., 2019; Kümmerle and Kühner, 2020; Agishev et al., 2023). Laconte et al. (2019) propose an approximated closed-form formula based on a model of the lidar return waveform. Kümmerle and Kühner (2020) use a polynomial model that is dependent only on the incidence angle. Agishev et al. (2023) propose using a polynomial model that is dependent on both incidence angle and distance, but also train their model parameters from data in a self-supervised manner.

In our experiments in Section 5.4 using FMCW lidar, we require a method for modelling bias in the Doppler velocity measurements. While our use case is not exactly the same as the range bias compensation discussed above, we can follow a similar approach and calibrate for the Doppler bias offline using a regression model that is an additive correction to our known measurement model. Our approach for modelling the bias is most similar to Agishev et al. (2023) since we will use a regression model with polynomial basis functions on the input features. However, the focus in our work is not on the particular model we chose, but rather the way we are proposing to train it. We demonstrate training using our parameter learning framework with ESGVI and EM, which can be viewed as self-supervised training since we will not be using the groundtruth trajectory as a training signal.

5.1.2 FMCW Lidar Odometry

Our odometry experiments in Section 5.4 involve a relatively new type of lidar sensor referred to as Frequency-Modulated Continuous Wave (FMCW) lidar, a lidar sensor that additionally measures per-return relative radial velocities via the Doppler effect. We present a brief overview of the current literature regarding FMCW and lidar odometry in this subsection.

Recent works have already demonstrated how FMCW lidars are beneficial for improving odometry. Hexsel et al. (2022) incorporate Doppler measurements into ICP to improve estimation in difficult, geometrically degenerate locations. Shortly after, Wu et al. (2023) improved upon their work by using a continuous-time estimator, not requiring motion compensation as a preprocessing step. A major bottleneck in lidar odometry is data association due to (i) the vast amount of data, and (ii) the need for iterative data association. With the introduction of Doppler measurements from FMCW lidars, we recently proposed a more efficient odometry method that avoids data association entirely (Yoon et al., 2023). We present our latest experimental results in Section 5.4 with models for measurement bias and noise trained without groundtruth using our parameter learning framework.

Lidar odometry algorithms have proven to be highly accurate in nominal conditions, but will struggle to perform in geometrically degenerate environments (e.g., long tunnels, barren landscapes). Using IMU data is a way of handling these difficult scenarios, with the added benefit of being able to use the IMU to motion-compensate pointclouds as a preprocessing step. Loosely coupled methods may only use the IMU data for motion compensation (Palieri et al., 2020), but can also fuse the pose estimates from pointcloud alignment with IMU data downstream (Tagliabue et al., 2021). Zhao et al. (2021) implement an odometry estimator for each sensor modality, where each estimator uses the outputs of the other estimators as additional observations. Chen et al. (2023) combine their pose estimates from ICP with IMU data using a hierarchical geometric observer. Tightly coupled methods incorporate IMU data into the pointcloud alignment optimization directly, which has been shown using an iterated extended Kalman filter (Qin et al., 2020; Xu et al., 2022) and factor graph optimization over a sliding window (Ye et al., 2019; Shan et al., 2020).

Our work differs from existing lidar-inertial methods in both motivation and implementation. Our fast odometry method does not require data association by using the Doppler measurements of a FMCW lidar. Our motivation for using IMU data is to compensate for the degrees of freedom not observable from the Doppler measurements of a single FMCW lidar. We only require gyroscope data (i.e., angular velocities) and exclude the accelerometer[2], permitting a linear continuous-time formulation. We do not require pre-integration of the gyroscope data (Forster et al., 2016), and instead efficiently incorporate data at their exact measurement times.

FMCW is a relatively new technology for lidar, but not for radar. Radar, in contrast to FMCW lidar, returns two-dimensional (2D) detections (azimuth and range). Similar to our proposed method, Kellner et al. (2013) estimate vehicle motion using radar Doppler measurements without data association. Using a single radar, they estimate a 2-degrees-of-freedom (DOF) vehicle velocity (forward velocity and yaw rotation) by applying a kinematic constraint on the lateral velocity to be zero. Using multiple radars allowed them to estimate a 3-DOF vehicle velocity (Kellner et al., 2014). As radars produce 2D data in lesser quantities compared to lidar, Kellner et al. (2013, 2014) limited their experiments to driven sequences that were a few hundred meters in length. Our FMCW lidar produces thousands of three-dimensional (3D) measurements at a frame rate of 10Hz. We take advantage of the richer data by efficiently applying them in a continuous-time linear estimator, and demonstrate reasonably accurate odometry over several kilometers.

Kramer et al. (2020) estimate the motion of a handheld sensor rig by combining radar Doppler measurements and IMU data. Park et al. (2021) estimate for 6-DOF motion by first estimating the 3D translational velocities, then loosely coupling them with IMU data in a factor graph optimization. We similarly use IMU data to help estimate 6-DOF motion. However, we only use the gyroscope data to keep the estimator linear and efficient.

5.2 Learning Feature-Dependent Bias and Covariance Models

In Section 3.3, we presented a brief sketch of how parameters may be learned using our ESGVI framework and EM. An example for a constant measurement covariance, $\mathbf{W}^{-1}$,

[2]The gravity vector would require a nonlinear estimator for orientation.

was shown for the following factor:

$$\phi(\mathbf{x}, \mathbf{y}|\mathbf{W}) = \frac{1}{2}\sum_{k=1}^{K}\Big(\mathbf{e}_k(\mathbf{x}_k, \mathbf{y}_k)^T \mathbf{W}\, \mathbf{e}_k(\mathbf{x}_k, \mathbf{y}_k) - \ln(|\mathbf{W}|)\Big)\,, \tag{5.1}$$

which resulted in the following m-step update:

$$\mathbf{W}^{-1} = \frac{1}{K}\sum_{k=1}^{K}\mathbb{E}_{q_k}\left[\mathbf{e}_k(\mathbf{x}_k, \mathbf{y}_k)\mathbf{e}_k(\mathbf{x}_k, \mathbf{y}_k)^T\right]. \tag{5.2}$$

In this section, we extend this idea to modelling and learning feature-dependent measurement covariances, as well as the measurement biases (if they exist). In practice, how noisy (or how biased) a sensor is can vary and depend on factors at the time of application. A constant model for the measurement covariance (or measurement bias) in such situations may not be sufficient. We can model the variation of the measurement covariance explicitly by using a regression model that depends on a handcrafted input feature. The input feature will be defined with known quantities that we expect the covariance to be correlated with and will be application-dependent. For example, later in Section 5.4 we will use features such as the lidar measurement range (magnitude) and intensity (signal strength). We can similarly model a regression model for a measurement bias (if needed).

As shown in Section 3.3, we will assume our measurement model factors in the following way:

$$\begin{aligned}\phi(\mathbf{x}, \mathbf{y}|\boldsymbol{\theta}) &= \sum_{k=1}^{K}\phi_k(\mathbf{x}_k, \mathbf{y}_k|\boldsymbol{\theta}) \\ &= \sum_{k=1}^{K}\frac{1}{2}\Big((\mathbf{e}_k + \mathbf{b}_{\boldsymbol{\psi}_{bk}|\theta_b})^T \mathbf{W}_{\boldsymbol{\psi}_{Wk}|\theta_W}(\mathbf{e}_k + \mathbf{b}_{\boldsymbol{\psi}_{bk}|\theta_b}) - \ln(|\mathbf{W}_{\boldsymbol{\psi}_{Wk}|\theta_W}|)\Big)\,,\end{aligned} \tag{5.3}$$

where we now partition our parameters, $\boldsymbol{\theta} = \{\boldsymbol{\theta}_W, \boldsymbol{\theta}_b\}$, to distinguish between the covariance parameters, $\boldsymbol{\theta}_W$, and the bias parameters, $\boldsymbol{\theta}_b$. We choose to model the feature-dependent (inverse) measurement covariance using a regression model,

$$\mathbf{W}_{\boldsymbol{\psi}_{Wk}|\theta_W} = \mathbf{W}(\boldsymbol{\psi}_{Wk}|\boldsymbol{\theta}_W), \tag{5.4}$$

that depends on an input feature, $\boldsymbol{\psi}_{Wk}$. Note that the regression model for the (inverse) covariance must be constrained to be a positive-definite matrix. Similarly, we choose to model the feature-dependent bias using a regression model,

$$\mathbf{b}_{\psi_{bk}|\theta_b} = \mathbf{b}(\boldsymbol{\psi}_{bk}|\boldsymbol{\theta}_b), \tag{5.5}$$

that depends on an input feature, $\boldsymbol{\psi}_{bk}$. Note that we have introduced our bias model as an additive term to the measurement error model. Another possibility is to incorporate a bias model inside the measurement model, but how that should be done is model-dependent.

We will apply EM to jointly optimize the latent state and our model parameters. For the e-step, we will hold the parameters fixed and optimize the state using an appropriate approximation of ESGVI. For the m-step, there are a couple cases to consider. Let us first look at the derivative of our loss with respect to our parameters again,

$$\frac{\partial V(q|\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \frac{\partial}{\partial \boldsymbol{\theta}} \mathbb{E}_q[\phi(\mathbf{x}, \mathbf{y}|\boldsymbol{\theta})] = \frac{\partial}{\partial \boldsymbol{\theta}} \mathbb{E}_q \left[\sum_{k=1}^{K} \phi_k(\mathbf{x}_k, \mathbf{y}_k|\boldsymbol{\theta}) \right] = \sum_{k=1}^{K} \mathbb{E}_{q_k} \left[\frac{\partial}{\partial \boldsymbol{\theta}} \phi_k(\mathbf{x}_k, \mathbf{y}_k|\boldsymbol{\theta}) \right]. \tag{5.6}$$

There are special cases where we can simply set the derivative to zero for an extremum and isolate for the parameter in closed form. One such case was the constant covariance example from Section 3.3. Another case worth noting is when our bias regression model, $\mathbf{b}(\boldsymbol{\psi}_b|\boldsymbol{\theta}_b)$, is a linear function with respect to the parameters, $\boldsymbol{\theta}_b$.

5.2.1 Special Case: Linear Bias Regression

Focusing on the bias regression, we will assume for now that the (inverse) covariance, $\mathbf{W}_k$, is known. We will assume the bias regression model is a linear function with respect to the parameters, $\mathbf{b}(\boldsymbol{\psi}_{bk}|\boldsymbol{\theta}_b) = \boldsymbol{\Theta}_b\boldsymbol{\psi}_{bk}$, where $\boldsymbol{\theta}_b = \text{vec}(\boldsymbol{\Theta}_b)$ (i.e., the vec$(\cdot)$ operator converts a matrix to a vector by stacking its columns). Note that we define our input feature, $\boldsymbol{\psi}_b$, with a scalar 1 as its last element to incorporate a constant addition term. We will next manipulate it using vector notation for a more convenient form with respect to the parameters,

$$\mathbf{b}(\boldsymbol{\psi}_{bk}|\boldsymbol{\theta}_b) = \boldsymbol{\Theta}_b\boldsymbol{\psi}_{bk} = \text{vec}(\boldsymbol{\Theta}_b\boldsymbol{\psi}_{bk}) = \underbrace{(\boldsymbol{\psi}_{bk}^T \otimes \mathbf{1})}_{\mathbf{U}_{bk}} \underbrace{\text{vec}(\boldsymbol{\Theta}_b)}_{\boldsymbol{\theta}_b}, \tag{5.7}$$

where the $\otimes$ operator is the Kronecker product and we used the identity (Magnus and Neudecker, 2019)

$$\text{vec}(\mathbf{ABC}) = (\mathbf{C}^T \otimes \mathbf{A})\text{vec}(\mathbf{B}). \tag{5.8}$$

Substituting our model back into (5.3) and our derivative expression in (5.6) with respect to $\boldsymbol{\theta}_b$,

$$\frac{\partial V(q|\boldsymbol{\theta}_b)}{\partial \boldsymbol{\theta}_b} = \sum_{k=1}^{K} \mathbb{E}_{q_k}\left[\mathbf{U}_{bk}^T \mathbf{W}_k (\mathbf{e}_k + \mathbf{U}_{bk}\boldsymbol{\theta}_b)\right]. \tag{5.9}$$

While our bias regression model is linear with respect to our parameters, our measurement error model, $\mathbf{e}_k$, can be nonlinear with respect to our state. In order to evaluate the expectation, we take a linear approximation for the state by using a perturbation, $\mathbf{x}_k = \bar{\mathbf{x}}_k + \delta\mathbf{x}_k$:

$$\frac{\partial V(q|\boldsymbol{\theta}_b)}{\partial \boldsymbol{\theta}_b} \approx \sum_{k=1}^{K} \mathbb{E}_{q_k}\left[\mathbf{U}_{bk}^T \mathbf{W}_k (\bar{\mathbf{e}}_k + \mathbf{E}_k \delta\mathbf{x}_k + \mathbf{U}_{bk}\boldsymbol{\theta}_b)\right] \tag{5.10}$$

$$= \sum_{k=1}^{K} \left(\mathbf{U}_{bk}^T \mathbf{W}_k (\bar{\mathbf{e}}_k + \mathbf{U}_{bk}\boldsymbol{\theta}_b) + \underbrace{\mathbb{E}_{q_k}\left[\mathbf{U}_{bk}^T \mathbf{W}_k \mathbf{E}_k \delta\mathbf{x}_k\right]}_{=\mathbf{0}} \right), \tag{5.11}$$

where we used the Jacobian of our error function, $\mathbf{E}_k$. Setting the derivative to zero for an extremum, we get a linear system for $\boldsymbol{\theta}_b$ that we can use as our m-step update:

$$\left(\sum_{k=1}^{K} \mathbf{U}_{bk}^T \mathbf{W}_k \mathbf{U}_{bk} \right) \boldsymbol{\theta}_b = \sum_{k=1}^{K} \mathbf{U}_{bk}^T \mathbf{W}_k \mathbf{e}_k. \tag{5.12}$$

If we were to combine this m-step for $\boldsymbol{\theta}_b$ with the m-step for a constant measurement covariance from Section 3.3, we can pick an order for which update to do first within each m-step. Since EM requires iteration anyway, we expect the choice of order to not be significant.

5.2.2 General Case: Nonlinear Regression

The more general case is when our feature-dependent model for covariance or bias is nonlinear with respect to the model parameters. In such cases, we cannot simply set the derivative in (5.6) to zero and isolate for a solution in closed form. Instead, we can formulate our m-step using gradient-descent optimization:

$$\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i)} - \eta \left. \frac{\partial V(q|\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} \right|_{\boldsymbol{\theta}^{(i)}}, \tag{5.13}$$

where i is the iteration index and η is a tunable learning rate.

A practical aspect to EM is that we do not need to perform the m-step to convergence before alternating back to the e-step. This is known as *Generalized EM* (GEM) (Bishop, 2006). For clarity, the GEM process will be as follows:

1. e-step: hold our parameters, $\boldsymbol{\theta}$, fixed and optimize our loss, $V(q|\boldsymbol{\theta})$, for the current best estimate of our approximate posterior, $q(\mathbf{x})$.
2. m-step: hold our posterior estimate, $q(\mathbf{x})$, fixed and run gradient descent for a short number of iterations to update our parameters, $\boldsymbol{\theta}$.
3. alternate between steps 1 and 2 until convergence.

There are other practical considerations for the m-step that can be helpful depending on the application. If the training dataset is very large, we can substitute standard gradient descent with a stochastic variant that approximates the gradient with a subset (mini-batch) of the dataset. Depending on the complexity of the model, we may wish to simply run an off-the-shelf optimizer that has built-in improvements to gradient descent, such as momentum and automated learning rate tuning.

5.3 Estimator Methodology: FMCW Lidar Odometry

Before diving into the experiments on FMCW lidar odometry, we dedicate this section to present the methodologies for the two estimators that we will use. The first *CF-based* method estimates for only the 6-DOF vehicle velocity using Doppler measurements from a FMCW lidar and gyroscope measurements from an IMU. Removing the vehicle pose from the estimator formulation results in a fast, linear odometry estimator that does not require data correspondence, which is often a computational bottleneck in lidar odometry. However, the Doppler measurements are biased and we cannot reliably train for the bias calibration using EM with the Doppler and gyroscope measurements alone. Instead, we can run EM using a slower *ICP-based* odometry method (i.e., effectively using the pointcloud alignment as a supervision signal) and apply the learned models to the faster CF odometry at test time.

The CF-based odometry methodology was presented in our publication, Yoon et al. (2023). We extend this work in this thesis by training our regression models for measurement noise and bias using EM. The slower ICP-based odometry methodology was presented in a second-author publication, Wu et al. (2023), which we slightly improve upon here by incorporating gyro measurements from an IMU.

5.3.1 Correspondence-Free Odometry

We formulate our odometry as a linear, continuous-time batch estimation for the 6-DOF vehicle body velocity, $\boldsymbol{\varpi}(t) = \boldsymbol{\varpi}_v^{iv}(t) \in \mathbb{R}^6$. As we will later see, this is possible because our measurement and motion prior models are linear with respect to the vehicle velocity. A continuous-time formulation allows each measurement to be applied at their exact measurement times efficiently. The relative pose estimate can be computed via numerical integration afterwards if pose is the desired output. For the rest of the methodology, we drop the superscripts and subscripts denoting the frames of our state for convenience, but note that they are defined between the inertial frame, i, and vehicle frame, v (i.e., not the sensor frame).

The proposed method is extremely lightweight as the estimation problem is linear and we do not require data association for the lidar data. For generality we present our methodology as batch state estimation. We can then adapt our method for online application by incrementally marginalizing out all past velocity state variables and only keeping around the latest state (see Gaussian marginalization in Section 2.1). This marginalization will result in a linear, continuous-time filter that handles measurements asynchronously. There is no need to keep additional states in a window (i.e., sliding window) since the problem is linear.

Motion Prior Factors

We apply the continuous-time estimation framework of Barfoot et al. (2014) to estimate the trajectory as a GP. We model our vehicle velocity prior as WNOA,

$$\dot{\boldsymbol{\varpi}}(t) = \mathbf{w}(t), \quad \mathbf{w}(t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{Q}_c \delta(t - t')), \tag{5.14}$$

where $\boldsymbol{\varpi}(t)$ is our continuous-time body velocity state, and $\mathbf{w}(t)$ is a (stationary) zero-mean GP with a power spectral density matrix, $\mathbf{Q}_c$. The underlying representation of $\boldsymbol{\varpi}(t)$ is a discrete trajectory, $\boldsymbol{\varpi}_k = \boldsymbol{\varpi}(t_k)$, for $k = 0, 1, \ldots, K$. For convenience, we set $\boldsymbol{\varpi}_k$ to correspond to the end time of the k^{th} lidar frame[3], where each lidar frame is the acquisition of lidar data over the time period of 100 ms[4]. The advantage of this GP approach is the separation of the low-frequency state times that we estimate from the high-frequency measurement times of our Doppler and gyroscope measurements. The

[3]We set $\mathbf{x}_0$ to be at the start of the first frame.

[4]While 100 ms is an application-specific value, 3D lidar sensors commonly operate at a rate of 10Hz, including the lidars we use in our experiments. A *frame* of a lidar is the accumulation of data over the full (horizontal) field of view of the sensor.

WNOA motion prior factor between state times $k-1$ and k is then

$$\phi_{\text{wnoa},k} = \frac{1}{2}\mathbf{e}_{\text{wnoa},k}^T \mathbf{Q}_k^{-1} \mathbf{e}_{\text{wnoa},k}, \tag{5.15a}$$

$$\mathbf{e}_{\text{wnoa},k} = \boldsymbol{\varpi}_k - \boldsymbol{\varpi}_{k-1}, \tag{5.15b}$$

where the covariance $\mathbf{Q}_k = (t_k - t_{k-1})\mathbf{Q}_c$. This prior conveniently results in linear interpolation in time for our velocity-only state (Barfoot, 2024):

$$\boldsymbol{\varpi}(\tau) = (1-\alpha)\boldsymbol{\varpi}_k + \alpha\boldsymbol{\varpi}_{k+1}, \quad \alpha = \frac{\tau - t_k}{t_{k+1} - t_k} \in [0,1], \tag{5.16}$$

with $\tau \in [t_k, t_{k+1}]$.

In addition to the WNOA motion prior, we found it beneficial to incorporate vehicle kinematics by penalizing velocities in specific dimensions. We center our vehicle frame at the rear axle of the vehicle and orient it such that the x-axis points forward, y-axis points left, and z-axis points up. The vehicle kinematics factor is

$$\phi_{\text{kin},k} = \frac{1}{2}\mathbf{e}_{\text{kin},k}^T \mathbf{Q}_{\text{kin}}^{-1} \mathbf{e}_{\text{kin},k}, \tag{5.17a}$$

$$\mathbf{e}_{\text{kin},k} = \mathbf{H}\boldsymbol{\varpi}_k, \tag{5.17b}$$

where a constant $\mathbf{H}$ extracts the dimensions of interest (e.g., the lateral, vertical, roll, and pitch dimensions) and $\mathbf{Q}_{\text{kin}}$ is the corresponding covariance matrix.

Measurements Factors

The (scalar) Doppler velocity measurement factor is

$$\phi_{\text{dv}} = \frac{1}{2}e_{\text{dv}}^2/\sigma_{\text{dv}}^2, \tag{5.18a}$$

$$e_{\text{dv}} = y_{\text{dv}} - \begin{bmatrix}\hat{\mathbf{q}}^T & \mathbf{0}^T\end{bmatrix} \boldsymbol{\mathcal{T}}_{sv}\boldsymbol{\varpi}(\tau) + b. \tag{5.18b}$$

The Doppler velocity error function is defined using a projection of the vehicle velocity state onto the radial direction of the scalar Doppler measurement, y_{dv}. We slightly abuse notation[5] to define the unit direction vector of the query point:

$$\hat{\mathbf{q}} = \frac{\mathbf{D}\mathbf{q}}{(\mathbf{q}^T\mathbf{D}^T\mathbf{D}\mathbf{q})^{\frac{1}{2}}}, \tag{5.19}$$

[5] $\hat{\mathbf{q}} \in \mathbb{R}^3$ is the unit direction vector of $\mathbf{q} \in \mathbb{R}^4$ ignoring the fourth homogeneous element.

where $\mathbf{D}$ is a selection matrix that removes the fourth homogeneous element. See Appendix A.1 for a derivation of this measurement model. We query our continuous-time trajectory for the vehicle velocity corresponding to the measurement time, $\boldsymbol{\varpi}(\tau)$, which is transformed using the adjoint of the sensor-vehicle extrinsic transformation, $\boldsymbol{\mathcal{T}}_{sv} = \text{ad}(\mathbf{T}_{sv})$. We model the measurement variance, σ^2_{dv}, and measurement bias, b, as feature-dependent regression models, which we further discuss in Section 5.4.1.

The gyroscope measurement factor is

$$\phi_{\text{gyro}} = \frac{1}{2}\mathbf{e}_{\text{gyro}}^T \mathbf{R}_{\text{gyro}}^{-1} \mathbf{e}_{\text{gyro}}, \tag{5.20a}$$

$$\mathbf{e}_{\text{gyro}} = \mathbf{y}_{\text{gyro}} - \mathbf{C}_{sv}\mathbf{D}_{\text{rot}}\boldsymbol{\varpi}(\tau), \tag{5.20b}$$

where $\mathbf{y}_{\text{gyro}} \in \mathbb{R}^3$ is a gyroscope measurement, $\mathbf{R}_{\text{gyro}}$ is the corresponding measurement covariance, $\mathbf{C}_{sv}$ is a known sensor-vehicle extrinsic rotation matrix, and $\mathbf{D}_{\text{rot}} \in \mathbb{R}^{3\times 6}$ is a constant selection matrix that removes the first three elements (translation components) of $\boldsymbol{\varpi}(\tau)$. We verified empirically that the bias of the gyroscopes used in our experiments is reasonably constant throughout the duration of a data sequence, therefore we found it sufficient to apply an offline calibration for a constant bias. Ideally we would incorporate the gyro bias as part of our state, but the estimation problem would then be ill-posed if only relying on the Doppler measurements (i.e., no ICP). We plan on revisiting this problem in future work.

Objective Function

Since all our models are linear with respect to our state, MAP is equivalent to ESGVI (Barfoot et al., 2020). Compiling all the factors together, our MAP objective function is

$$J = \sum_k \left(\phi_{\text{wnoa},k} + \phi_{\text{kin},k}\right) + \sum_i \phi_{\text{dv},i} + \sum_j \phi_{\text{gyro},j}. \tag{5.21}$$

We can simply differentiate this objective with respect to the state and set it to zero for an optimum.

Figure 5.1 illustrates the states and factors in our online problem. For the latest lidar frame, k, the Doppler measurements are incorporated at their measurement times using our continuous-time interpolation scheme. The gyroscope measurements are similarly handled at their respective measurement times. For online application, we incrementally marginalize out older state variables, $\boldsymbol{\varpi}_i$, where $i < k$, and estimate the latest velocity, $\boldsymbol{\varpi}_k$ (i.e., a filter implementation).

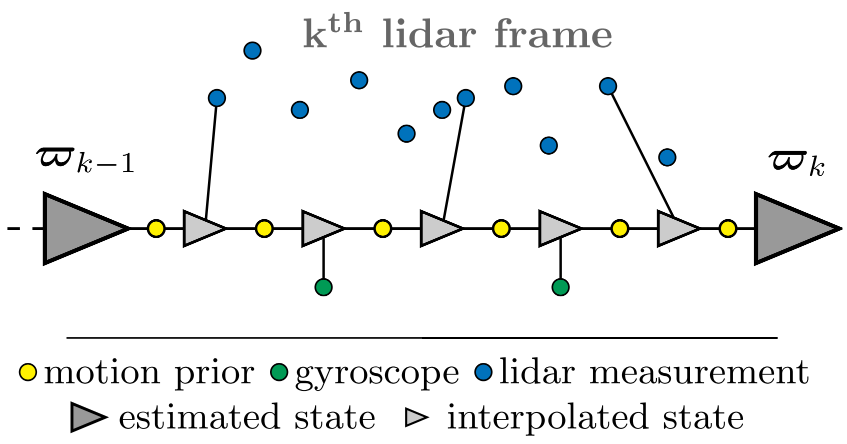


Figure 5.1: An illustration of the factors involved in the online velocity estimation problem. The Doppler and gyroscope measurements are applied at their exact measurement times using our continuous-time interpolation scheme. There is no data association for the lidar measurements. We marginalize out past state variables (i.e., $\boldsymbol{\varpi}_{k-1}$), resulting in a filter for the latest velocity, $\boldsymbol{\varpi}_k$.

Numerical Integration for Pose

Using our estimates of the vehicle velocity, we can approximate the relative pose by numerically sampling $\boldsymbol{\varpi}(t)$ with a small timestep, $\triangle t$, and creating a chain of $SE(3)$ matrices spanning the time interval (Anderson and Barfoot, 2013):

$$\begin{aligned}\mathbf{T}_{k,k-1} \approx \prod_{s=1}^{\overset{\curvearrowleft}{S}} \exp(\triangle t\, \boldsymbol{\varpi}(t_{k-1} + s\triangle t)^\wedge) = \exp(\triangle t\, \boldsymbol{\varpi}(t_k)^\wedge) \dots \\ \times \exp(\triangle t\, \boldsymbol{\varpi}(t_{k-1} + 3\triangle t)^\wedge) \exp(\triangle t\, \boldsymbol{\varpi}(t_{k-1} + 2\triangle t)^\wedge) \exp(\triangle t\, \boldsymbol{\varpi}(t_{k-1} + \triangle t)^\wedge),\end{aligned} \tag{5.22}$$

where $\exp(\cdot)$ is the exponential map, $\boldsymbol{\varpi}(t)$ is the vehicle velocity interpolated between boundary velocities $\boldsymbol{\varpi}_{k-1}$ and $\boldsymbol{\varpi}_k$, and we have divided the time interval by S steps, making $\triangle t = (t_k - t_{k-1})/S$. Note that we use $\overset{\curvearrowleft}{(\cdot)}$ in our matrix product notation to indicate the multiplication order, i.e., each subsequent matrix is multiplied on the left side of the accumulated matrix product. In practice, our vehicle pose estimates drift while the vehicle is stationary (e.g., no movement due to traffic) due to noise and residual error in the velocity estimates. We propose mitigating this drift by checking a tolerance (we use 0.03m/s) on the forward translational speed estimate. If the speed of a boundary estimate is less than the tolerance, we set that boundary estimate $\boldsymbol{\varpi}_i = \mathbf{0}$ before interpolation to mitigate pose drift.

Outlier Rejection

Outliers in the Doppler measurements are often caused by erroneous reflections and moving objects in the environment. Fortunately, each lidar frame is dense and, in practice, the majority of the measurements are of the stationary environment (inliers). Similar to Kellner et al. (2013), we found RANSAC (Fischler and Bolles, 1981) to be a suitable method for outlier filtering. We classify between inliers and outliers using a constant threshold (0.2m/s) on the Doppler error model (5.18b). We run RANSAC on each lidar frame independently. We assume the vehicle velocity is constant throughout each frame and solve for it using two randomly sampled Doppler measurements. The solve is made observable by enforcing vehicle kinematic constraints, i.e., we solve for a 2-DOF velocity[6] $\boldsymbol{\varpi}^T = [v \quad 0 \quad 0 \quad 0 \quad 0 \quad \omega]$. We can optionally include the gyroscope measurements in the RANSAC solve. In practice, 20 iterations of RANSAC was sufficient for each lidar frame.

5.3.2 ICP-based Odometry

The detailed methodology of our ICP-based odometry with Doppler measurements can be found in our publication (Wu et al., 2023), where we referred to the method as STEAM-DICP. We provide a brief summary of the methodology here. In contrast to the publication, we slightly improve upon the original method in this thesis by adding gyroscope measurements from an IMU and learned feature-dependent models for the Doppler variance and bias.

We again formulate our odometry as a continuous-time estimation problem using GP regression in order to efficiently handle high-frequency measurements at their exact timestamps. What differs now from the CF-based method is the introduction of the vehicle pose. We now represent our continuous-time trajectory as $\mathbf{x}(t) = \{\mathbf{T}(t), \boldsymbol{\varpi}(t)\}$, where $\mathbf{T}(t) = \mathbf{T}_{vi}(t) \in SE(3)$ is our continuous-time pose trajectory between the *vehicle* and stationary *inertial* frames, and $\boldsymbol{\varpi}(t) = \boldsymbol{\varpi}_v^{iv}(t) \in \mathbb{R}^6$ is the same vehicle body velocity we estimated in the CF-based method. We again drop the superscripts and subscripts denoting the frames of our state variables for convenience, but note that they are defined between the inertial frame and vehicle frame (and not the sensor frame).

With the introduction of the vehicle pose to our state, which is a variable that belongs to $SE(3)$, we need a different motion prior for our GP approach. Following Anderson and Barfoot (2015), we apply a WNOA motion prior in $SE(3)$ in a piecewise fashion

[6]Recall that we orient our vehicle frame such that the x-axis points forward, y-axis points left, and z-axis points up.

across an underlying discrete trajectory of states, $\mathbf{x}_k = \mathbf{x}(t_k)$, $k = 0, 1, \ldots, K$. We previously applied this motion prior factor for our experiments in Chapter 4 (see details in Section 4.3). For convenience, we set our discrete states $\mathbf{x}_k$ to correspond to the end time of the k^{th} lidar frame. The time period of the k^{th} frame lies between the states $\mathbf{x}_{k-1}$ and $\mathbf{x}_k$.

ICP Factor

Unique to the ICP-based method is the point-to-plane factor,

$$\phi_{\text{p2p}} = \frac{1}{2} e_{\text{p2p}}^2 / \sigma_{\text{p2p}}^2, \tag{5.23a}$$

$$e_{\text{p2p}} = \hat{\mathbf{n}}^T \mathbf{D}(\mathbf{p} - \mathbf{T}(\tau)^{-1} \mathbf{T}_{sv}^{-1} \mathbf{q}). \tag{5.23b}$$

The point-to-plane error function is defined as a nearest-neighbour correspondence between a homogeneous query point, $\mathbf{q}$, to a homogeneous map point, $\mathbf{p}$. We query our continuous-time trajectory for the pose variable corresponding to the (measurement) time of the query point, $\mathbf{T}(\tau)$. The unit surface normal, $\hat{\mathbf{n}}$, is a computed estimate of the surface normal at point $\mathbf{p}$ in the map. The sensor-vehicle extrinsic transformation, $\mathbf{T}_{sv}$, is a known calibration and we use a constant selection matrix, $\mathbf{D}$, to remove the fourth homogeneous element. The factor, ϕ_{p2p}, is expressed as the usual squared cost term, but in practice we replace the squared cost with a Cauchy robust cost term for outlier rejection. We set $\sigma_{\text{p2p}} = 0.1\text{m}$.

Sliding-Window ICP

The combination of the $SE(3)$ WNOA motion prior and the point-to-plane factor in a sliding-window implementation results in our continuous-time ICP-based odometry method. We can additionally include the kinematic prior on the vehicle velocity, the Doppler measurement, and the gyroscope measurement factors from the CF-based estimator to improve performance. We optimize for the state using a MAP objective (more specifically, Gauss-Newton) rather than applying a higher-order approximation of ESGVI since (i) the posterior distribution in practice will be highly concentrated because of the dense, highly-accurate[7] range measurements, (ii) a derivative-free implementation is not necessary, and (iii) ICP is already computationally demanding. We also set the window size to 1 since ICP performance is not a priority in our experiments in this thesis.

[7]Manufacturer specification of the noise on range measurements for commercial 3D lidar sensors are often in the ballpark of a few centimeters.

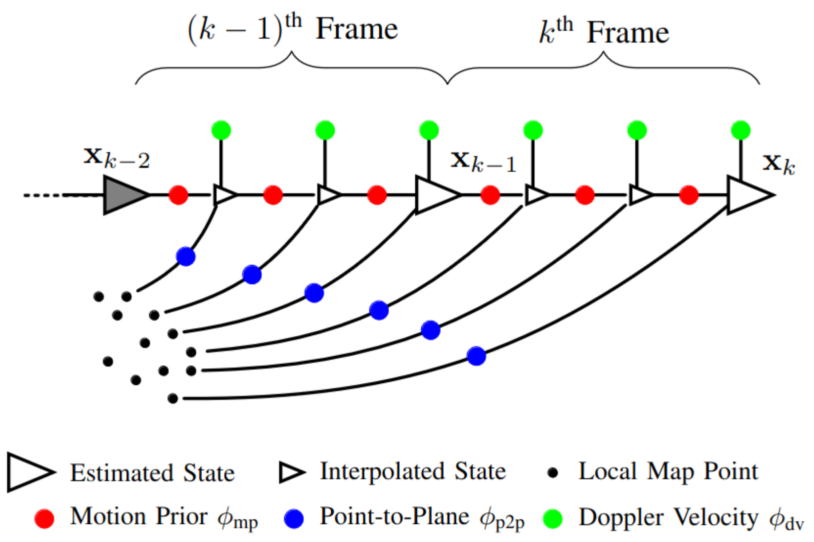

$(k-1)^{\text{th}}$ Frame
k^{th} Frame
$\mathbf{x}_{k-2}$
$\mathbf{x}_{k-1}$
$\mathbf{x}_k$
Estimated State
Interpolated State
Local Map Point
Motion Prior ϕ_{mp}
Point-to-Plane ϕ_{p2p}
Doppler Velocity ϕ_{dv}

Figure 5.2: This diagram shows the states and factors involved in our sliding-window estimation with an example window size of two. We query the trajectory states at the acquisition time of each keypoint in each lidar frame. The motion prior factor ϕ_{mp} connects neighbouring trajectory states. The point-to-plane factor ϕ_{p2p} requires local map information while the Doppler velocity factor ϕ_{dv} does not. Not shown in the diagram is an optional gyroscope measurement, which we can directly apply on the velocity component of the state via interpolation at the gyroscope measurement times.

Figure 5.2 shows a factor graph illustration of our ICP-based method. A local point-cloud map is incrementally built and updated as we slide the window. We follow the approach of Dellenbach et al. (2022) for the front-end processing of the pointcloud data. Keypoints are extracted from each lidar frame via voxel grid downsampling. We use a grid size of 1.5 m and keep one random point in each voxel. Our local map is a pointcloud accumulated from the most recent frames and cropped to be within 100 m of the latest estimate of the vehicle after each frame update. The local map is stored in a sparse voxel grid structure with a 1 m grid size and a maximum of 20 points per voxel. We use point-to-plane ICP for frame-to-map registration. Each frame point is associated with a map point via nearest-neighbour association, and the corresponding plane normal is computed by applying Principle Component Analysis (PCA) to the 20 closest neighbours of the associated map point.

5.4 Experiment: FMCW Lidar Odometry

We demonstrate learning covariance[8] and bias for Doppler velocity measurements from FMCW lidar sensors without supervision from the groundtruth trajectory. We apply the feature-dependent approach discussed in Section 5.2 and show experimental results on two datasets: the Boreas FMCW dataset and the Aeva HQ dataset.

5.4.1 Models and Training

The FMCW lidar sensors that we use in our experiments have biased Doppler measurements that we found in practice to be reasonably stationary, making our proposed offline training (calibration) approach to be effective. We apply the feature-dependent approach discussed in Section 5.2 to model the bias, as well as the measurement uncertainty (variance), which we discuss in this subsection.

Unseen from our perspective of the lidar data is an algorithm within the sensor that extracts the range and relative velocity from the raw lidar signal. Any inaccuracy in the assumptions made for this algorithm may result in a residual bias that we observe during operation of the sensor. We found the Doppler measurements to be biased with a large dependence on the range (distance). A few other input features that were found to be useful in our experiments are: the bearing (azimuth and elevation) of the measurement, the intensity of the measurement, and the speed of the vehicle. As it is not the focus of our work, we relegate the explanation of a speed input feature to Appendix B.1.

Our priority in the design of our regression model is computational efficiency, therefore our choice of model for bias compensation is a linear regression model (i.e., a model that is linear with respect to its learnable parameters). In order to strike a balance between model complexity and computational efficiency, we adopt a lookup table approach. Input features that we expect to be highly nonlinear with respect to our learnable parameters we choose to discretize into bins that can be accessed with constant-time complexity, where each bin contains its own regression model. Input features that we expect to have a linear or mildly nonlinear relationship with our learnable parameters are handled using polynomial basis functions. The input feature we discretize into a lookup table is the measurement bearing, i.e., we partition each lidar frame into (approximately) uniform bins of azimuth and elevation. We can express our model as

$$b^{ae}(\boldsymbol{\psi}_{bk}|\boldsymbol{\theta}_b^{ae}) = \boldsymbol{\psi}_{bk}^{aeT}\boldsymbol{\theta}_b^{ae}, \tag{5.24}$$

[8]Technically variance since the Doppler measurements are scalar and we assume they are statistically independent from one another.

as we have shown in Section 5.2.1, but now written for a scalar bias output that is indexed by the azimuth bin, a, and elevation bin, e. We apply the polynomial basis functions within the feature vector, $\boldsymbol{\psi}_{bk}^{ae}$. For example, using a polynomial order of up to 2, our input feature is

$$\boldsymbol{\psi}_{bk}^{aeT} = \begin{bmatrix} r^{ae} & (r^{ae})^2 & i^{ae} & (i^{ae})^2 & s & s^2 & 1 \end{bmatrix}, \tag{5.25}$$

where r^{ae} is the measurement range, i^{ae} is the measurement intensity, and s is the speed input feature.

Partitioning the lidar data by azimuth and elevation also has the effect of downsampling since we keep one measurement per azimuth-elevation bin. We uniformly partition along the azimuth by 0.2°. The measurements are already partitioned in elevation by the scan pattern of the sensor, which produces 80 or 64 horizontal sweeps[9]. This downsampling effectively projects each lidar frame into a 80×500 or 64×500 image (see Figure 5.3 for an example).

We model our variance regression model similarly to our bias model, but with a constraint to output positive values. We follow Liu et al. (2018) and use the exponential function to constrain our model output to be positive. We model our inverse-variance model as

$$W(\boldsymbol{\psi}_{Wk}^{ae} | \boldsymbol{\theta}_W) = \exp\left({\boldsymbol{\psi}_{Wk}^{ae}}^T \boldsymbol{\theta}_W \right), \tag{5.26}$$

which, in contrast to the bias regression model, is a nonlinear model with respect to our learnable parameters, $\boldsymbol{\theta}_W$. Note the intentional omission of the azimuth-elevation superscripts, ae, on our model $W(\cdot)$. This indicates a single regression model that is not dependent on azimuth and elevation, as we found it unnecessary for the measurement variance in practice. We additionally incorporate an input feature specifically for the variance, which we refer to as the *pseudo-variance*. For clarity and in the interest of space, please see Appendix B.2 for the details of this input feature.

In summary, our Doppler measurement model is

$$y_k^{ae} = \begin{bmatrix} \hat{\mathbf{q}}^T & \mathbf{0}^T \end{bmatrix} \boldsymbol{\mathcal{T}}_{sv} \boldsymbol{\varpi}(\tau) + b^{ae}(\boldsymbol{\psi}_{bk} | \boldsymbol{\theta}_b^{ae}) + n, \quad n \sim \mathcal{N}(0, W(\boldsymbol{\psi}_{Wk}^{ae} | \boldsymbol{\theta}_W)^{-1}). \tag{5.27}$$

Our Doppler factor when training using the m-step is

$$\phi_k^{ae} = \frac{1}{2} \left(y_k^{ae} - \begin{bmatrix} \hat{\mathbf{q}}^T & \mathbf{0}^T \end{bmatrix} \boldsymbol{\mathcal{T}}_{sv} \boldsymbol{\varpi}(\tau) + b^{ae}(\boldsymbol{\psi}_{bk} | \boldsymbol{\theta}_b^{ae}) \right)^2 - \frac{\alpha}{2} \ln\left(W(\boldsymbol{\psi}_{Wk}^{ae} | \boldsymbol{\theta}_W) \right), \tag{5.28}$$

where we have additionally placed a hyperparameter, α, that can be used to scale the

[9]This is a configuration setting in the sensor hardware.

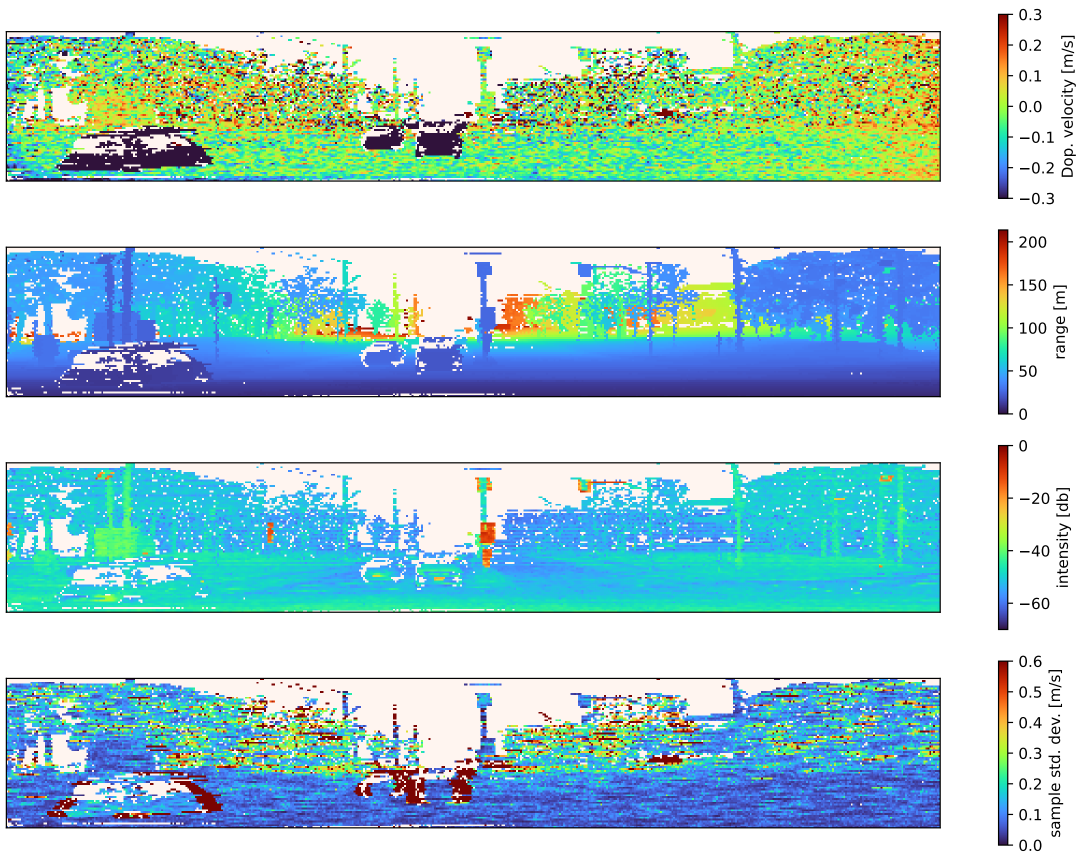


Figure 5.3: We partition the FMCW data into (approximately) uniform azimuth-elevation bins, effectively creating an image-space projection of the lidar data. The first row shows the biased Doppler velocity measurements, which noticeably exhibit more bias on certain parts of the image, e.g., near the left and right edges. The second row shows the range measurement, the third row shows the intensity, and the last row shows the square root of the pseudo-variance (pseudo-standard-deviation) input feature (see Appendix B.2 for details). The pseudo-variance shows higher values on uneven surfaces (e.g., foliage) and lower values for flat surfaces (e.g., road).

balance between the (first) error term and the (second) logarithm term. When $\alpha < 1$, we decrease the contribution of the logarithm term, which in practice results in a more conservative (larger) measurement variance. Due to unmodelled nonidealities, the default value of $\alpha = 1$ that we obtain from the negative-log of a Gaussian may result in overconfident estimates of the measurement uncertainty. These nonidealities could be from residual measurement bias that was not modelled by our method, time correlations in the measurement noise, and/or noise characteristics not modelled well by a Gaussian (e.g., a heavy-tailed noise distribution). In our experiments we set $\alpha = 0.1$, which we found to work well for us in practice.

We train our two models using EM. For the e-step, we apply our ICP-based odometry

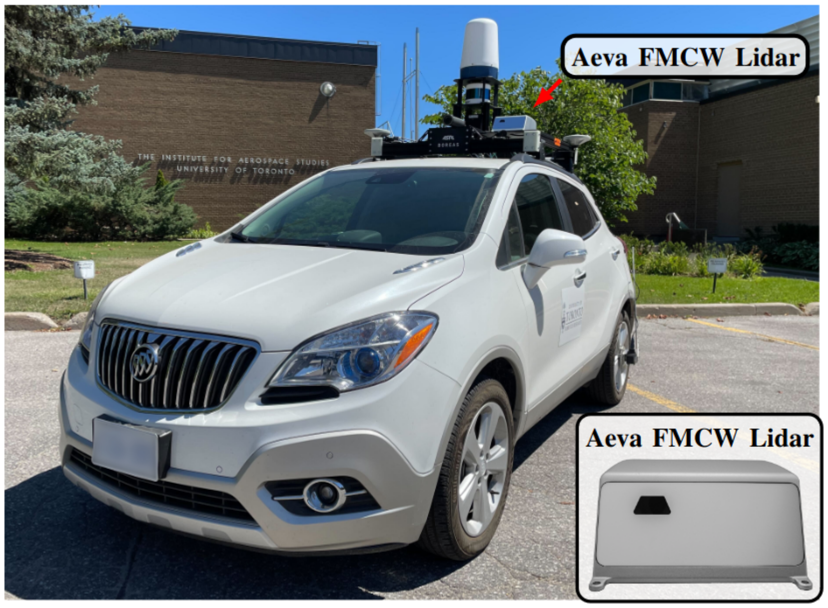


Figure 5.4: Our platform, *Boreas*, was previously used to collect the Boreas Multi-Season dataset (Burnett et al., 2023) and additionally equipped with an Aeva Aeries I FMCW Lidar for use in this work.

method, i.e., we take a Gauss-Newton (GN) approximation for ESGVI. For the m-step, we apply an off-the-shelf stochastic gradient optimizer. We use the Adam optimizer (Kingma and Ba, 2014) and optimize the parameters for a single epoch in each m-step (i.e., a full pass through the training dataset) before alternating back to the e-step. After training, we can replace the slower ICP-based odometry method with our faster CF-based odometry method for a trade-off between odometry performance and computational speed.

5.4.2 Datasets

We collected two datasets with FMCW lidar data on which we show experimental results in the following subsection. We provide a brief description of the datasets in this section.

Boreas FMCW Dataset

The Boreas FMCW dataset is a dataset collected using our data-collection vehicle at the Autonomous Space Robotics lab at the University of Toronto. An image of our data collection vehicle, *Boreas*, is shown in Figure 5.4. The vehicle was previously used for the Boreas Multi-Season dataset (Burnett et al., 2023), which does not include data collected with a FMCW lidar. Boreas was later equipped with an Aeva Aeries I FMCW

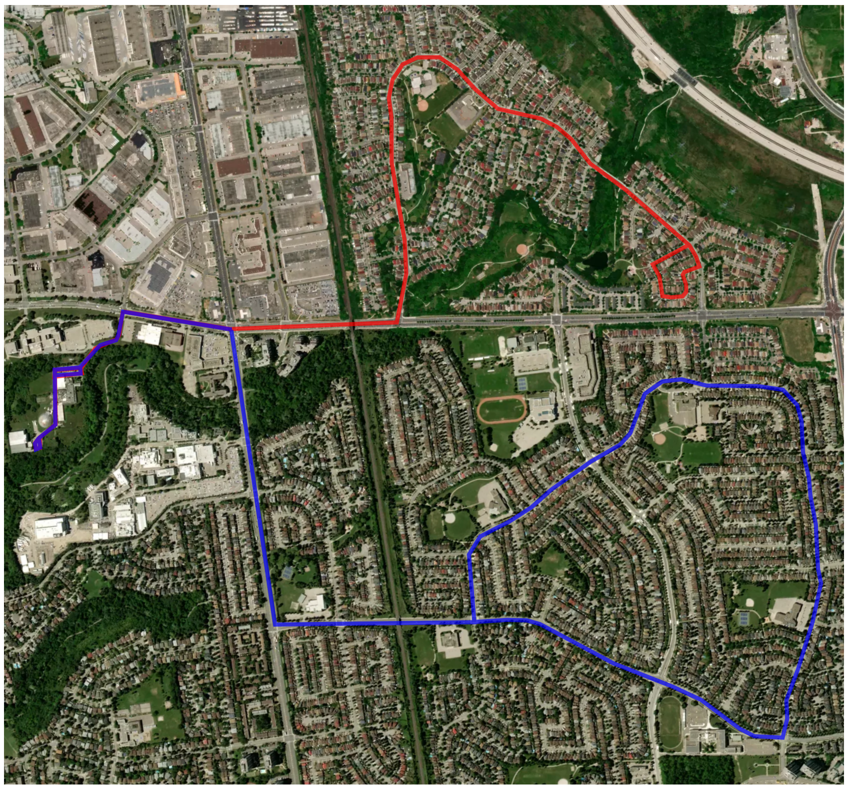


Figure 5.5: The two routes of the Boreas FMCW dataset in Toronto, Ontario, Canada. We have four sequences of the Glen Shields route (red) and one sequence of a different route collected in the same area (blue). Mapbox was used to generate this figure.

lidar sensor, which has a horizontal field-of-view of 120°, a vertical field-of-view of 30°, a 300 m maximum operating range, and a sampling rate of 10 Hz. The lidar includes a Bosch BM160 IMU, which we use for our experiments to get gyroscope measurements. We use the post-processed estimates from an Applanix POS LV as our groundtruth.

We collected five data sequences near the University of Toronto Institute for Aerospace Studies (UTIAS). Sequences 1 to 4 follow the *Glen Shields* route of the Boreas Multi-Season dataset (Burnett et al., 2023). Sequence 5 is a different route collected in the same area. Figure 5.5 shows the paths of the two routes overlaid on a map.

Aeva HQ Dataset

The Aeva HQ dataset is a dataset collected in Mountain View, CA where Aeva Technologies Inc. is located. The data collection vehicle was equipped with four Aeva Aeries II FMCW lidar sensors. Similar to the Aeries I, the Aeries II has a horizontal field-of-view of 120°, a vertical field-of-view of 30°, a 300 m maximum operating range, a sampling rate of 10 Hz, and a built-in gyroscope. Figure 5.6 shows images of the data collection vehicle and the four lidar sensors. As in the Boreas FMCW dataset, this vehicle was equipped with an Applanix POS LV for obtaining trajectory groundtruth.

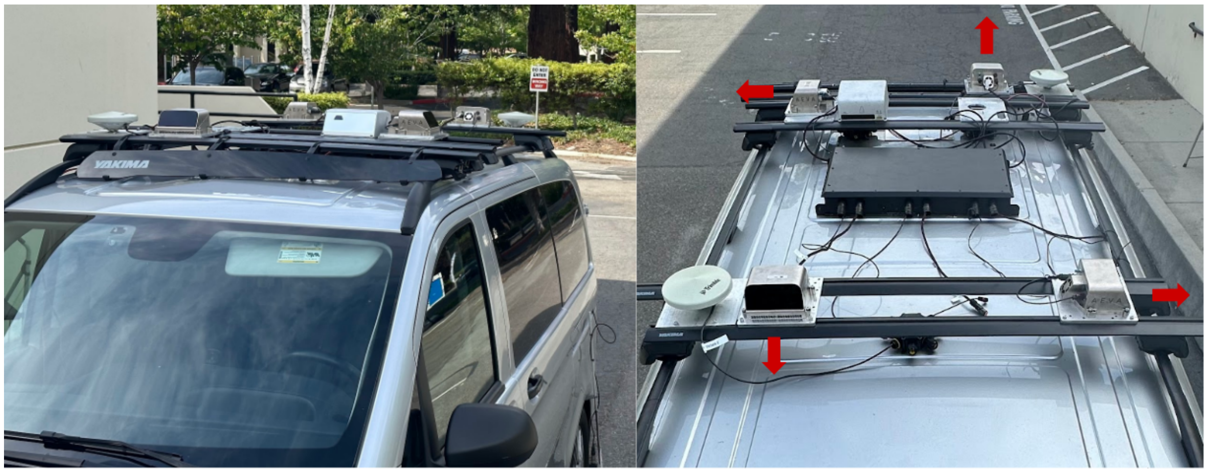


Figure 5.6: Images of the data collection platform used for the Aeva HQ dataset. The platform is equipped with four Aeries II FMCW lidar sensors and a Applanix POS LV for providing groundtruth. Each sensor is facing in a different direction (i.e., forward, left, right, and back), which is indicated by the red arrows in the right image.

We collected data over three different driving routes. There are three sequences for each of the three routes (A, B, and C), resulting in a total of nine sequences. Figure 5.7 illustrates the three routes overlaid on a map.

5.4.3 Results and Discussion

Recall that the intention of our training pipeline with EM is to use the slower ICP-based odometry to learn the parameters of our bias and noise models, which can be done offline. The main motivation for using EM is to be able to train without the groundtruth trajectory. After training, we can apply our learned models to our CF-based odometry for fast online estimation. The main objective of our experiments is to determine how well our EM-trained models will perform. Our baseline comparison will be the CF-based odometry performance when trained using high-quality groundtruth (i.e., supervised training).

Evaluation Metrics

Following existing work on vehicle odometry, we evaluate odometry using the KITTI odometry metric (Geiger et al., 2012), which averages errors over path lengths that vary from 100 m to 800 m in 100 m increments.

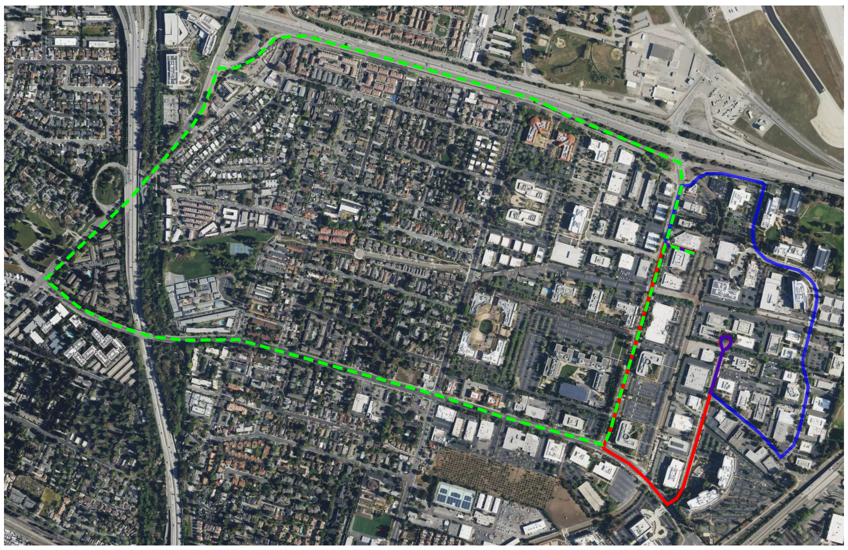


Figure 5.7: The routes of the Aeva HQ dataset: route A (red), route B (blue), and route C (green). We collected three sequences for each route, resulting in a total of nine sequences for this dataset. Mapbox was used to generate this figure.

As our CF-based odometry method directly estimates the vehicle velocity, we also evaluate the overall RMSE of the vehicle velocity in the translational and rotational dimensions. For the translational dimensions, we can compute the T-RMSE as

$$\text{T-RMSE} = \sqrt{\left(\frac{1}{K}\sum_{k} \|\mathbf{v}_k - \mathbf{v}_{\text{gt},k}\|^2\right)} \quad [\text{m/s}], \tag{5.29}$$

where $\mathbf{v}_k \in \mathbb{R}^3$ is the translational vehicle velocity estimate at lidar frame k and $\mathbf{v}_{\text{gt},k} \in \mathbb{R}^3$ is the corresponding translational vehicle velocity groundtruth. Similarly, we can compute the R-RMSE as

$$\text{R-RMSE} = \sqrt{\left(\frac{1}{K}\sum_{k} \|\boldsymbol{\omega}_k - \boldsymbol{\omega}_{\text{gt},k}\|^2\right)} \quad [\text{rad/s}], \tag{5.30}$$

where $\boldsymbol{\omega}_k \in \mathbb{R}^3$ is the rotational vehicle velocity estimate at lidar frame k and $\boldsymbol{\omega}_{\text{gt},k} \in \mathbb{R}^3$ is the corresponding rotational vehicle velocity groundtruth.

Boreas FMCW Dataset

We use the first two sequences as our training set. Within our training set, we leave out sequence 2 as our validation set. When training using EM, we stop training when the validation training loss does not continue to improve. When training supervised using the

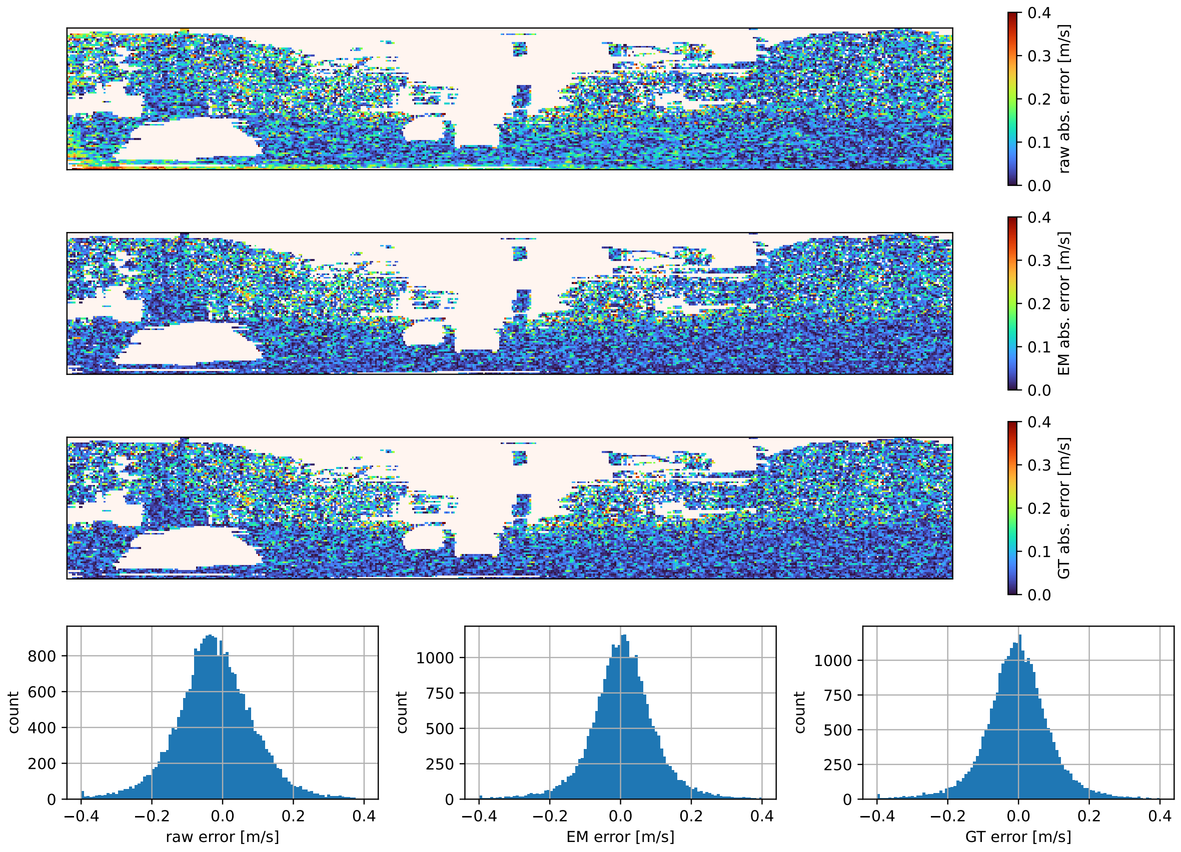


Figure 5.8: A comparison of the learned bias on the same stationary lidar frame shown in Figure 5.3. The first three rows show the Doppler error magnitude evaluated using the groundtruth for three configurations: (row 1) no compensation, (row 2) compensation using our regression model trained using EM (i.e., no groundtruth), and (row 3) compensation using our regression model trained using the groundtruth. The last row shows the error histograms corresponding to the same three configurations. The uncompensated histogram shows a clear bias, while the two compensated histograms appear reasonably centered about 0 m/s error. Outliers were removed in all examples using RANSAC.

groundtruth, which we use as a baseline comparison, we stop training when the KITTI odometry translation error stops improving on the validation sequence. Sequences 3, 4, and 5 are the test sequences.

Figure 5.8 shows an illustrative comparison of the learned measurement bias on the same stationary lidar frame shown in Figure 5.3. The first row shows the biased Doppler measurements. The second row and third row show the compensated Doppler measurements using our regression models trained using EM (i.e., trained without groundtruth) and trained using the groundtruth, respectively. The fourth row shows three error histograms of (i) the uncompensated Doppler error, (ii) the Doppler error compensated using EM, and (iii) the Doppler error compensated using the groundtruth. The uncom-

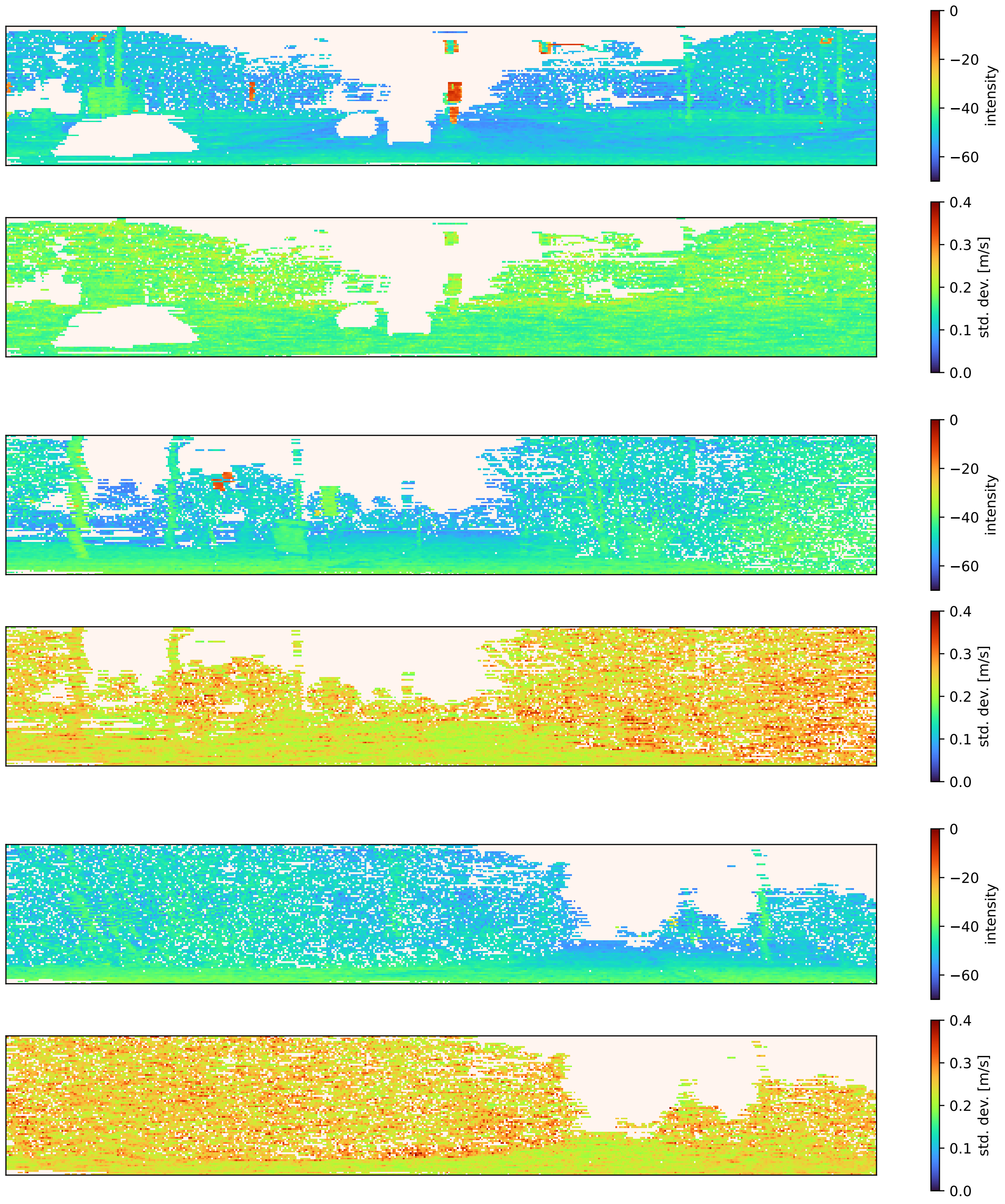


Figure 5.9: Three qualitative examples that demonstrate the Doppler uncertainty prediction on the Boreas FMCW dataset. The intensity values are shown along with the prediction to provide visual context of the scene. Doppler error tends to be higher on unstructured surfaces such as the foliage on trees and lower on flat surfaces such as the ground.

pensated histogram shows a clear bias, while the two compensated histograms appear similar and reasonably centered around 0 m/s error. Outliers were removed in all examples using RANSAC. Figure 5.9 shows qualitative examples of learning the measurement

Table 5.1: Quantitative performance evaluation on data sequences from the Boreas FMCW dataset. We show the translation values for the KITTI metric and the velocity RMSE. A detailed explanation of the various methods and analysis of the results are discussed in the main body text. The main outcome is that we can train our models without high-quality groundtruth using EM and perform reasonably close to training with the groundtruth. Training sequences are highlighted in grey and not counted towards the average. Best results in each of the three categories are in bold font.

	Translation Error [%]					
ICP-Based	01	02	03	04	05	**AVG**
No Dop.	0.269	0.266	0.265	0.256	0.320	0.281
w/ Dop. (GT)	**0.251**	**0.242**	**0.248**	**0.230**	**0.271**	**0.249**
w/ Dop. (EM)	0.287	0.258	0.265	0.253	0.293	0.270
Range Limited	21.78	21.66	45.44	58.97	24.50	42.97
CF-Based	01	02	03	04	05	**AVG**
Train GT	**1.122**	**1.017**	**1.066**	**1.057**	**0.927**	**1.017**
Train EM	1.132	1.062	1.124	1.109	0.998	1.077
Train EM*	1.179	1.114	1.141	1.132	1.048	1.107
No Train	1.465	1.382	1.439	1.423	1.343	1.402
	Translational Velocity Error [m/s]					
CF-Based	01	02	03	04	05	**AVG**
Train GT	**0.071**	**0.071**	**0.070**	**0.072**	**0.059**	**0.067**
Train EM	0.072	0.072	0.071	0.074	0.062	0.069
Train EM*	0.076	0.076	0.075	0.078	0.066	0.073
No Train	0.121	0.119	0.117	0.116	0.105	0.113

*EM trained using range limited ICP

uncertainty for three example lidar frames. Additional qualitative examples of bias and variance outputs are available in Appendix B.3.

Table 5.1 shows our quantitative results for the Boreas FMCW dataset. We present the KITTI translation metric for various settings of the ICP-based method and the CF-based method. We also show the T-RMSE for the CF-based methods. As expected, the ICP-based methods perform the best (when not artificially limited in range). Comparing among just the ICP-based methods, using Doppler measurements shows an improvement in performance over ICP on its own. The performance difference is most significant when the Doppler regression models are trained using the groundtruth. Note that evaluating the performance of ICP in comparison to our CF-based methods is not a primary objective of our experiments here, but it is reassuring to see that the results are consistent with our prior published works (Wu et al., 2023; Yoon et al., 2023).

The primary objective of our experiments is to compare the performance our Doppler bias and noise models when trained with the groundtruth (i.e., supervised training) and without the groundtruth trajectory (i.e., training using EM). In Table 5.1, we see that the performance of our CF-based odometry when trained using EM is reasonably similar to when trained using the groundtruth. Unsurprisingly, training with groundtruth performs slightly better, but that is attributed to the high quality of the Applanix POS LV groundtruth we use. Groundtruth of this quality is costly to obtain, so we believe the (small) performance difference we show here is a good outcome. We do not show the rotation metrics in Table 5.1 in the interest of space as the performance on those metrics is extremely similar. When trained with the groundtruth, our CF-based odometry achieves an average 0.39 °/(100 m) error for the KITTI rotation metric and 0.0239 rad/s for the R-RMSE. When trained with EM, our CF-based odometry achieves an average 0.40 °/(100 m) error for the KITTI rotation metric and 0.0239 rad/s for the R-RMSE. We show the performance without training as an additional baseline, which is the CF-based method without bias compensation and a constant Doppler noise model set to 0.5 m (standard deviation). Unsurprisingly, odometry without training performs the worst.

Given that the performance between training with groundtruth and training without groundtruth is very similar, and that ICP on its own is very accurate, a remaining question is whether there is any benefit in using EM. In the absence of our Applanix POS LV groundtruth, we could utilize the ICP estimates as a pseudo-groundtruth and still train our models in a supervised fashion (i.e., without EM). In order to demonstrate the effectiveness of EM, we designed an experiment where we artificially limited the measurements available for the ICP factors by setting a maximum range of 40 m. We show the evaluation of ICP using this artificial range limit in Table 5.1 as the 'Range Limited' entry under the 'ICP-based' section, which we see is extremely poor and clearly not suitable to use as a reference signal for supervised training. We then train our models using EM[10] with the range-limited ICP factors and show the evaluation of this model as 'Train EM*' in the same table. We view this artificial range limit on ICP as a way to simulate instances of difficult real-world scenarios where there is insufficient environment geometry for ICP to fully constrain the vehicle motion (e.g., barren landscapes or long tunnels). Therefore we show this experiment as a stress test to demonstrate that there is reasonable robustness to using training data from difficult environment settings.

[10]While we limited the range for the ICP factors during training, we do not apply this limit on the Doppler factors.

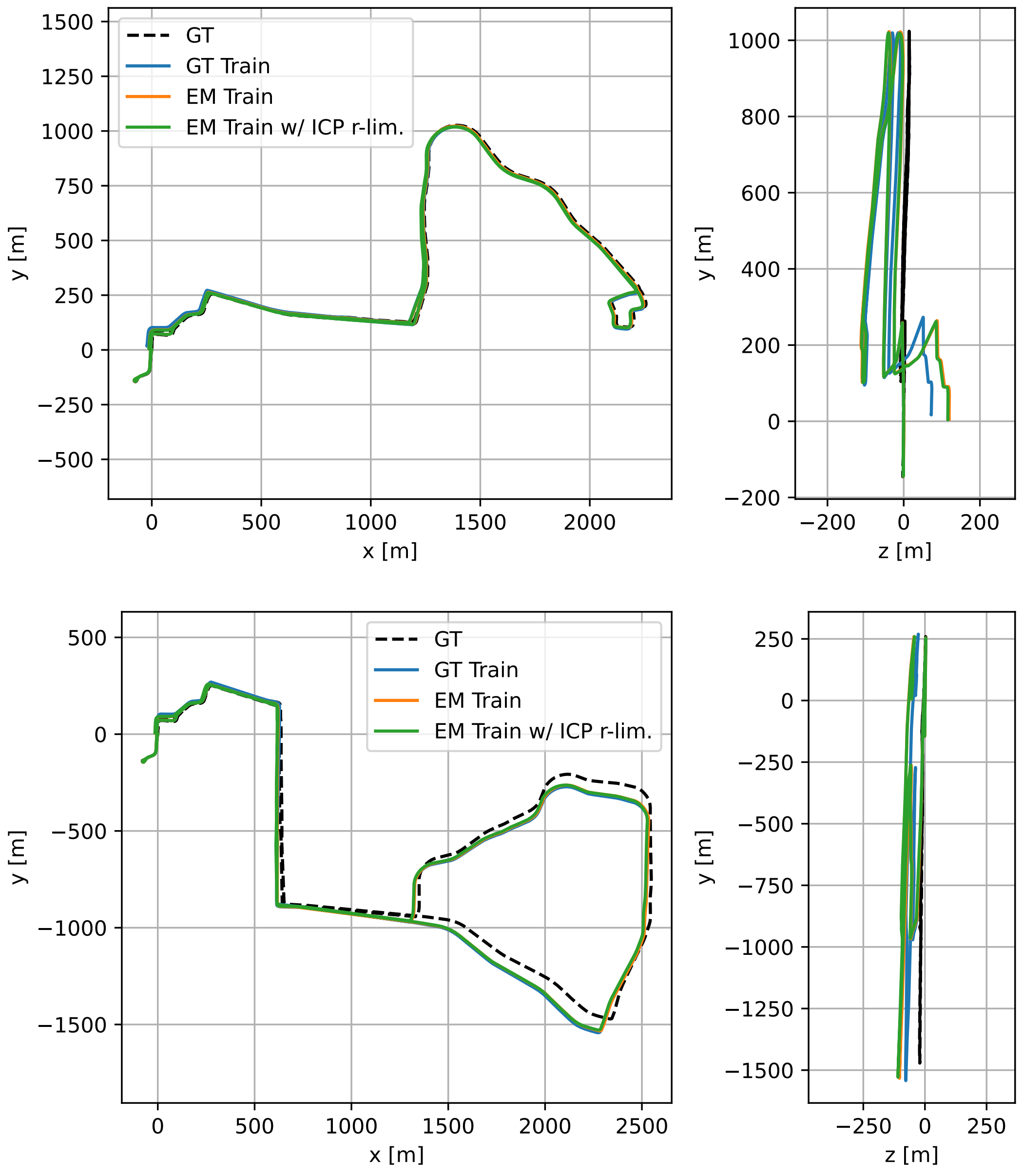


Figure 5.10: Plot of the CF-based odometry paths on the Boreas FMCW dataset for sequence 4 (first row) and sequence 5 (second row) using our learned Doppler bias and variance models. There is little difference between training supervised using the groundtruth and training unsupervised using EM, which is consistent with the results we see in Table 5.1.

Figure 5.10 shows a qualitative comparison of the odometry paths for the EM-trained models and GT-trained models. The first row shows the odometry on sequence 4, while the second row shows the odometry on sequence 5. Consistent with the quantitative results we see in Table 5.1, the plots of the paths are nearly identical to each other.

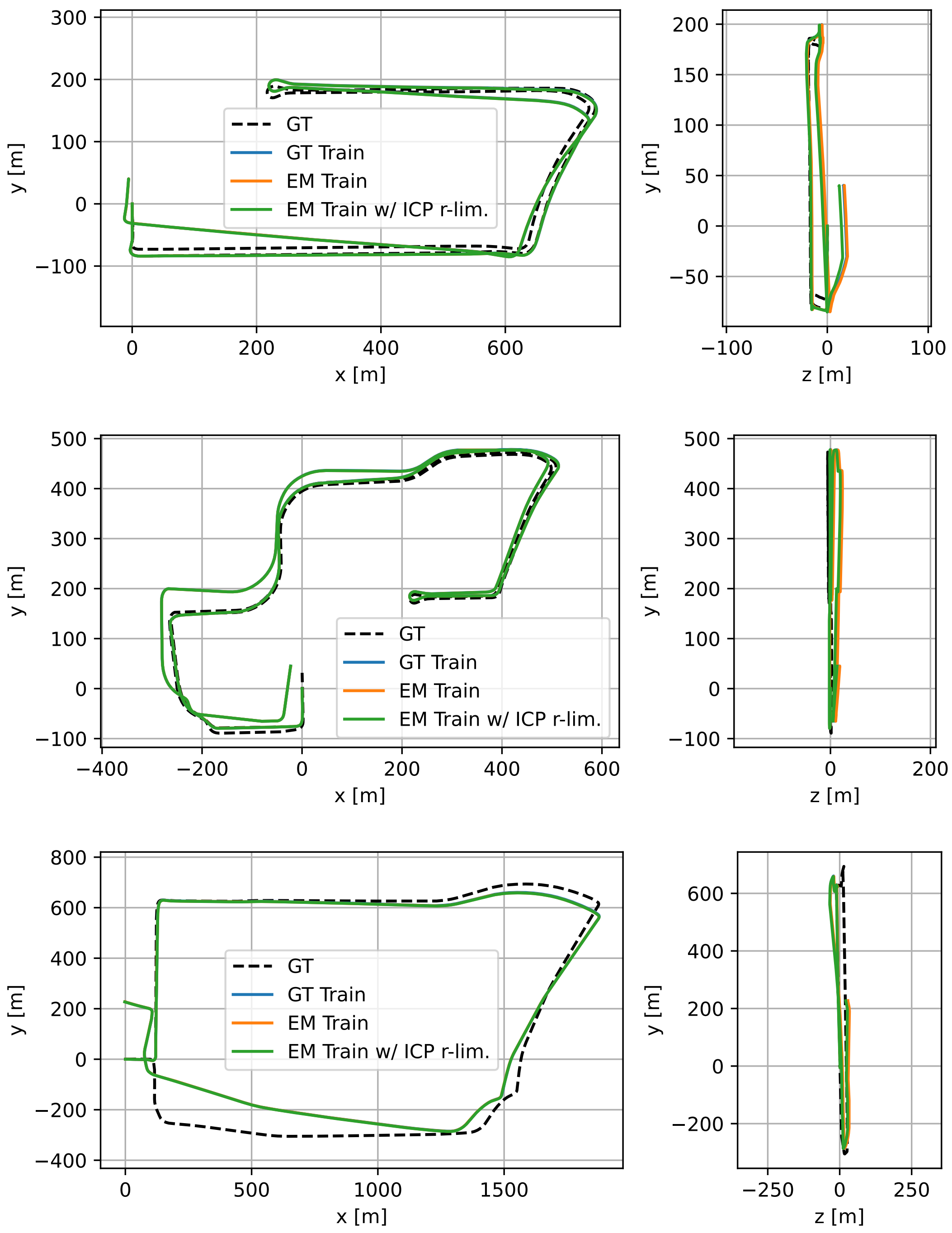


Figure 5.11: Plot of the CF-based odometry paths on the Aeva HQ dataset for sequence A3 (first row), sequence B3 (second row), and sequence C3 (third row) using our learned Doppler bias and variance models. There is little difference between training supervised using the groundtruth and training unsupervised using EM, which is consistent with the results we see in Table 5.2.

Aeva HQ Dataset

We repeat the experiments of the Boreas FMCW dataset on the Aeva HQ dataset using a single lidar and gyroscope[11] combination. For these single sensor experiments, we present results using the back facing lidar sensor (see Figure 5.6). Out of the nine available sequences, we use A1, A2, A3, and C1 as our training sequences. Out of these training sequences, we use A2 as our validation sequence.

Table 5.2 shows the quantitative results in the same format as Table 5.1. Our observations here are a close mirror to what we saw previously for the Boreas FMCW experiments and so we will keep the explanation brief. The performance of the CF-based odometry when trained using EM is reasonably similar to when trained using the groundtruth. Compared to our Boreas experiments, the results of EM training in this table are even closer, and sometimes favour EM over supervised training on some sequences. We repeat the artificial range limit experiment here to demonstrate the effectiveness of EM. Overall, it is reassuring to see that we can replicate our observations from the previous experiment on an entirely different dataset (i.e., different location, platform, sensor).

Figure 5.11 shows a qualitative comparison of the odometry paths for the EM-trained models and GT-trained models. The first row shows the odometry on sequence A3, the second row shows the odometry on sequence B3, and the third row shows the odometry on sequence C3. Consistent with the quantitative results we see in Table 5.2, the plots of the paths are nearly identical to each other.

The performance of the ICP-based methods is noticeably not as accurate as what we observed in the Boreas experiments, but to be fair, ICP performance was not a priority for us here. Given that EM still achieves odometry performance on par with supervised training, the lower performance of ICP does not seem to be a major contributing factor to the performance of our models trained using EM.

Unique to the Aeva HQ dataset is its multi-lidar sensor setup. We test the performance of using multiple FMCW lidar sensors on our CF-based odometry method. Despite each FMCW lidar sensor having a built-in gyroscope, we refer to the lidar as a separate sensor from its gyroscope for convenience in our experiment setup. In order to evaluate the effectiveness of including multiple sensors, we evaluate odometry using the following combinations: 1 lidar with 1 gyro, 1 lidar with 4 gyros, and 4 lidars with 4 gyros. Table 5.3 shows the quantitative results. As expected, using more sensors improves odometry performance. Including multiple gyroscopes is the biggest contributing factor to the improvement in performance. The '4 lidar with 4 gyro' combination does

[11] As a clarification, we use the gyroscope that corresponds to the chosen lidar.

Table 5.2: Quantitative performance evaluation on data sequences from the Aeva HQ Dataset. We show the translation values for the KITTI metric and the velocity RMSE. A detailed explanation of the various methods and analysis of the results are discussed in the main body text. The main outcome is that we can train our models without high-quality groundtruth using EM and perform reasonably close to training with the groundtruth. Training sequences are highlighted in grey and not counted towards the average. Best results in each of the three categories are in bold font.

	Translation Error [%]									
ICP-Based	A1	A2	A3	B1	B2	B3	C1	C2	C3	**AVG**
No Dop.	0.559	**0.534**	0.551	**0.552**	**0.531**	**0.543**	0.857	0.802	0.848	0.655
w/ Dop. (GT)	**0.539**	0.570	**0.544**	0.616	0.586	0.590	**0.723**	**0.662**	**0.723**	**0.635**
w/ Dop. (EM)	0.655	0.675	0.660	0.750	0.726	0.726	0.885	0.821	0.883	0.781
Range Limited	19.60	24.22	18.71	7.037	9.741	7.505	23.56	24.83	25.81	14.98
CF-Based	A1	A2	A3	B1	B2	B3	C1	C2	C3	**AVG**
Train GT	1.296	1.082	**0.860**	1.219	1.078	1.144	**1.323**	**0.830**	**1.373**	**1.129**
Train EM	1.335	1.100	0.882	1.229	1.075	**1.142**	1.341	0.841	1.391	1.136
Train EM*	**1.287**	**1.079**	0.881	**1.216**	**1.068**	1.144	1.372	0.857	1.424	1.142
No Train	1.776	1.415	1.386	1.718	1.485	1.582	1.900	1.554	1.941	1.656

	Translational Velocity Error [m/s]									
CF-Based	A1	A2	A3	B1	B2	B3	C1	C2	C3	**AVG**
Train GT	**0.054**	**0.050**	**0.048**	**0.051**	**0.051**	**0.048**	**0.055**	**0.063**	**0.057**	**0.054**
Train EM	0.056	0.052	0.050	0.053	0.053	0.051	0.059	0.067	0.059	0.056
Train EM*	0.056	0.053	0.050	0.054	0.054	0.052	0.059	0.067	0.059	0.057
No Train	0.123	0.124	0.117	0.122	0.123	0.117	0.135	0.142	0.133	0.127

*EM trained using range limited ICP

perform slightly better, but most of the improvement in performance is clearly from the inclusion of multiple gyroscopes. Comparing between training with groundtruth and without groundtruth, we again see very similar performance between the two. There are more instances where EM outperforms supervised training, but we do not believe the performance difference to be significant enough to judge one method to be better than the other.

Lastly, an outstanding question that remains is whether we can apply the CF-based odometry method without gyroscope measurements, i.e., by only using the Doppler velocity factors and our motion prior factors. In our published work, we derived an algebraic observability study and determined that the 6-DOF vehicle velocity is well-constrained

Table 5.3: Quantitative results on our Aeva HQ data sequences using multiple lidar sensors. As expected, performance improves with the inclusion of multiple sensors. Most of the gain in performance is due to using multiple gyroscopes. Training sequences are highlighted in grey and not counted towards the average. Best results in each of the four categories are in bold font.

	Translation Error [%]									
GT Train	A1	A2	A3	B1	B2	B3	C1	C2	C3	**AVG**
1 Lidar 1 Gyro	1.296	1.082	0.860	1.219	1.078	1.144	1.323	0.830	1.373	1.129
1 Lidar 4 Gyro	**0.796**	0.802	**0.699**	0.743	0.741	0.703	0.951	0.754	1.072	0.803
4 Lidar 4 Gyro	0.820	**0.756**	0.765	**0.700**	**0.695**	**0.659**	**0.867**	**0.685**	**0.993**	**0.746**
EM Train	A1	A2	A3	B1	B2	B3	C1	C2	C3	**AVG**
1 Lidar 1 Gyro	1.335	1.100	0.882	1.229	1.075	1.142	1.341	0.841	1.391	1.136
1 Lidar 4 Gyro	**0.800**	0.799	**0.676**	0.709	0.717	**0.683**	0.925	0.731	1.037	0.775
4 Lidar 4 Gyro	0.858	**0.742**	0.711	**0.685**	**0.694**	0.689	**0.808**	**0.675**	**0.918**	**0.732**
	Translational Velocity Error [m/s]									
GT Train	A1	A2	A3	B1	B2	B3	C1	C2	C3	**AVG**
1 Lidar 1 Gyro	0.054	0.050	0.048	0.051	0.051	0.048	0.055	0.063	0.057	0.054
1 Lidar 4 Gyro	0.053	0.050	0.049	0.050	0.049	0.048	0.054	0.061	0.055	0.053
4 Lidar 4 Gyro	**0.050**	**0.047**	**0.045**	**0.043**	**0.046**	**0.045**	**0.052**	**0.055**	**0.049**	**0.047**
EM Train	A1	A2	A3	B1	B2	B3	C1	C2	C3	**AVG**
1 Lidar 1 Gyro	0.056	0.052	0.050	0.053	0.053	0.051	0.059	0.067	0.059	0.056
1 Lidar 4 Gyro	**0.055**	0.051	0.050	0.050	**0.049**	**0.048**	**0.055**	0.063	0.056	**0.053**
4 Lidar 4 Gyro	**0.055**	**0.047**	**0.046**	**0.049**	0.050	0.050	0.056	**0.061**	**0.053**	**0.053**

without gyroscope measurements if we have at least three FMCW lidar sensors (Yoon et al., 2023), which we show in Appendix A.2. As a preliminary test, we evaluate our CF-based odometry method using all four lidar sensors without gyroscope measurements.

Unfortunately our experiments show that, while the vehicle velocity is fully constrained, the performance of odometry is very poor. On average, we obtain 4.808 % translation error and 0.18 °/(100 m) rotation error on the test sequences using the KITTI odometry metrics. The average RMSE errors for velocity are 0.069 m/s and 0.031 rad/s, which seem comparable to when we used gyroscope measurements, but we believe this is due to our data sequences mostly being composed of driving in a straight motion. The velocity errors are largest while turning, which causes significant drift that is reflected in the poor KITTI odometry metrics. We suspect that the poor performance is due to

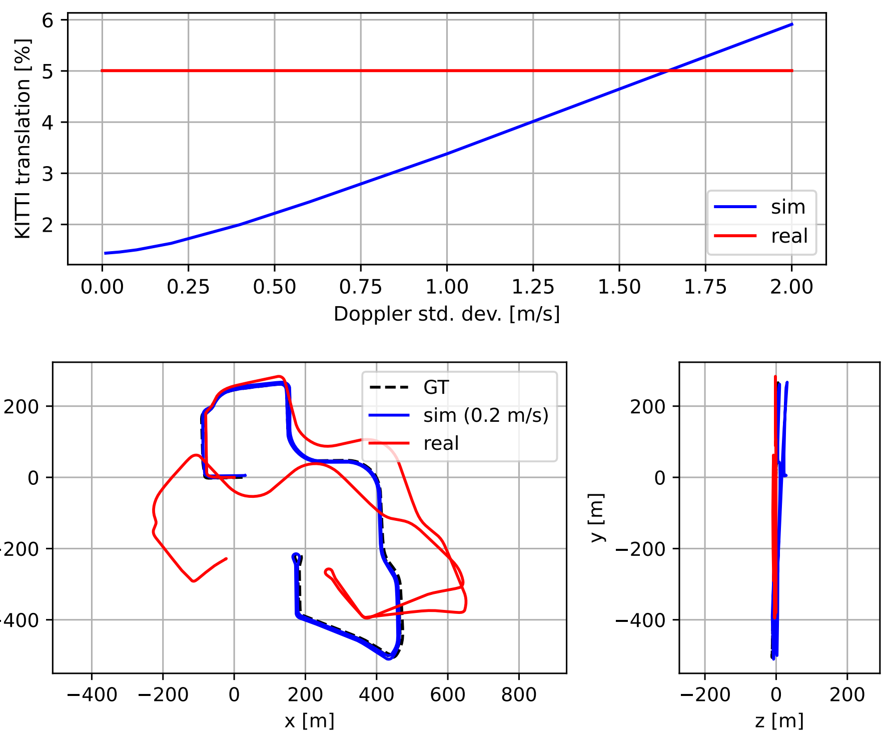


Figure 5.12: (top) A simulated sensitivity study for lidar-only CF-based odometry (no gyroscopes) using four FMCW lidars. We simulate measurements by replacing the real measurements using the groundtruth, the measurement model, and zero-mean Gaussian noise. The plot shows the translation error using the KITTI odometry metric as we increase the simulated measurement noise. For comparison, our result on the real data is shown as the (red) horizontal line. (bottom) The corresponding plots of odometry for the simulated result using 0.2 m/s standard deviation and real data.

unmodelled biases that remain even after our learned bias compensation. We chose a linear regression model using handcrafted features to favour computation over accuracy, but evidently the achievable accuracy using our relatively simple model falls short when we do not use gyroscope measurements.

We additionally demonstrate the performance of odometry by substituting the real Doppler measurements with simulated (unbiased) measurements using our Doppler measurement model and groundtruth. Figure 5.12 (top) shows a sensitivity study where we vary the standard deviation of the simulated noise on sequence B1 and plot the KITTI translation error. The sensor realistically returns Doppler measurements with a standard deviation of approximately 0.1 m/s (see an example histogram of errors in Figure 5.8), but we do not match the performance on real data until we increase the simulated noise to as high as 1.6 m/s. A comparison plot of the path for a simulated run with 0.2 m/s noise is shown below the sensitivity plot. This further validates our algebraic observability study and makes it clear that further performance gains will depend on improved handling of the real data nonidealities. We leave improving the performance of Doppler-only odometry as future work and look towards using models with higher learning capacity.

5.5 Summary and Conclusions

Our focus in this chapter was on learning measurement bias and noise models. Due to nonidealities in the real world, measurements can be biased in comparison to our known sensor models and their uncertainty may require a richer noise model that can vary depending on the application setting. We proposed modelling bias and covariance using feature-dependent regression models, where our main interest was on training the models using ESGVI and EM. We demonstrated EM training for odometry using FMCW lidar sensors, where we trained our models using the observed sensor data without supervision from the groundtruth trajectory. The odometry estimators we used for this problem appeared in two of our previously published works (Wu et al., 2023; Yoon et al., 2023).

In summary, the contributions of Chapter 5 are:

1. A methodology for training feature-dependent regression models for measurement bias and covariance using ESGVI and EM.

2. A lightweight correspondence-free odometry method using Doppler measurements from a FMCW lidar and gyroscope measurements from an IMU.

3. Experiments using the proposed method for learning Doppler measurement bias and variance, demonstrating the ability to train the regression models without supervision from the groundtruth trajectory.

Chapter 6

Learning Neural Networks

In the previous chapters, we investigated how one could tune measurement models for compensating measurement bias and/or assigning measurement uncertainty (i.e., covariance). We were interested in learning our models from the onboard sensor data, rather than relying on a source of groundtruth for supervision. A motivation for having such a methodology is to enable training on datasets that were collected in the field, rather than carefully curated datasets collected specifically for tuning our models. We now wish to enrich the same idea with Deep Neural Networks (DNNs).

According to the *universal approximation theorem* (Cybenko, 1989; Hornik et al., 1989; Goodfellow et al., 2016), a feedforward network with a linear output layer and at least one hidden layer (with enough hidden units and nonlinear activation functions) can approximate any continuous function on a closed and bounded subset of $\mathbb{R}^N$. In other words, a DNN is a general function approximator and we can in theory use it to approximate any function we require, given we make the network model large enough and have sufficient training data. In this chapter, we will use DNNs to learn more aspects of our measurement model than we have shown in the previous chapters. We will use our estimation and parameter learning framework using ESGVI and EM. By doing so, we can maintain a probabilistic estimation back-end, while taking advantage of the learning capacity of DNNs, all under a single data-likelihood learning objective. Maintaining a probabilistic estimation back-end gives us familiar benefits, such as outlier rejection (M-estimation), motion priors, continuous-time machinery, and interpretability of our estimator.

In summary, the main contribution of this chapter is the incorporation of DNNs to our parameter learning framework using ESGVI and EM, which we demonstrate using two different methodologies.

In Section 6.2, we first look into how we can model a measurement model in its entirety using a DNN. We demonstrate application of the derivative-free approach for ESGVI (see Section 3.2.5), which allows us to efficiently train and use a neural network measurement model without explicitly taking derivatives of the model with respect to our vehicle state. Instead of derivatives, we use multiple forward passes with sigmapoints, which complements the parallel work-flow of DNNs well. We demonstrate our methodology on the robot dataset from Section 3.6 to learn a range-bearing measurement model without supervison from the groundtruth trajectory. We present our experiments on this low-dimensional problem as a proof-of-concept and hope to enable further research on this methodology in future work.

In Section 6.4, we present an alternative way of using a DNN for application to rich sensor data. Instead of modelling the measurement model directly, which can be costly due to the density of measurements, we can model a DNN front-end that processes the rich data and outputs sparse features that can be used for state estimation. Our methodology is not specific to one type of sensor and generalizes to many because of our factor graph formulation to estimation and parameter learning. However, for this thesis we focus our experiments on lidar odometry and demonstrate learning without supervision from the groundtruth trajectory. Our experiments on lidar odometry was published to the IEEE Robotics and Automation Letters (Yoon et al., 2021). We additionally demonstrated the approach on radar odometry, which was presented at the Robotics: Science and Systems conference (Burnett et al., 2021) (equal-contribution work). We briefly discuss the methodology of radar odometry in this chapter and forward readers to the corresponding publication for the experimental results.

6.1 Related Work

6.1.1 Network Training

Our idea of parameter learning with ESGVI originates from a linear system identification method, where Ghahramani and Hinton (1996) optimize the likelihood of the observed measurements (data) by introducing a latent trajectory (state) and applying EM. In the e-step, model parameters are held fixed and the trajectory is optimized with Kalman smoothing. In the m-step, the trajectory is held fixed and the model parameters are optimized. Critically, this method is able to learn entire linear models from just the observed data, with no prior knowledge. Ghahramani and Roweis (1999) extend this concept to simple nonlinear models modelled using a Gaussian Radial Basis Functions (RBF) network. Inspired by the work of Ghahramani and Roweis (1999), which dates back to approximately two decades from the time of this thesis work, we revisit this idea of using EM to train network parameters, but with advances in both state estimation and DNN learning.

A similar concept based on optimizing the data likelihood by introducing a latent state is applied in the Variational Autoencoder (VAE) (Kingma and Welling, 2013) framework. With VAEs, in contrast to our approach using EM, inference of the latent state is approximated as a deep network (i.e., data input mapped to state output using a DNN). This approximation is restrictive for time-series application, such as trajectory estimation. EM parameter learning with ESGVI gives us all the benefits of probabilistic estimation, such as (i) information propagation and regularization over the entire latent trajectory through a kinematically motivated motion prior, (ii) robustness to outliers using techniques such as M-estimation (Zhang, 1997), (iii) continuous-time estimation for efficiently handling dense and/or asynchronous measurements (Barfoot et al., 2014; Anderson and Barfoot, 2015), and (iv) the flexibility of incorporating multiple sensor modalities as additional factors.

Alternatively, Bloesch et al. (2018) use a VAE in probabilistic trajectory estimation without directly inferencing the state with a deep network. They train a VAE to learn an efficient (lower-dimensional) latent space for geometry, and optimize over this domain jointly with pose variables at test time for monocular vision estimation problems. Czarnowski et al. (2020) extend this work by using the same depth representation in a full dense SLAM system. Unlike our approach, their network is trained independently from the trajectory estimator, and a training dataset with groundtruth depth images is required. One motivation of ours for using EM is to train our network parameters without groundtruth.

Evidently, deep learning with core components of probabilistic estimation is an increasingly popular avenue of research. Tang and Tan (2018) maintain the differentiability of the Levenberg-Marquardt optimizer by iterating for a fixed number of steps and proposing a network to predict the damping factor. Similarly, von Stumberg et al. (2020) backpropagate through the Gauss-Newton update step from a random initial condition. Jatavallabhula et al. (2020) go even further by proposing differentiable alternatives to all modules in full SLAM systems as computational graphs. Most recently, Pineda et al. (2022) present *Theseus*, an application-agnostic open source library for differentiable nonlinear least squares. In contrast to these methods, our approach does not rely on making the estimator differentiable and so facilitates using any available probabilistic estimation method and related tools. As an example, we can easily incorporate an outlier rejection technique such as RANSAC (Fischler and Bolles, 1981) without worrying about differentiability.

We first present a methodology that learns a measurement model in its entirety using a DNN. Taking derivatives of a DNN's output with respect to its input can be computationally demanding, requiring backpropagation for each output dimension. Sample-based estimators can be used instead, where function model evaluations with samples of the latent state replaces the need for taking analytical gradients. Schön et al. (2011b) applies EM with a particle smoother for parameter estimation, and similarly, Kokkala et al. (2016) applies cubature (e.g., sigmapoints) smoothing. Neither of these works have applied their method for neural network models. Rather, only a handful of parameters are estimated where most of the nonlinearity is predefined (e.g., for calibration or covariance estimation). We will use ESGVI, which can similarly be evaluated derivative-free with cubature sampling. ESGVI is not restricted to smoothing problems (i.e., does not require a block-tridiagonal inverse covariance) and can exploit the problem structure regardless of how the joint likelihood factors. There has been work that proposes learning nonlinear measurement models using Gaussian Processes (GPs) (Turner et al., 2010; Ko and Fox, 2009, 2011). Inference with GPs is a form of 'lazy learning' and scales cubically in computation with respect to the size of the training data. Sparse GP approximations to query on a subset of the training data (Snelson and Ghahramani, 2006) may be necessary for online application. In contrast, DNNs generalize to training data offline and off-the-shelf libraries readily have support for hardware-accelerated computing (i.e., parallel processing on a GPU).

Our methodology for modelling a sensor using a DNN is not computationally favourable for dense measurements from a rich sensor. Instead, for rich sensors we propose an alternative approach that uses a DNN as a front-end processor that outputs sparse features. We demonstrate the ability to train this network model without supervision from the groundtruth trajectory using EM. Most similar to this approach is the work of DeTone et al. (2018), where they alternate between training a deep network front-end that outputs visual features from images, and a bundle adjustment back-end that optimizes the feature observations as landmarks. The optimized landmarks become the training signal for learning the front-end network. While their idea is similar in practice to our approach with EM, our formulation is motivated from probabilistic data-likelihood objective and so the details of our optimization objective are different.

6.1.2 Application to Lidar Odometry

While our learning framework is sensor agnostic, our choice of application to demonstrate our work on rich sensor data is lidar odometry. We briefly discuss the current literature related to lidar odometry in this subsection, including works that apply deep learning. Note the lidar used in our experiments of this chapter is a standard 3D lidar sensor, and not a FMCW lidar sensor that we used in the previous chapter.

The current state of the art for non-learned lidar estimation are those based on ICP. ICP estimates the relative transformation between two pointclouds by iteratively re-associating point measurements via nearest-neighbour search (Besl and McKay, 1992; Pomerleau et al., 2015). Lidar odometry methods that achieve state-of-the-art performance apply this simple-but-powerful concept of nearest-neighbour data association in a low-dimensional space (e.g., Cartesian). Zhang and Singh (2017) present *LOAM*, which has been the top contender for lidar-only odometry in the KITTI odometry benchmark (Geiger et al., 2012) since its inception. Behley and Stachniss (2018) present *SuMa*, which is notably the method used as the trajectory groundtruth in SemanticKITTI (Behley et al., 2019), the KITTI odometry sequences with semantic labels.

Modern lidars output high-resolution, 3D pointclouds by mechanical actuation. Consequently, pointclouds acquired from a moving vehicle will be motion distorted, similar to a rolling-shutter effect. One can motion-compensate (de-skew, or undistort) the data as a preprocessing step (Ye et al., 2019; Vizzo et al., 2023). Alternatively, data can be incorporated at their exact measurement times by estimating a continuous-time trajectory (Furgale et al., 2012; Anderson and Barfoot, 2013; Barfoot et al., 2014). Continuous-time ICP-based methods have been successfully demonstrated in several works (Wong et al., 2020a; Dellenbach et al., 2022; Wu et al., 2023). State-of-the-art lidar odometry methods address the motion compensation problem and are capable of achieving highly accurate, real-time performance (Pan et al., 2021; Dellenbach et al., 2022; Vizzo et al., 2023). Pan et al. (2021) extract low-level geometric features to apply multiple error metrics in their ICP optimization. Dellenbach et al. (2022) use a sparse voxel data structure for downsampling and nearest-neighbour search in a single-threaded implementation. Vizzo et al. (2023) demonstrate faster performance with comparable accuracy by proposing a simplified registration pipeline that requires few tuning parameters in a multi-threaded implementation.

At the opposite end, we have fully-learned lidar odometry methods that infer relative poses with a deep network. (Li et al., 2019) present *LO-Net*, a network that takes two pointclouds as input and outputs the relative pose. Their method demonstrates competent odometry performance comparable to ICP-based ones, but requires training with supervision from groundtruth trajectories. Cho et al. (2020) present *DeepLO*, a network that similarly outputs a relative pose change from two input pointclouds and is trained unsupervised. However, their unsupervised approach comes at a cost, as the odometry performance they present falls short compared to LO-Net and existing ICP-based methods. The learned estimator in LO-Net is impressive, but Li et al. (2019) demonstrate better odometry in the same publication with ICP enhanced with point measurement masks that were trained alongside LO-Net. In similar fashion, Chen et al. (2019) improve on SuMa with SuMa++, which incorporates a pretrained semantic classification network. These outcomes suggest that a hybrid of non-learned estimators with learned components can be beneficial. Our work is motivated by the idea that in the spectrum between non-learned and fully learned estimators, there is an optimal balance that can benefit from the advantages of both extremes. The learning framework and odometry solution we present is our attempt at meeting this balance.

Compared to fully learning the estimator, such as in DeepLO (Cho et al., 2020) and LO-Net (Li et al., 2019), our odometry solution achieves better performance while being trained unsupervised. Compared to SuMa++ (Chen et al., 2019) and LO-Net with ICP, our approach learns more and does not rely on nearest-neighbours for data association. Having more learnable components has the advantage of automation, i.e., being able to tune models from data for different deployments, rather than requiring an expert engineer. Another advantage of not relying on ICP is that our method inherently has a reasonably good estimate of trajectory uncertainty. Uncertainty estimation for ICP is on its own a challenging research problem (Landry et al., 2019; Brossard et al., 2020b).

6.2 Neural Network Measurement Model

In this section we present the methodology for modelling and training a DNN measurement model. Using our ESGVI and EM parameter learning framework, we can take advantage of the learning capacity of DNNs while using a classic estimation back-end.

Recall from Section 3.3 our data likelihood objective. Repeating the objective here for convenience,

$$\mathscr{L} = -\ln p(\mathbf{y}|\boldsymbol{\theta}) = \underbrace{\int_{-\infty}^{\infty} q(\mathbf{x}) \ln \left(\frac{p(\mathbf{x}|\mathbf{y},\boldsymbol{\theta})}{q(\mathbf{x})} \right) d\mathbf{x}}_{\leq 0} \underbrace{- \int_{-\infty}^{\infty} q(\mathbf{x}) \ln \left(\frac{p(\mathbf{x},\mathbf{y}|\boldsymbol{\theta})}{q(\mathbf{x})} \right) d\mathbf{x}}_{\text{upper bound}},$$

where we know that the upper bound term is equivalent to our ESGVI loss,

$$V(q|\boldsymbol{\theta}) = \mathbb{E}_q[\phi(\mathbf{x}, \mathbf{y}|\boldsymbol{\theta})] + \frac{1}{2}\ln(|\boldsymbol{\Sigma}^{-1}|). \tag{6.1}$$

As per usual, we will learn our model parameters, $\boldsymbol{\theta}$, while simultaneously estimating for the latent state, $q(\mathbf{x})$, using EM.

We will for now not specify the details of the network architecture as that can be largely dependent on the application. We instead represent our network model as a generic measurement function, $\mathbf{y}_k = \mathbf{g}(\mathbf{x}_k|\boldsymbol{\theta})$, where $\mathbf{y}_k$ is the output measurement from our model that takes the state estimate, $\mathbf{x}_k$, as input, and depends on the network parameters, $\boldsymbol{\theta}$. We assume that we are applying a large enough network to learn the measurement model from our training data.

We will assume our joint log-likelihood between our states and measurements factors apart in the following way:

$$\phi(\mathbf{x}, \mathbf{y}|\boldsymbol{\theta}) = \sum_k \left(\phi_p(\mathbf{x}_{k-1}, \mathbf{x}_k) + \phi_m(\mathbf{y}_k|\mathbf{x}_k, \boldsymbol{\theta})\right), \tag{6.2}$$

where $\phi_p(\cdot,\cdot)$ are motion prior factors and $\phi_m(\cdot|\cdot,\boldsymbol{\theta})$ are measurement factors that depend on the parameters, $\boldsymbol{\theta}$. Focusing on just the measurements, we can write each factor as

$$\phi_m(\mathbf{y}_k|\mathbf{x}_k, \boldsymbol{\theta}) = \frac{1}{2}\mathbf{e}_{m,k}^T \mathbf{W}_k^{-1}\mathbf{e}_{m,k} - \frac{1}{2}\ln(|\mathbf{W}_k^{-1}|), \tag{6.3a}$$

$$\mathbf{e}_{m,k} = \mathbf{y}_k - \mathbf{g}(\mathbf{x}_k|\boldsymbol{\theta}), \tag{6.3b}$$

In the m-step, we need to optimize for $\boldsymbol{\theta}$ by taking the derivative of our loss functional while holding our latent state, $q(\mathbf{x})$, fixed. However, our network measurement model is nonlinear with respect to our latent state. We can approximate the uncertainty of our latent state using cubature approximations of the expectations. Then the gradients of the loss functional can easily be computed with existing, off-the-shelf DNN training tools (i.e., backpropagation with automatic differentiation libraries). For clarity, we can write the required gradient evaluated with sigmapoints as follows:

$$\begin{aligned}\frac{\partial V(q|\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} &= \sum_k \mathbb{E}_{q_k}\left[\frac{\partial}{\partial \boldsymbol{\theta}}\phi_{m,k}(\mathbf{y}_k|\mathbf{x}_k, \boldsymbol{\theta})\right] = \sum_k \sum_i w_{k,i}\left(\frac{\partial}{\partial \boldsymbol{\theta}}\phi_{m,k}(\mathbf{x}_{k,i}, \mathbf{y}_k|\boldsymbol{\theta})\right) \\ &= \frac{\partial}{\partial \boldsymbol{\theta}}\left(\sum_k \sum_i \frac{w_{k,i}}{2}\left(\mathbf{y}_k - \mathbf{g}(\mathbf{x}_{k,i}|\boldsymbol{\theta})\right)^T \mathbf{W}_k^{-1}\left(\mathbf{y}_k - \mathbf{g}(\mathbf{x}_{k,i}|\boldsymbol{\theta})\right)\right),\end{aligned} \tag{6.4}$$

where $w_{k,i}$ are weights, $\mathbf{x}_{k,i} = \boldsymbol{\mu}_k + \sqrt{\boldsymbol{\Sigma}_k}\boldsymbol{\xi}_{k,i}$ are sigmapoints, and $\boldsymbol{\xi}_{k,i}$ are unit sigmapoints.

Both the weights and unit sigmapoints are specific to the cubature method. With the inclusion of cubature sampling, we see that with the gradient expression is in a familiar form for supervised regression training.

In the e-step, we use our current best measurement model to improve our estimate of our latent state, $q(\mathbf{x})$, using ESGVI. While it is possible to use automatic differentiation from our usual DNN tools to compute the gradient of our learned measurement model with respect to our state, running backpropagation on multiple dimensions of the model output can become costly[1]. We can instead use our derivative-free formulation of ESGVI. This will require running multiple forward passes through our network model for each sigmapoint, a task that is easily computed in parallel on a GPU. Intuitively, we could also run multiple forward passes through our model to calculate a numerical approximation of the required gradient. Our approach with derivative-free ESGVI is similar in concept, but takes into account the uncertainty of our latent state to determine the spread of the perturbed states, rather than choosing them ad hoc.

In Chapter 3, we took a deep look into how Gaussian cubature can be applied to approximate our expectation expressions. Higher-order expressions with respect to our state will require more cubature points for an accurate approximation, requiring more compute. Therefore it may be beneficial to consider the alternate loss functional to reduce the number of required cubature points. Recall that the alternate loss is a good, conservative approximation of the full ESGVI loss under mild nonlinearities and/or concentrated posteriors. While we do not expect the nonlinearity of our network model to be mild, we may find in our applications that the posterior is concentrated enough to justify the alternate loss. For example, the required gradient for our m-step using the alternate ESGVI loss will be

$$
\begin{aligned}
\frac{\partial V'(q|\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} &= \frac{\partial}{\partial \boldsymbol{\theta}} \sum_k \left(\frac{1}{2} \mathbb{E}_{q_k}\left[\mathbf{y}_k - \mathbf{g}(\mathbf{x}_k|\boldsymbol{\theta})\right]^T \mathbf{W}_k^{-1} \mathbb{E}_{q_k}\left[\mathbf{y}_k - \mathbf{g}(\mathbf{x}_k|\boldsymbol{\theta})\right] \right) \\
&= \frac{\partial}{\partial \boldsymbol{\theta}} \sum_k \left(\frac{1}{2} \left(\sum_i w_{k,i}(\mathbf{y}_k - \mathbf{g}(\mathbf{x}_{k,i}|\boldsymbol{\theta})) \right)^T \mathbf{W}_k^{-1} \left(\sum_i w_{k,i}(\mathbf{y}_k - \mathbf{g}(\mathbf{x}_{k,i}|\boldsymbol{\theta})) \right) \right).
\end{aligned}
\tag{6.5}
$$

[1]Off-the-shelf libraries for DNN training are designed for automatic differentiation of parameters with respect to a scalar loss term. Using the same tool to compute a measurement Jacobian will require multiple queries for each dimension of the measurement.

6.3 Experiment: Learning a Range-Bearing Model without Derivatives

6.3.1 Experiment Setup

We demonstrate our derivative-free approach to train a DNN range-bearing measurement model on the same robot dataset used in Experiment 3 of Chapter 3 (see Section 3.6). While there is little practical value in learning a well-defined measurement model, we use this experiment to establish a proof of concept of our method for training without groundtruth and to motivate further research in this direction. Once again, we assume that the data association (i.e., which measurement corresponds to which landmark) is known in this experiment. We partition the 12,000 timesteps of the dataset into 6 sequences of 2,000 timesteps each. Out of the 6 sequences, we will use 3 as our training set, 1 as our validation set, and 2 for our test set.

Since our focus is on learning the range-bearing measurement model, we simplify the estimation problem in a few aspects: (i) we switch the estimation problem to localization by assuming that the map of landmarks is known (i.e., we do not estimate landmarks as part of the state), (ii) we reduce the dimension of the state to 3 by removing the robot velocity, and (iii) we remove the WNOA motion prior. By reducing the dimension of the state, we can reduce the number of sigmapoints required for our ESGVI method, reducing compute. Since we no longer estimate the robot velocity, we do not require the WNOA motion prior. We instead re-purpose the wheel encoder measurements into a motion model, which is common practice in many robot applications. We assume our motion model is known and we will learn the range-bearing model. Therefore there are three types of factors in our estimation problem: (i) the range-bearing measurement factors using our learned DNN model, (ii) the motion model factors, and (iii) an initial condition factor on the first robot pose of each sequence.

We define our state to be

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_K \end{bmatrix}, \tag{6.6}$$

where $\mathbf{x}_k$ is the robot state at timestep t_k, $\mathbf{x}_k = \begin{bmatrix} x_k & y_k & \theta_k \end{bmatrix}^T$. The measurement model we require our DNN to learn is the range-bearing model. For convenience, we repeat the model here:

$$\begin{bmatrix} r_k^\ell \\ b_k^\ell \end{bmatrix} = \begin{bmatrix} \sqrt{(x_\ell - x_k - d\cos\theta_k)^2 + (y_\ell - y_k - d\sin\theta_k)^2} \\ \operatorname{atan2}(y_\ell - y_k - d\sin\theta_k, x_\ell - x_k - d\cos\theta_k) - \theta_k \end{bmatrix}, \tag{6.7}$$

where $\mathbf{m}_\ell = \begin{bmatrix} x_\ell & y_\ell \end{bmatrix}^T$ is a known landmark position, and d is an extrinsic calibration parameter that describes the placement of the sensor on the robot platform.

Since this is a relatively simple, low-dimensional measurement model, our network can be relatively shallow in comparison to networks that work with rich data. We take the intuitive approach, which is to apply a fully connected network with input dimension 5, corresponding to the sum of the state and landmark dimensions, and output dimension 2, corresponding to the dimension of the range-bearing measurement. We apply the ReLU nonlinear function as our activation for the hidden layers, while the width and depth of the hidden layers remain as tuning hyperparameters. We can write this model as a function

$$\mathbf{g}(\mathbf{x}_k, \mathbf{m}_k^\ell | \boldsymbol{\theta}), \tag{6.8}$$

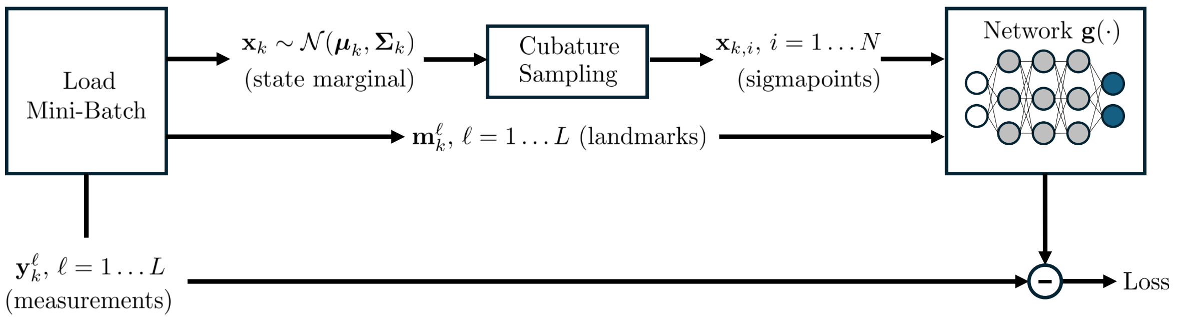
where we have two distinct inputs: the state, $\mathbf{x}_k$, and a landmark position, $\mathbf{m}_k^\ell$.

However, it turns out that the network model above is not capable of learning the range-bearing model well without the groundtruth trajectory. Our data likelihood learning objective does not provide enough information to learn a latent state space that is consistent with the given map. Our solution is to combine the robot state input with the landmark input by transforming the landmark into the local frame of the robot. We can write this alternate model as a function

$$\mathbf{g}(\mathbf{T}_k \mathbf{m}_k^\ell | \boldsymbol{\theta}), \tag{6.9}$$

where we have converted the vectorspace $\mathbf{x}_k$ to a transformation matrix $\mathbf{T}_k \in SE(2)$, which we use to transform the landmark position into the local robot frame. We provide experimental results using both models and further discuss the comparison in the following subsection.

In summary, we train our model by alternating between the e-step and m-step. We found that using the alternative loss functional for ESGVI works sufficiently, which we recall helps reduce compute by lowering the number of sigmapoints we require. For the e-step, we run a batch optimization for each sequence independently. For the m-step, we train our model with the Adam optimizer (Kingma and Ba, 2014) over the entire training



Load Mini-Batch
$\mathbf{x}_k \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ (state marginal)
Cubature Sampling
$\mathbf{x}_{k,i}$, $i = 1 \ldots N$ (sigmapoints)
Network $\mathbf{g}(\cdot)$
$\mathbf{m}_k^\ell$, $\ell = 1 \ldots L$ (landmarks)
$\mathbf{y}_k^\ell$, $\ell = 1 \ldots L$ (measurements)
Loss

Figure 6.1: A block diagram illustrating the evaluation of our loss to train our network measurement model. For convenience, we illustrate each mini-batch as data from a single timestamp at time t_k, but in practice we use a mini-batch of several timestamps. The state marginals are from the latest solve of the e-step.

set while holding our trajectory estimates fixed. We use a validation set for early stopping within each m-step, i.e., we evaluate the loss functional of the validation set and stop training once it ceases to further decrease. We apply the same early-stopping criterion to terminate the outer EM loop. Figure 6.1 illustrates a block diagram for evaluating our loss (see (6.5)) during the m-step. We load a mini-batch of data consisting of the observed measurements, $\mathbf{y}_k^\ell$, the corresponding landmark positions, $\mathbf{m}_k^\ell$, and the corresponding state marginals, $\mathbf{x}_k \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ from the latest e-step. We use Gaussian cubature to sample sigmapoints from the state marginals and forward propagate the sigmapoints through our network measurement model. We then evaluate our loss using the network outputs and the observed measurements, for which we apply backpropagation on to update the network parameters.

6.3.2 Results and Discussion

We evaluate localization using our learned range-bearing measurement models that we train without the groundtruth trajectory. Recall that we have two network models: a model that takes the robot state and landmark position as inputs separately, $\mathbf{g}(\mathbf{x}_k, \mathbf{m}_k^\ell|\boldsymbol{\theta})$ (model 1), and a model that takes a combined input of the two via a transformation of the landmark into the local robot frame using the state, $\mathbf{g}(\mathbf{T}_k\mathbf{m}_k^\ell|\boldsymbol{\theta})$ (model 2).

Figure 6.2 is a plot of the paths of our localization with the two models and the groundtruth. Notice how the shape of the path for model 1 is approximately congruent to the groundtruth, but seemingly offset by a translation. It turns out that the ESGVI loss functional is doing what we asked it to, i.e., the loss encourages the network model to

Table 6.1: A table of the quantitative results for localization on two test sequences for learning a range-bearing measurement model without the groundtruth trajectory. We present the RMSE for translation [m] and rotation [rad] (units are shown as [m]/[rad]). Best values are highlighted in bold font between the two proposed models (excluding the known model). The results clearly demonstrate the performance improvement of using model 2 over model 1. Comparing model 2 to the known model, which we can consider as the groundtruth model, it appears to perform on par in translation, but not in rotation.

Test Seq.	Model 1	Model 2	Known Model
1	0.352/0.109	**0.030/0.086**	0.031/0.043
2	0.363/0.113	**0.033/0.080**	0.031/0.039
Avg.	0.358/0.111	**0.032/0.083**	0.031/0.041

learn a parameter configuration that can explain the data (increase the data likelihood). However, EM ends up converging to a local minimum that has the latent state space in a different reference frame than the given map. Using model 1, we provided too little information to encourage the latent space to be in the same reference frame as the given map, which is problematic for a localization application. While the prior factor on the first pose does provide this information, it alone is not strong enough to encourage convergence to a good solution in our experiments.

Understanding why model 1 does not work well, we can see the reasoning behind the design of model 2. We can encourage the latent space to be in the same reference as the known map by transforming the input landmark position into the local robot frame before inputting it into the network. Looking at Figure 6.2 again, we see that we are able to localize to the map reasonably well using our learned range-bearing model.

Table 6.1 shows the quantitative results of localization for our 2 models and the known range-bearing model from (6.7). Model 2 clearly performs better in both translation and rotation than model 1. Interestingly, model 2 appears to perform on-par with the known model in translation. However, the known model clearly outperforms model 2 in rotation. While we have exhausted the data available, it would be interesting to see in future work if increasing the training dataset size would help reduce this gap in performance.

6.4 Deep Features for Rich Data

In the previous two sections of this chapter, we discussed a methodology for learning a measurement model using a DNN and our ESGVI and EM parameter learning framework.

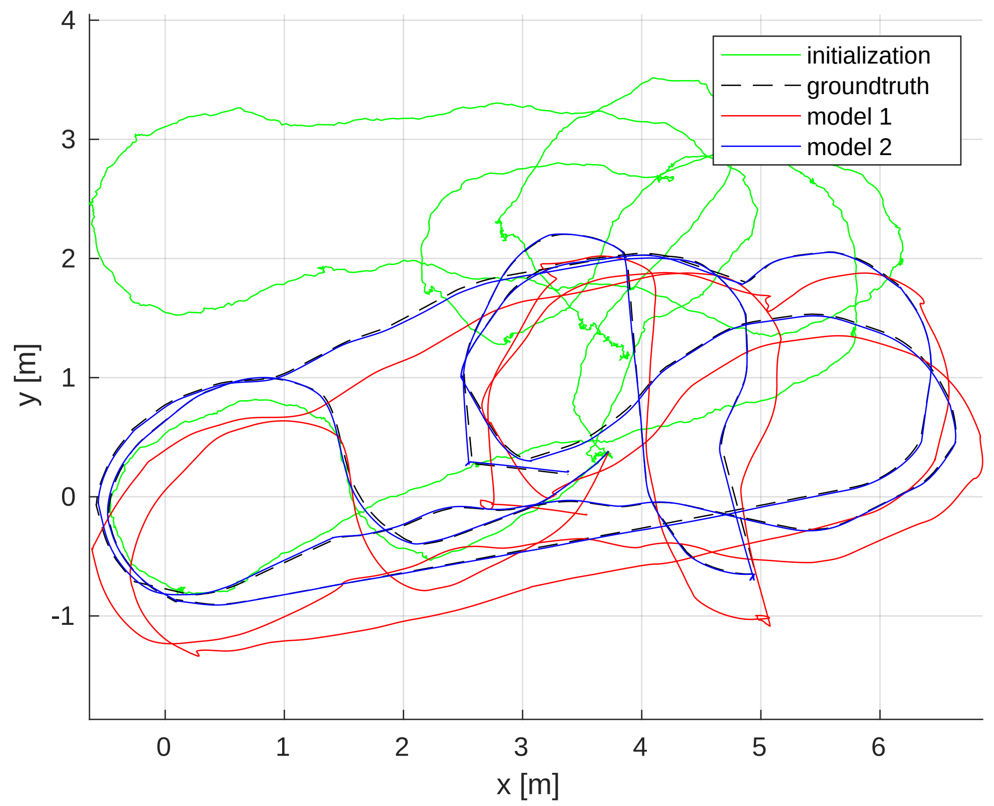


Figure 6.2: A comparison of localization with the two learned range-bearing models and the groundtruth. Model 1 takes the robot state and landmark as the input, while model 2 takes the transformed landmark in the local robot frame as input. Model 1 is unable to learn a latent state space that is in the same reference frame as the given map. Transforming the landmark positions into the local robot frame helps resolve this issue.

The appeal of the approach was to take advantage of the learning capacity of DNNs while using a probabilistic estimation back-end. However, we limited our experiment to a low-dimensional problem because the current methodology without further improvements may not be computationally suitable for handling rich sensor data (e.g., dense pointclouds from a lidar). In this section, we present an alternative approach to incorporating a DNN that is designed specifically for rich sensor data. We accomplish this by learning a CNN front-end that processes rich sensor data into sparse features. Those sparse features can then be used in our probabilistic estimation back-end (i.e., ESGVI).

We will focus our methodology on odometry using rich sensor data. We define our state at time t_k as $\mathbf{x}_k = \{\mathbf{T}_{k,0}, \boldsymbol{\varpi}_k\}$, where the pose $\mathbf{T}_{k,0} \in SE(3)$ is a transformation matrix between frames at t_k and t_0, and $\boldsymbol{\varpi}_k \in \mathbb{R}^6$ is the body-centric velocity. We assume we receive a new sensor frame from the sensor at each new time t_k.

Our odometry implementation is an optimization over a window of w sensor frames,

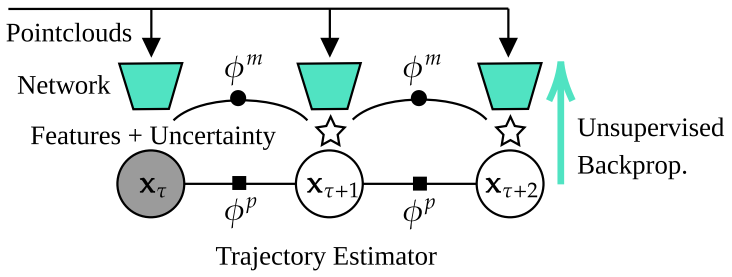


Figure 6.3: An example factor graph diagram of a odometry problem using pointcloud data from a lidar. We optimize the trajectory over a sliding window of w time frames (e.g., $w = 3$ above), where $\mathbf{x}_k$ is our state at time t_k. The first frame is locked (grey) and is the reference frame, we do not optimize it. Each frame receives a pointcloud from the lidar sensor. A deep network takes each pointcloud as input and outputs features with uncertainty (stars) that can be associated to other frames and composed into measurement factors, ϕ^m (circles). Motion prior factors, ϕ^p (squares), are applied between every frame. We do not require supervision, and only learn from the on-board lidar data.

$t_\tau, \ldots, t_{\tau+w-1}$. The first pose of the window at t_τ, $\mathbf{T}_{\tau,0}$, is locked (not optimized) and treated as the reference frame for keypoint matching. The factorization of our joint likelihood of the state and data is

$$\phi(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta}) = \sum_{k=\tau+1}^{\tau+w-1} \left(\phi^p(\mathbf{x}_{k-1}, \mathbf{x}_k) + \sum_{\ell=1}^{L_k} \phi^m(\mathbf{z}_k^\ell | \mathbf{x}_\tau, \mathbf{x}_k, \boldsymbol{\theta}) \right), \tag{6.10}$$

where $\mathbf{z}_k^\ell$ is the ℓth keypoint measurement in sensor frame k, which has a total of L_k keypoints. Figure 6.3 shows an example factor graph illustration for a lidar odometry problem.

Referring to Figure 6.3, the square factors, ϕ^p, are motion prior factors. We use the $SE(3)$ WNOA motion prior by Anderson and Barfoot (2015). The circle factors in Figure 6.3[2], ϕ^m, are the measurement factors defined by the sparse keypoints:

$$\phi^m(\mathbf{z}_k^\ell | \mathbf{x}_\tau, \mathbf{x}_k, \boldsymbol{\theta}) = \frac{1}{2}\left(\mathbf{z}_k^\ell - \mathbf{g}(\mathbf{x}_\tau, \mathbf{x}_k)\right)^T \mathbf{W}_k^\ell \left(\mathbf{z}_k^\ell - \mathbf{g}(\mathbf{x}_\tau, \mathbf{x}_k)\right) - \frac{1}{2}\ln\left|\mathbf{W}_k^\ell\right|, \tag{6.11}$$

where we use the log-likelihood of a Gaussian as the factor, and $\mathbf{W}_k^\ell$ is the inverse covariance matrix corresponding to measurement $\mathbf{z}_k^\ell$. The keypoint, $\mathbf{z}_k^\ell$, its inverse covariance matrix, $\mathbf{W}_k^\ell$, and the measurement model, $\mathbf{g}(\cdot)$, are quantities that depend on the network parameters, $\boldsymbol{\theta}$. These quantities will be further explained in the following subsection.

[2]This is a general illustration with measurement factors between frames. For implementation, we associate each frame only to the reference (see (6.11)).

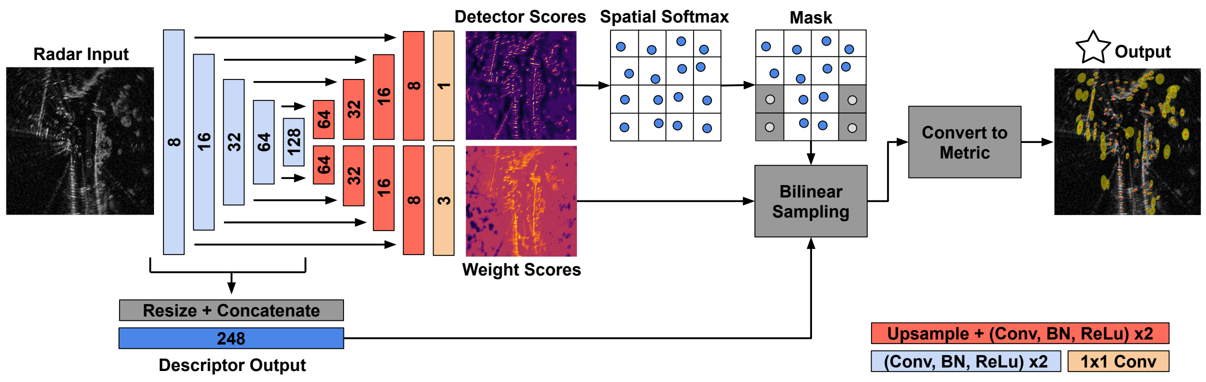


Figure 6.4: The network architecture from our publication on radar odometry (Burnett et al., 2021). The network outputs detector scores for keypoint detection, weight scores predicting keypoint uncertainty, and descriptors for matching. The weight scores are composed into 2×2 inverse covariance matrices. Descriptors are the concatenation of all encoder layer outputs after resizing via bilinear interpolation. The encoder and decoder layers are a double application of a 3×3 convolution, batch normalization, and ReLU nonlinearity. The layer sizes vary by a factor of 2 through max-pooling (encoder) and bilinear upsampling (decoder). Note that the output size is the same as the input, and are visually smaller in the interest of space. An output 1×1 convolution is applied for the detector and weight scores. The detector score map is partitioned into uniform cells, where a spatial softmax and weighted summation of coordinates are applied to yield a keypoint for each cell. Corresponding weights and descriptors are obtained via bilinear sampling.

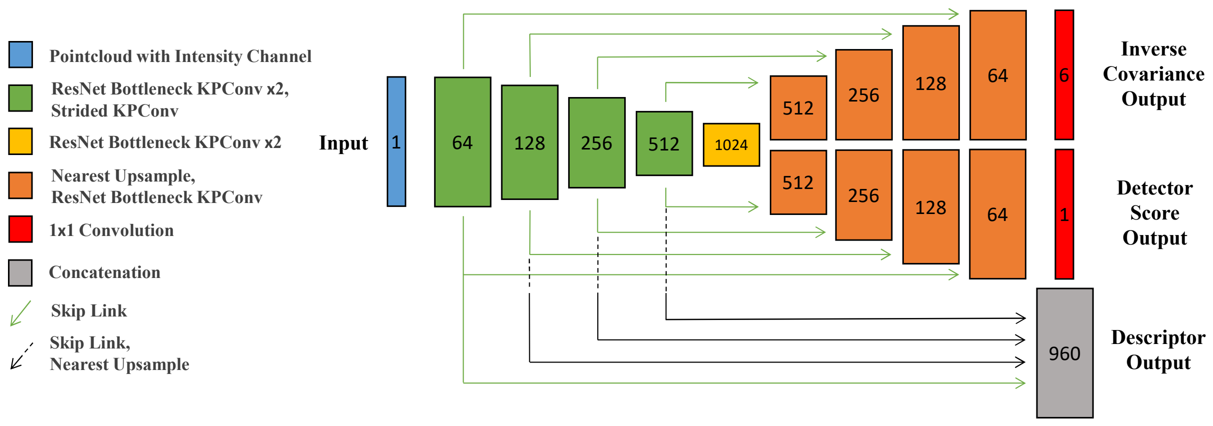

Pointcloud with Intensity Channel
ResNet Bottleneck KPConv x2, Strided KPConv
ResNet Bottleneck KPConv x2
Nearest Upsample, ResNet Bottleneck KPConv
1x1 Convolution
Concatenation
Skip Link
Skip Link, Nearest Upsample
Input
1
64
128
256
512
1024
512
256
128
64
6
512
256
128
64
1
960
Inverse Covariance Output
Detector Score Output
Descriptor Output

Figure 6.5: The network architecture from our publication on lidar odometry (Yoon et al., 2021). We apply convolutions on pointclouds using the KPConv (Thomas et al., 2019) pointcloud convolution operator. Input to the network is a lidar pointcloud with an intensity channel. Descriptor vectors for each point are composed from the output of the first four encoder layers. The 6 channel output of the top decoder are composed into inverse measurement covariances (see (6.13) in body text). The single channel output of the remaining decoder are detector scores used to compute keypoints (see (6.12) in body text). Refer to the KPConv (Thomas et al., 2019) publication for implementation details of the various operations.

6.4.1 Network Architecture for Deep Features

The design of the network architecture for learning sparse features from rich sensor data will differ depending on the sensor modality. In general, we follow the architecture design of Barnes and Posner (2020). They present a U-Net (Ronneberger et al., 2015) style convolutional encoder-multi-decoder network architecture that outputs keypoints and descriptors from radar data projected into a 2D bird's-eye view image.

In our publication on unsupervised radar odometry (Burnett et al., 2021), we can closely apply the architecture of Barnes and Posner (2020). Figure 6.4 shows an illustration of the network architecture. Similar to Barnes and Posner (2020), we have a single encoder that branches into 2 decoder outputs. The first decoder output determines the locations of the sparse keypoints via a spatial softmax on a uniformly partitioned grid and is unchanged from the original work. An aspect that differs from the architecture of the original work is the output of the second decoder. Barnes and Posner (2020) output a scalar weight score that measures the quality of each keypoint. We instead output a vector of weights that are constructed into the keypoint (inverse) covariance, i.e., we learn the measurement uncertainty. This uncertainty is visualized in the output on the far right of Figure 6.4 as the uncertainty ellipses.

We keep the discussion on radar odometry brief as our intention is to focus on the methodology for lidar odometry in this thesis, which also requires more explanation as the sensor modality is different from the radar used in the work of Barnes and Posner (2020). In the radar architecture, the convolutions are applied on radar data projected into a 2D bird's-eye view image, meaning that the convolutional kernels have a spatial extent in 2D Euclidean space. If we wish to apply this idea to lidar data, we can achieve an equivalent effect for 3D pointclouds with KPConv (Thomas et al., 2019), a pointcloud convolution method that uses kernel points arranged in a sphere of fixed radius[3]. Figure 6.5 shows the network architecture for pointcloud data, where in place of *pixels* of an image with feature channels, we have *points* of a pointcloud with feature channels.

The input to the network is a pointcloud with a channel for intensity data. Each network layer j, including the input layer 0, uniformly subsamples the pointcloud into a voxel grid of dimension dl_j. Successive layers in the encoder increase the grid dimension by a factor of 2, i.e., $dl_{j+1} = 2\,dl_j$, and therefore the convolutions are applied at different scales in Euclidean space. The opposite is true for the decoder layers in order to have the input and output dimensions be equal. We set dl_0 to be 0.3 m.

Each layer of the encoder consists of two KPConv variations of bottleneck ResNet

[3]Thomas et al. (2019) also present a deformable kernel implementation, but we do not apply it in our work.

blocks (He et al., 2016), followed by a strided variation for spatial dimension reduction. Each layer of the decoder consists of a nearest upsample operation, for spatial dimension enlargement, and a single KPConv bottleneck ResNet block. These convolution blocks apply the leaky ReLU nonlinearity and batch normalization. We direct readers to the KPConv publication (Thomas et al., 2019) for detailed definitions of the various block operations. As in U-Net (Ronneberger et al., 2015), we use skip connections between encoder and decoder layers.

Similar to the radar architecture, the lidar network outputs descriptors, weights, and detector scores for each input point. The descriptor vectors are computed for each point in the input layer and are composed of the output feature channels of all but the last encoder layer. The channel output dimension of the first layer is 64, and doubles for each subsequent layer. The output channels of the layers are concatenated with nearest upsampling to create descriptors of length 960, which are normalized into unit vectors. The detector scores are used to compute the keypoint locations, $\mathbf{z}_k^\ell$, and the weights are used to construct the corresponding inverse covariance matrices, $\mathbf{W}_k^\ell$.

We convert the detector scores into keypoint locations (coordinates), $\mathbf{z}_k^\ell$, using a spatial softmax operation. We partition the pointcloud into voxels of grid size dg (we set dg to be 1.6 m) for the purpose of computing one keypoint per voxel. For each voxel, we apply a softmax over the detector scores, resulting in weights we use to compute the keypoint's coordinates along with its descriptor and inverse covariance. For example, the ℓth keypoint coordinate in frame k is

$$\mathbf{z}_k^\ell = \sum_{i=1}^{M} \frac{\exp s_i}{\sum_{j=1}^{M} \exp s_j} \mathbf{p}^i, \tag{6.12}$$

where $s_1, \ldots, s_M$ are the detector scores of voxel ℓ, and $\mathbf{p}^1, \ldots, \mathbf{p}^M \in \mathbb{R}^3$ are the corresponding point coordinates. A similar computation is done to get the descriptor vector, $\mathbf{d}_k^\ell$, and inverse covariance, $\mathbf{W}_k^\ell$, for each keypoint. For the inverse covariance, we apply the weighted summation over the 6D weight vector, and compose it into the 3×3 matrix afterward.

We compose the inverse covariance matrices following the approach of Liu et al. (2018), which uses the following LDU decomposition for symmetric, positive definite matrices:

$$\mathbf{W} = \begin{bmatrix} 1 & 0 & 0 \\ \ell_1 & 1 & 0 \\ \ell_2 & \ell_3 & 1 \end{bmatrix} \begin{bmatrix} \exp d_1 & 0 & 0 \\ 0 & \exp d_2 & 0 \\ 0 & 0 & \exp d_3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ \ell_1 & 1 & 0 \\ \ell_2 & \ell_3 & 1 \end{bmatrix}^T, \tag{6.13}$$

where $[\ell_1, \ell_2, \ell_3, d_1, d_2, d_3]$ is the 6D weight output for each point. Note that in the radar architecture in Figure 6.4, we would only require a 3D weight output to construct a 2×2 inverse covariance matrix.

We use the keypoint descriptor for data association, which will be matched to a point in the reference pointcloud at time t_τ. Differentiability is maintained by approximating all matches with a softmax (Barnes and Posner, 2020; Wang and Solomon, 2019). We compute the dot product between each keypoint descriptor and all descriptors of the reference pointcloud:

$$\mathbf{c}_k^{\ell T} = \mathbf{d}_k^{\ell T} \begin{bmatrix} \mathbf{d}_\tau^1 & \dots & \mathbf{d}_\tau^N \end{bmatrix}, \tag{6.14}$$

where $\mathbf{d}_k^\ell$ is the descriptor vector of keypoint $\mathbf{z}_k^\ell$, and $\mathbf{d}_\tau^1, \dots, \mathbf{d}_\tau^N$ are the descriptor vectors of the N points in the reference frame. We apply a softmax function on $\mathbf{c}_k^\ell$, and compute a weighted summation. The reference point match for keypoint $\mathbf{z}_k^\ell$ is therefore

$$\mathbf{r}_\tau^{\ell_k} = \sum_{i=1}^{N} \frac{\exp c_{k,i}^\ell}{\sum_{j=1}^N \exp c_{k,j}^\ell} \mathbf{p}_\tau^i, \tag{6.15}$$

where $c_{k,1}^\ell, \dots, c_{k,N}^\ell$ are the scalar elements of $\mathbf{c}_k^\ell$, and $\mathbf{p}_\tau^1, \dots, \mathbf{p}_\tau^N$ are the reference point coordinates.

We can now fully define the measurement factor in (6.11) with outputs of the network:

$$\phi^m(\mathbf{z}_k^\ell | \mathbf{x}_\tau, \mathbf{x}_k, \boldsymbol{\theta}) = \frac{1}{2}\left(\mathbf{z}_k^\ell - \mathbf{D}\mathbf{T}_{k,0}\mathbf{T}_{0,\tau}\begin{bmatrix} \mathbf{r}_\tau^{\ell_k} \\ 1 \end{bmatrix}\right)^T \mathbf{W}_k^\ell \left(\mathbf{z}_k^\ell - \mathbf{D}\mathbf{T}_{k,0}\mathbf{T}_{0,\tau}\begin{bmatrix} \mathbf{r}_\tau^{\ell_k} \\ 1 \end{bmatrix}\right) - \frac{1}{2}\ln\left|\mathbf{W}_k^\ell\right|, \tag{6.16}$$

where $\mathbf{D}$ is a 3×4 constant selection matrix that removes the homogeneous element.

Figure 6.6 shows visualizations of the learned detector scores and covariances. The detector favours points on, and in the vicinity of, structure such as wall corners and vertical posts. Interestingly, the nearby ground is favoured over vehicles, possibly due to their dynamic nature. We visualize sphericity (Thomas et al., 2018) to demonstrate the covariance. Instead of manually choosing the error metric (e.g., point-to-plane), the network adapts to low-level geometry.

6.4.2 Training and Inference

We simultaneously estimate the vehicle trajectory and train the parameters of our network using EM. In the e-step, we hold all network parameters, $\boldsymbol{\theta}$, fixed and optimize

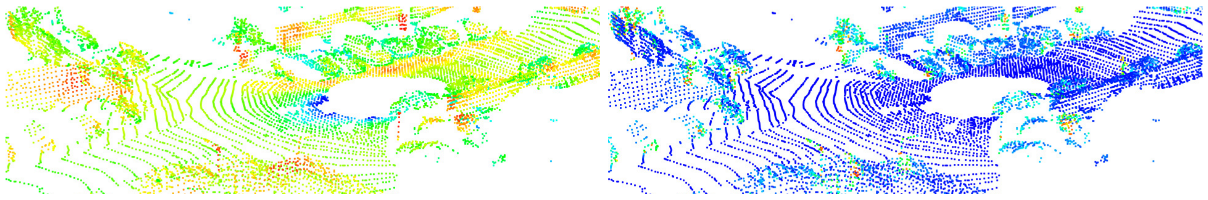


Figure 6.6: Network outputs coloured in the following order: blue (low value), cyan, green, yellow, and red (high value). (Left) Detector score visualization, highlighting structure such as wall corners and vertical posts. The nearby ground is favoured over vehicles, possibly due to their dynamic nature. (Right) Visualization of neighbourhood sphericity computed with the learned measurement covariance. Planar surfaces have low values, which is the expected result.

for the posterior, $q(\mathbf{x})$. In the m-step, we hold the posterior, $q(\mathbf{x})$, fixed and optimize for the network parameters, $\boldsymbol{\theta}$. As discussed previously in Chapter 5 when learning feature-dependent measurement covariances, the m-step does not have to be computed to convergence, before alternating to the e-step, to satisfy the iterative update scheme of the data likelihood. In other words, we will apply generalized EM (GEM).

We adapt GEM to seamlessly fit into conventional network training (i.e., stochastic gradient optimization) by including the e-step in the forward propagation routine. A window of sequential sensor frames is treated as a mini-batch of data, and forward propagation involves the following steps:

1. Evaluate the sparse features and other associated outputs of each sensor frame (see Section 6.4.1).

2. Construct the motion prior, ϕ^p, and measurement, ϕ^m, factors for our estimator formulation.

3. The e-step: inference for the current best posterior estimate $q(\mathbf{x})$ of the mini-batch (window) of frames.

Similar to our ICP-based work in the previous chapter, we will approximate the e-step using MAP via Gauss-Newton. The reasoning is the same as before, i.e., lidar measurements are dense and highly accurate, resulting in a concentrated posterior that is reasonably approximated with a lower-order approximation of ESGVI (i.e., MAP Gauss-Newton).

For the m-step, we compute backpropagation for the network parameters, $\boldsymbol{\theta}$, on the loss functional (3.3), where only the measurement factors, ϕ^m, are affected since the motion prior factors are constant with respect to $\boldsymbol{\theta}$. We can use the spherical-cubature rule (Särkkä, 2013) to compute sigmapoints for the posterior, $q(\mathbf{x})$, in order to approximate the expectation in (3.3). Recall that we do not need to compute sigmapoints over the entire posterior, which can be expensive, but just the marginals for each factor (Barfoot et al., 2020). Training continues until the loss functional (3.3) converges. Once converged, inference can be computed on new sequences of lidar frames for odometry (i.e., the e-step).

6.4.3 Outlier Rejection

Outlier rejection is an important component to improve robustness for any estimation algorithm, and is traditionally handled with M-estimation (Zhang, 1997) in factor graph optimization. We apply M-estimation in the e-step by applying the Geman-McClure cost function on the measurement factors when optimizing with Gauss-Newton. M-estimation is applied at both train and test time.

While the robust cost function is sufficient for the e-step, we cannot apply it to the measurement factor when backpropagating to learn the inverse measurement covariances, $\mathbf{W}$, in the m-step. Instead, we apply a hard threshold on the squared Mahalanobis term in the measurement factor with the current best posterior estimate,

$$\left(\mathbf{z}_k^\ell - \mathbf{g}(\mathbf{x}_\tau, \mathbf{x}_k)\right)^T \mathbf{W}_k^\ell \left(\mathbf{z}_k^\ell - \mathbf{g}(\mathbf{x}_\tau, \mathbf{x}_k)\right) > \alpha, \tag{6.17}$$

and do not backpropagate any factor terms that exceed the threshold, α. This threshold, which we set to 4, is only applied during training at the backpropagation step.

Our keypoint detector determines the best keypoint in each voxel partition of the pointcloud. This is suboptimal for our problem formulation, as it results in keypoints in uninteresting areas (e.g., the ground plane). We can compensate at test time by judging the quality of each keypoint, $\mathbf{z}_k^\ell$, with the learned inverse measurement covariance, $\mathbf{W}_k^\ell$. Computing the sphericity metric (Thomas et al., 2018) using the eigenvalues of the measurement covariance, λ_3/λ_1, where $\lambda_1 \geq \lambda_2 \geq \lambda_3 \in \mathbb{R}$ are the eigenvalues[4] of the covariance ${\mathbf{W}_k^\ell}^{-1}$, is a potential way to judge the quality of each keypoint. However, we found the computation of the eigenvalues to be too inefficient in practice. Alternatively, we apply a metric that achieves a similar effect using the diagonal elements of $\mathbf{W}_k^\ell$ (see (6.13)). We define this metric with a threshold as

$$\left.{\exp d_{\min}}\middle/{\exp d_{\max}}\right. = \exp\left(d_{\min} - d_{\max}\right) < \beta, \tag{6.18}$$

where $d_{\min}$ and $d_{\max}$ are the smallest and largest of the diagonal elements, respectively.

[4]Eigenvalues of ${\mathbf{W}_k^\ell}^{-1}$ are the reciprocals of the eigenvalues of $\mathbf{W}_k^\ell$.

We do not use keypoints less than the threshold, β. We found through experimentation that this metric works well on planar surfaces that are axis-aligned to the sensor frame. This threshold, which we set to 0.05, is only applied in the e-step, i.e., we still backpropagate keypoints less than the threshold for covariance learning.

6.5 Experiment: Deep Features for Lidar Odometry

6.5.1 Experiment Setup

We evaluate lidar odometry on two publicly available datasets: the KITTI odometry dataset (Geiger et al., 2012) and the Oxford RobotCar dataset (Maddern et al., 2017; Barnes et al., 2020).

The KITTI odometry benchmark (Geiger et al., 2012) has 22 sequences of Velodyne HDL-64 data collected at 10 Hz. The first 11 sequences (00-10) are provided as the training set, and the remaining 11 sequences (11-21) are provided without groundtruth and act as the online public benchmark. Following existing work (Li et al., 2019; Cho et al., 2020), we split the first 11 sequences into training and testing sequences and evaluate against the provided groundtruth. We additionally submitted our estimator results into the public benchmark for a fair comparison to other existing methods.

Velodyne HDL-32 sensors were introduced to the Oxford RobotCar dataset (Maddern et al., 2017) in the radar dataset extension (Barnes et al., 2020). Two 20 Hz HDL-32 sensors were placed on the roof of the data collection vehicle. We opted for the simpler setup of only evaluating odometry using one of the two sensors. The dataset contains 30 sequences, each 9 km in length, and 2 shorter sequences. All sequences were collected from a similar driving route over a the time span of a week, and thus there is little variation for lidar data. We use 6 of the 9 km sequences in our experiments, where 2 of the 6 are used for training.

KITTI preprocessed the lidar data to account for motion distortion. While the Oxford dataset does not motion-compensate the data, we chose to not account for this effect as it is not the focus of this work, and the faster spin-rate alleviates this problem to some degree. We demonstrated motion compensation in the earlier chapter for our ICP-based odometry, and note that it is applicable to this work as well. We also applied motion compensation in our published work on unsupervised radar odometry (Burnett et al., 2021), which applies the same methodology for training the radar processing front-end.

We follow the KITTI odometry evaluation metric for all datasets, which averages the relative position and orientation errors over trajectory segments of 100 m to 800 m.

We implemented the front-end network using a KPConv implementation in PyTorch[5]. Network parameters were trained with the Adam optimizer (Kingma and Ba, 2014), and always trained from random initialization (i.e., trained from scratch). Pointclouds were augmented during training with random rotations in the z-axis for more variation to large rotations. The back-end estimator was implemented using STEAM[6], a C++ optimization library for state estimation. Loop closures were not implemented.

Our current implementation[7] is not real-time for a HDL-64, taking on average 359 ms for a window of 4 frames. KPConv is a bottleneck, taking 180 ms for pre-processing and 43 ms for forward propagation for each frame[8]. Updates are in the works to improve runtime. Data association for each frame takes 19 ms ($\times 3$ for window of 4), while Gauss-Newton takes 58 ms. Gauss-Newton is the only C++ implementation and runs on the CPU. The rest is overhead.

6.5.2 Odometry results

We train and evaluate odometry with a window of 4 frames for our experiments on the KITTI dataset. The relative pose between the latest two frames of the window are taken as the odometry output, reflecting online operation (i.e., we do not evaluate odometry performance on estimates that take future data into consideration). We compare to methods that fully learn the estimator: LO-Net (Li et al., 2019) and DeepLO (Cho et al., 2020). Since DeepLO only presents the average of sequences 00-08, Table 6.2 presents the results in the same way. DeepLO train on sequences 00-08, while we follow LO-Net and train on sequences 00-06. DeepLO does not perform as well as LO-Net, but has the advantage that it is unsupervised. Our method maintains the advantage of being unsupervised, and achieves better performance than both methods.

The uncertainty output of our estimator is in Figure 6.7, which shows the relative pose error of sequence 07 with 3σ variance envelopes. Errors are computed as

$$\boldsymbol{\xi}_{k,k-1} = [\rho_1\ \rho_2\ \rho_3\ \psi_1\ \psi_2\ \psi_3]^T = \ln\left(\mathbf{T}_{k,k-1}\mathbf{T}^{\mathrm{gt}^{-1}}_{k,k-1}\right)^{\vee}, \tag{6.19}$$

where $\mathbf{T}_{k,k-1}$ is the relative pose estimate between frames t_k and t_{k-1}, $\mathbf{T}^{\mathrm{gt}}_{k,k-1}$ is the groundtruth, $\ln(\cdot)$ is the inverse exponential map, and $\vee$ is the inverse of the $\wedge$ operator (Barfoot, 2024).

Table 6.3 compares our odometry to the state of the art for lidar odometry at the time

[5]https://github.com/HuguesTHOMAS/KPConv-PyTorch

[6]https://github.com/utiasASRL/steam

[7]On an Nvidia Tesla V100 GPU and 2.2 GHz Intel Xeon CPU.

[8]Only computed for the latest frame since previous ones are saved.

Table 6.2: A comparison of our odometry method to those that fully learn the estimator with a deep network. DeepLO (Cho et al., 2020) and our method are trained unsupervised, while LO-Net (Li et al., 2019) is trained with supervision from the groundtruth trajectory. Our method and LO-Net trained on sequences 00-06, while DeepLO trained on sequences 00-08. Using the KITTI odometry benchmark metric (Geiger et al., 2012), the average translation (%) and orientation (°/100 m) errors over lengths of 100 m to 800 m are presented. The average over sequences 00-08 are presented, as DeepLO does not present them individually. The best results are in bold.

Seq.	Ours (Unsupervised)	DeepLO (Cho et al., 2020) (Unsupervised)	LO-Net (Li et al., 2019) (Supervised)
00-08	**0.82/0.32**	3.68/0.87	1.27/0.67
09	**0.97/0.34**	4.87/1.95	1.37/0.58
10	**1.38/0.51**	5.02/1.83	1.80/0.93
Avg.	**0.89/0.34**	3.91/1.06	1.33/0.69

of publication of our work, which are ICP-based methods. Lidar Odometry and Mapping (LOAM) (Zhang and Singh, 2017) is a well-known method that often led the online KITTI benchmark leaderboard, and LO-Net+Mapping is the ICP solution presented by Li et al. (2019) that applies point masks trained with LO-Net[9] and manually computed surface normals. Overall, we demonstrate that our method is comparable to the other methods. Compared to LO-Net+Mapping, our method is learned unsupervised and does not rely on ICP for data association. Compared to LOAM, our method can easily be tuned for different platforms and lidar sensors by learning from just the on-board lidar data. Note that the orientation results for LOAM are not provided in their publication, so we omit them in our table.

Our submission to the online KITTI benchmark, with the same network and parameters, achieved 1.07 % average translation and 0.36 °/100 m average orientation error. In comparison, LOAM currently has 0.55% translation and 0.13°/100 m orientation (0.88% translation in original publication (Zhang and Singh, 2017)). Our results are more comparable to Surfel-based Mapping (SuMa) (Behley and Stachniss, 2018) (1.39 % and 0.34 °/100 m) and SuMa++ (Chen et al., 2019) (1.06 % and 0.34 °/100 m), both being well-regarded odometry methods. DeepLO and LO-Net currently do not have submissions. Considering we do not apply loop closure, which the benchmark permits, we believe our

[9]Our understanding is that the masks are trained without supervision from mask targets, but with supervision from groundtruth trajectory (Li et al., 2019).

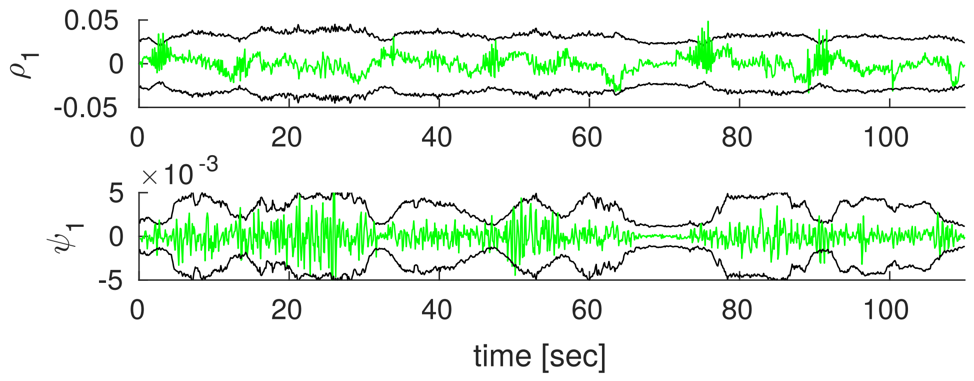


Figure 6.7: Odometry error of KITTI sequence 07 with 3σ variance envelopes. In the interest of space, only two dimensions, ρ_1 and ψ_1, are shown (See (6.19)). Our estimator is in general consistent, but at times slightly overconfident.

method achieved reasonable performance.

Recently, our own published work on continuous-time ICP-based odometry (Wu et al., 2023) achieved results more comparable to the latest state-of-the-art results of LOAM on the publicly available sequences[10]. An aspect to lidar odometry from our latest work that we do not apply here is building and maintaining a pointcloud submap that we optimize (register) the latest lidar frames to (i.e., the difference between map-to-frame optimization and frame-to-frame). From our experience working with ICP, building and maintaining a high-quality submap is a key component in greatly reducing odometry drift. We believe that incorporating a submap is possible in our proposed method with a DNN front-end given our application of pointcloud convolutions in our network architecture, and propose pursuing this avenue in future work.

We demonstrate the capability of our method for automated tuning by training and testing on the Oxford dataset, and comparing it to performance of the network trained on KITTI dataset (i.e., the trained network applied in Tables 6.2 and 6.3). We optimize over a window of 7 frames and use the lidar data at 10 Hz (i.e., skip every other frame), settings with which the KITTI trained network performed best on the Oxford data. The same settings were applied when training a new network on the Oxford dataset.

We present qualitative results of our odometry in Figure 6.8, which shows plots of the odometry paths for sequence 2019-01-15-14-24-38. Interestingly, we see that the performance when using a network trained on the KITTI dataset is still reasonable. However, there is a clear improvement in odometry when we train on the Oxford data, which is expected. We additionally show the performance with an untrained network (i.e., randomly initialized weights), which is clearly incapable of performing odometry.

[10]We did not submit our results to the online benchmark.

Table 6.3: A comparison of our method to existing lidar odometry methods at the time of publication. Using the KITTI odometry benchmark metric (Geiger et al., 2012), the average translation (%) and orientation (°/100 m) errors over lengths of 100 m to 800 m are presented. The best results are in bold.

Seq.	Ours (Unsupervised)	LO-Net+Mapping (ICP)(Li et al., 2019) (Supervised)	LOAM (Zhang and Singh, 2017) (Non-Learned)
00†	0.92/**0.39**	**0.78**/0.42	**0.78**/-
01†	**1.30**/**0.28**	1.42/0.40	1.43/-
02†	1.11/**0.42**	1.01/0.45	**0.92**/-
03†	0.77/**0.38**	**0.73**/0.59	0.86/-
04†	0.62/**0.22**	**0.56**/0.54	0.71/-
05†	0.68/**0.30**	0.62/0.35	**0.57**/-
06†	**0.50**/**0.17**	0.55/0.33	0.65/-
07*	**0.49**/**0.33**	0.56/0.45	0.63/-
08*	**1.01**/**0.36**	1.08/0.43	1.12/-
09*	0.97/**0.34**	**0.77**/0.38	**0.77**/-
10*	1.38/0.51	0.92/**0.41**	**0.79**/-
Avg.	0.89/**0.34**	**0.82**/0.43	0.84/-

†: Sequences that our method and LO-Net train on.
*: Sequences that are not used for training.

Table 6.4 shows the quantitative results of odometry, which show a clear improvement when parameters are trained with data related to the deployment.

6.5.3 Ablation Study

Table 6.5 shows the results of an ablation study, where we remove various components of our method. In addition to the KITTI benchmark metrics for translation and orientation, we compute an average Mahalanobis distance metric (Brossard et al., 2020b),

$$\left(\sum_{k=1}^{K} \frac{\boldsymbol{\xi}_{k,k-1}^T \mathbf{Q}_{k,k-1}^{-1} \boldsymbol{\xi}_{k,k-1}}{\dim(\boldsymbol{\xi}_{k,k-1}) K} \right)^{1/2}, \tag{6.20}$$

where $\boldsymbol{\xi}_{k,k-1}$ is the error as defined in (6.19) and $\mathbf{Q}_{k,k-1}$ is the corresponding covariance. A value close to 1 is ideal for this metric.

'No Sampling' refers to evaluating the expectation in the loss functional (3.3) at only the mean of the posterior in the m-step. We see that approximating the expectation with

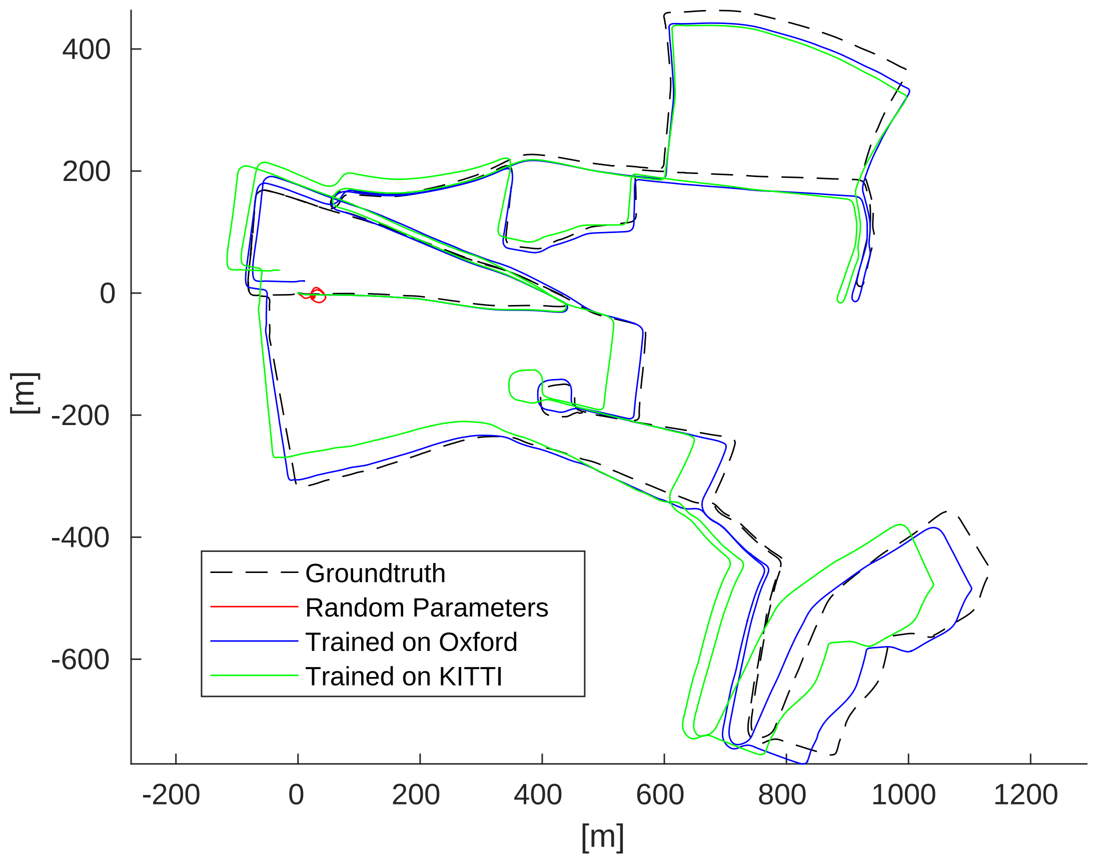


Figure 6.8: Odometry paths for sequence 2019-01-15-14-24-38 of the Oxford RobotCar dataset (Maddern et al., 2017; Barnes et al., 2020) with a Velodyne HDL-32. We compare against the performance when using a network trained on a different dataset, KITTI (Geiger et al., 2012), which is a Velodyne HDL-64. Odometry fails prematurely due to numerical instability when the network is not trained (red).

sigmapoints is insignificant for this problem, which is consistent with the approximation made in the e-step. 'No β' and 'No α' refers to not applying the β and α thresholds in Section 6.4.3. It is clear that the performance is worse in translation and orientation for both configurations. The exception is the Mahalanobis metric for 'No α', which performs the most consistently and more conservatively than the rest. Backpropagating outliers means the learned uncertainties must account for them, which may explain why the estimator became more conservative.

Table 6.4: Odometry results for our method on Velodyne HDL-32 data of the Oxford RobotCar dataset (Maddern et al., 2017; Barnes et al., 2020). Using the KITTI odometry benchmark metric Geiger et al. (2012), the average translation (%) and orientation (°/100 m) errors over lengths of 100 m to 800 m are presented. The best results are in bold.

Seq.	Trained on Oxford	Trained on KITTI
2019-01-10-11-46-21†	**2.85/1.29**	3.21/1.55
2019-01-18-15-20-12†	**2.41/1.13**	3.04/1.47
2019-01-15-13-06-37*	**2.48/1.20**	2.89/1.43
2019-01-15-14-24-38*	**2.60/1.25**	2.95/1.51
2019-01-16-13-09-37*	**2.56/1.20**	3.15/1.48
2019-01-16-14-15-33*	**2.99/1.47**	3.60/1.76
Avg.	**2.65/1.26**	3.14/1.53

†: Sequences that we train on (applicable only to 'Trained on Oxford').
*: Sequences that are not used for training.

Table 6.5: An ablation study over components of our method on the KITTI odometry dataset. Using the KITTI odometry benchmark metric (Geiger et al., 2012), the average translation (%) and orientation (°/100 m) errors over lengths of 100 m to 800 m are presented. We additionally compute the average squared Mahalanobis distance (third metric in each column) of the relative pose estimates (see (6.20)), which ideally is 1 for a consistent estimator.

Seq.	Full method	No Sampling	No β	No α
07	0.49/0.33/1.22	**0.48**/**0.29**/1.23	0.61/0.38/1.67	1.22/0.96/**1.06**
08	1.01/0.36/2.66	**0.96**/**0.33**/2.71	1.17/0.45/6.98	2.23/0.84/**2.09**
09	**0.97**/**0.34**/1.31	0.98/0.36/1.34	1.36/0.56/1.98	2.44/0.92/**1.14**
10	**1.38**/**0.51**/1.54	1.56/0.58/1.55	2.13/0.84/2.09	2.13/1.58/**1.28**
Avg.	**0.96**/**0.38**/1.68	0.99/0.39/1.71	1.32/0.56/3.18	2.00/1.07/**1.39**

6.6 Summary and Conclusions

We presented in this chapter applications of DNNs in our parameter learning framework with ESGVI and EM. Our motivation in doing so was to incorporate a richer learning model (in contrast to the work in Chapter 5) for our sensor measurements (i.e., a front-end), while maintaining a probabilistic state estimation back-end. We presented two types of models: the first being a network that learns the measurement model in its entirety by mapping the input state to the output measurement, and the second being a front-end processor of dense, rich sensor data that outputs sparse features. For the former model type, we presented experimental results on a localization problem and learned a range-bearing measurement model without groundtruth. For the latter model type, we presented experimental results on lidar odometry, which appeared in Yoon et al. (2021).

In summary, the contributions of this chapter are:

1. Learning a DNN measurement model using EM and ESGVI without having to compute the derivative (Jacobian) of the model with respect to the state.

2. Experiments on a real-robot dataset for localization and learning a range-bearing measurement model without the groundtruth trajectory.

3. Learning a DNN using EM and ESGVI that is specifically tailored to rich sensor data via a CNN architecture that outputs sparse keypoints, descriptors, and uncertainty.

4. Experiments on lidar odometry, again demonstrating that our parameter learning framework can train the network parameters without the groundtruth trajectory.

There are several avenues for further exploration of this idea of using EM to train network parameters, which we discuss in the following chapter.

Chapter 7

Conclusion

In this chapter, we provide a short summary of the contributions and publications that arose from the work presented in this thesis. We also provide some narrative on directions for future work in the area of estimation and parameter learning for vehicle trajectory estimation.

7.1 Summary of Contributions

In the first major chapter of this thesis (Chapter 3), we presented a Gaussian Variational Inference (GVI) approach to batch state estimation that is computationally tractable for large-scale estimation problems, which we call Exactly Sparse Gaussian Variational Inference (ESGVI). The motivation was to develop an estimator that finds a 'closer' fit to the full Bayesian posterior PDF, in contrast to MAP that optimizes the posterior approximation as a point estimate.

The methods and experiments for ESGVI that were presented in Chapter 3 appeared in the publication:

- Barfoot, T. D., Forbes, J. R., and Yoon, D. J. (2020). Exactly sparse gaussian variational inference with application to derivative-free batch nonlinear state estimation. *International Journal of Robotics Research (IJRR) (second author contribution)*

In summary, the contributions of Chapter 3 are:

1. A computationally tractable approach to Gaussian Variational Inference (GVI) for large-scale estimation problems via exploitation of the sparsity formed from the factorization of the joint likelihood.

2. An implementation of sparse GVI that uses Gaussian cubature to avoid the need for computing analytical derivatives.

3. A conservative, cheaper (compute) approximation of Gaussian Variational Inference (GVI) that is applicable under mild nonlinearities and/or when the posterior is concentrated.

4. Various experiments in both simulation and on real-data demonstrating an improvement in performance over MAP.

While the motivation behind our initial work on ESGVI was to improve upon existing estimation tools, such as MAP, we found interest in the application of ESGVI to parameter learning due to its synergy with Expectation-Maximization (EM). We were motivated to apply parameter learning to facilitate data-driven model tuning, rather than requiring an expert engineer to design and tune models for each different deployment. The remaining chapters of this thesis were dedicated to applications of our ESGVI and EM parameter learning framework.

In Chapter 4, we presented a methodology for modelling a varying noise model by placing an Inverse-Wishart prior on the measurement covariance. This approach treats the measurement covariance as part of the state and not a parameter. A useful outcome of this work was robustness to measurement outliers similar to the application of a robust cost function. We presented the methodology and experiments in a second-authored publication:

- Wong, J. N., Yoon, D. J., Schoellig, A. P., and Barfoot, T. D. (2020b). Variational inference with parameter learning applied to vehicle trajectory estimation. *IEEE Robotics and Automation Letters (RAL)*, 5(4):5291–5298 *(second author contribution)*

In summary, the contributions of Chapter 4 are:

1. A methodology for estimating measurement covariance by using an IW prior in a EM framework.

2. Experimental results on a lidar localization dataset that demonstrates learning a varying measurement covariance without the groundtruth trajectory. We also show that the resulting covariance model is robust to measurement outliers during both training and testing.

Our focus in Chapter 5 was on learning measurement bias and noise models using feature-dependent regression models. Due to nonidealities in the real world, measurements can be biased in comparison to our known sensor models and their uncertainty may require a richer noise model that can vary depending on the application setting. We propose modelling bias and covariance using feature-dependent regression models. The focus, however, was not on the particular choice of regression model we chose to use, but rather the application of training the models using EM without the groundtruth trajectory. We demonstrated application on odometry using FMCW lidar sensors, where we trained our models using only the observed sensor data. The odometry estimators we used in our work, a ICP-based odometry method and a fast Correspondence-Free (CF)-based odometry method using Doppler measurements, appeared previously in two publications:

- Wu, Y., Yoon, D. J., Burnett, K., Kammel, S., Chen, Y., Vhavle, H., and Barfoot, T. D. (2023). Picking up speed: Continuous-time lidar-only odometry using doppler velocity measurements. *IEEE Robotics and Automation Letters (RAL)*, 8(1):264–271 *(second author contribution)*

- Yoon, D. J., Burnett, K., Laconte, J., Chen, Y., Vhavle, H., Kammel, S., Reuther, J., and Barfoot, T. D. (2023). Need for speed: Fast correspondence-free lidar-inertial odometry using doppler velocity. In *International Conference on Intelligent Robots and Systems (IROS)*

In summary, the contributions of Chapter 5 are:

1. A methodology for training feature-dependent regression models for measurement bias and covariance using ESGVI and EM.

2. Experiments using the proposed method on odometry using a FMCW lidar, demonstrating the ability to train the regression models without supervision from the groundtruth trajectory.

In Chapter 6, we continued the application of our ESGVI and EM parameter learning framework, but enriched the framework by incorporating Deep Neural Networks (DNNs). We first proposed modelling the measurement model in its entirety using a DNN, which utilized our derivative-free optimization of ESGVI to avoid the need to compute Jacobians of the learned model. Numerical experiments were conducted and presented in Section 6.3. We then proposed an alternative DNN model using a CNN architecture that processes rich sensor data into sparse features. Applications of our method to lidar odometry and radar odometry appeared in the following publications:

- Yoon, D. J., Zhang, H., Gridseth, M., Thomas, H., and Barfoot, T. D. (2021). Unsupervised learning of lidar features for use in a probabilistic trajectory estimator. *IEEE Robotics and Automation Letters (RAL)*, 6(2):2130–2138
- Burnett, K., Yoon, D. J., Schoellig, A. P., and Barfoot, T. D. (2021). Radar odometry combining probabilistic estimation and unsupervised feature learning. In *Robotics: Science and Systems (RSS) (equal contribution between Burnett, K. and Yoon, D.)*

The contents in this thesis chapter focused on the lidar odometry problem. We briefly discussed the methodology for radar odometry and refer readers to the corresponding publication for the experiments. In summary, the contributions of Chapter 6 are:

1. Learning a DNN measurement model using EM and ESGVI without having to compute the derivative (Jacobian) of the model with respect to the state.
2. Experiments on a real-robot dataset for localization and learning a range-bearing measurement model without the groundtruth trajectory.
3. Learning a DNN using EM and ESGVI that is specifically tailored to rich sensor data via a CNN architecture that outputs sparse keypoints, descriptors, and uncertainty.
4. Experiments on lidar odometry, again demonstrating that our parameter learning framework can train the network parameters without the groundtruth trajectory.

7.2 Future Work

For future work regarding ESGVI, we suggest the following:

1. Computational improvements to the implementation of ESGVI. A few avenues of interest regarding this are: parallel computation of sigmapoints (possibly on a GPU), variable reordering and other schemes such as Givens rotations (Golub and Van Loan, 1996) in combination with the Takahashi method (Takahashi et al., 1973) for efficiently computing blocks of the posterior covariance, and modifications for online implementation (e.g., sliding-window filter).
2. An extension of the variational approach to multi-modal estimation problems, e.g., a mixture of Gaussians, in contrast to the single multivariate Gaussian we have shown in our work.

For future work regarding learning covariances and biases, we suggest the following:

1. For our learned covariance models, we made the assumption that measurements from different times are corrupted by statistically independent noise. Extending our methods (both prior-based and feature-based) to handle time-correlated noise and being able to train them without the groundtruth trajectory is of interest. We recently worked on a feature-based approach to modelling a time-correlated measurement covariance trained using the groundtruth trajectory (Yoon and Barfoot, 2023). We believe extending this work using EM to train without groundtruth is a promising avenue for future work.

2. Our CF-based odometry method using multiple sensors does not perform accurately without the gyroscope measurements. Incorporating a richer regression model for compensating measurement bias and covariance could be beneficial. For that, we can look to applying aspects of our work we explored in Chapter 6 on DNNs.

Finally, for future work on applying DNNs to our estimation and parameter learning framework, we suggest the following:

1. Recent works on neural radiance fields demonstrate the viability of applying a multi-layer perception to model a rich *implicit* representation of a given scene (Mildenhall et al., 2021). Combining ideas from this area of machine learning with the ideas we presented on learning a DNN measurement model can be an interesting, viable way of applying DNNs to state estimation with rich sensor data.

2. The performance of lidar odometry using our learned sparse features was once competitive to the state of the art at the time of publication, but now falls short in comparison to recent improvements in lidar odometry. Current state of the art odometry methods maintain and update a pointcloud submap that is used as the reference pointcloud for alignment. Our method, which uses pointcloud convolutions using KPConv (Thomas et al., 2019), should be capable of handling a submap and we hypothesize that this addition will bring our method closer to the current state of the art.

Appendix A

Doppler Derivations

A.1 Doppler Measurement Model

Here we show a derivation of our Doppler measurement model, which was presented in our publication (Wu et al., 2023) (second-authored contribution). We first define $\dot{\mathbf{q}}$ and $\dot{\mathbf{q}}_i$ to be $\mathbf{q}$'s velocity expressed in the lidar and inertial reference frame, respectively. Applying the *transport theorem*,

$$\dot{\mathbf{q}}_i = \mathbf{T}_{i\ell}(t)\left(\dot{\mathbf{q}} - \boldsymbol{\varpi}_{\ell}^{i\ell}(t)^{\wedge}\mathbf{q}\right). \tag{A.1}$$

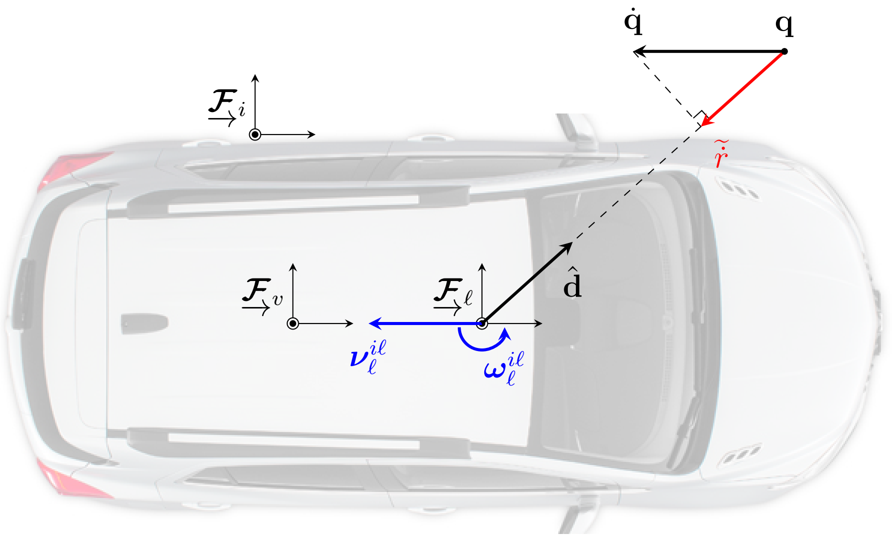
Assuming $\mathbf{q}$ is static in the inertial frame such that $\dot{\mathbf{q}}_i = 0$, we can rearrange (A.1) to be (see $SE(3)$ identities in (Barfoot, 2024))

$$\begin{aligned}\dot{\mathbf{q}} &= \boldsymbol{\varpi}_{\ell}^{i\ell}(t)^{\wedge}\mathbf{q} \\ &= \mathbf{q}^{\odot}\boldsymbol{\varpi}_{\ell}^{i\ell}(t) \\ &= \mathbf{q}^{\odot}\boldsymbol{\mathcal{T}}_{\ell v}\boldsymbol{\varpi}_{v}^{iv}(t),\end{aligned} \tag{A.2}$$

where (A.2) relates the point velocity $\dot{\mathbf{q}}$ to the body-centric velocity component of our trajectory state $\boldsymbol{\varpi}_{v}^{iv}$. Next, we project $\dot{\mathbf{q}}$ onto $\mathbf{q}$ to obtain the predicted Doppler velocity:

$$\tilde{\breve{r}} = \underbrace{\frac{\mathbf{q}^T\mathbf{D}^T}{\left(\mathbf{q}^T\mathbf{D}^T\mathbf{D}\mathbf{q}\right)^{1/2}}}_{\text{Projection } \hat{\mathbf{d}}}\mathbf{D}\mathbf{q}^{\odot}\boldsymbol{\mathcal{T}}_{\ell v}\boldsymbol{\varpi}_{v}^{iv}(t), \tag{A.3}$$

where the projection, $\hat{\mathbf{d}}$, is simply a unit vector along the direction of $\mathbf{q}$. This derivation is graphically illustrated in Figure A.1.



$\dot{\mathbf{q}}$
$\mathbf{q}$
$\underrightarrow{\mathcal{F}}_i$
$\tilde{r}$
$\underrightarrow{\mathcal{F}}_v$
$\underrightarrow{\mathcal{F}}_\ell$
$\hat{\mathbf{d}}$
$\boldsymbol{\nu}_\ell^{i\ell}$
$\boldsymbol{\omega}_\ell^{i\ell}$

Figure A.1: This diagram provides a graphical illustration of the Doppler velocity error term derivation. $\underrightarrow{\mathcal{F}}_i$, $\underrightarrow{\mathcal{F}}_v$, and $\underrightarrow{\mathcal{F}}_\ell$ are the inertial, vehicle, and lidar reference frames, respectively. $\mathbf{q}$ and $\dot{\mathbf{q}}$ are the measured point position and velocity expressed in $\underrightarrow{\mathcal{F}}_\ell$, respectively. $\hat{\mathbf{d}}$ is a unit vector along the direction of $\mathbf{q}$, and $\tilde{r}$ is the relative radial velocity according to $\boldsymbol{\varpi}_\ell^{i\ell} = \begin{bmatrix} \boldsymbol{\nu}_\ell^{i\ell T} & \boldsymbol{\omega}_\ell^{i\ell T} \end{bmatrix}^T$.

A.2 Doppler Observability Study

A.2.1 Observability Study - Multiple FMCW Lidars

We present an observability study for the 6-DOF vehicle velocity using Doppler measurements from multiple FMCW lidars, which we first presented in our published work (Yoon et al., 2023). In order to simplify the proof, we focus on estimating the vehicle velocity over the interval of one lidar frame, assuming that the data from multiple lidars are synchronized and each have m measurements. We also remove the continuous-time aspect of the problem by assuming the vehicle velocity is constant throughout the frame duration.

For the i^{th} measurement seen by the j^{th} sensor,

$$\begin{aligned} e_{\text{dv}}^{ij} &= y_{\text{dv}}^{ij} - \frac{1}{(\mathbf{q}_j^{ij^T}\mathbf{q}_j^{ij})^{\frac{1}{2}}} \begin{bmatrix} \mathbf{q}_j^{ij^T} & \mathbf{0}^T \end{bmatrix} \boldsymbol{\mathcal{T}}_{jv} \boldsymbol{\varpi} \\ &= y_{\text{dv}}^{ij} - \frac{1}{(\mathbf{q}_j^{ij^T}\mathbf{q}_j^{ij})^{\frac{1}{2}}} \begin{bmatrix} \mathbf{q}_j^{ij^T} \boldsymbol{R}_{jv} & \mathbf{q}_j^{ij^T} \boldsymbol{t}_j^{vj\wedge} \boldsymbol{R}_{jv} \end{bmatrix} \boldsymbol{\varpi} \\ &= y_{\text{dv}}^{ij} - \boldsymbol{c}_{ij}^T \boldsymbol{\varpi}, \end{aligned} \tag{A.4}$$

where the additional superscript j indicates the sensor[1], $\boldsymbol{\mathcal{T}}_{sv} \in \mathrm{Ad}\left(SE(3)\right)$ is the extrinsic adjoint transformation between the jth sensor and vehicle frames, and

$$\boldsymbol{c}_{ij}^T = \begin{bmatrix} \hat{\mathbf{q}}_v^{ij^T} & \hat{\mathbf{q}}_v^{ij^T} \boldsymbol{t}_v^{vj\wedge} \end{bmatrix}, \quad \hat{\mathbf{q}}_v^{ij} = \frac{\boldsymbol{R}_{jv}^T \mathbf{q}_j^{ij}}{\|\mathbf{q}_j^{ij}\|}. \tag{A.5}$$

Note how the measurement model does not depend on the magnitude (range) of $\mathbf{q}$ since $\hat{\mathbf{q}}$ are unit vectors. We define the stacked quantity $\boldsymbol{C}_j = [\boldsymbol{c}_{1j} \cdots \boldsymbol{c}_{mj}]^T$ for sensor j. In the case of N sensors, we have

$$\boldsymbol{C}^T\boldsymbol{C} = \sum_{j=1}^{N} \boldsymbol{C}_j^T \boldsymbol{C}_j = \sum_j \begin{bmatrix} \boldsymbol{Q}_j & \boldsymbol{Q}_j \boldsymbol{t}_v^{vj\wedge} \\ \boldsymbol{t}_v^{vj\wedge^T} \boldsymbol{Q}_j & \boldsymbol{t}_v^{vj\wedge^T} \boldsymbol{Q}_j \boldsymbol{t}_v^{vj\wedge} \end{bmatrix}, \tag{A.6}$$

where $\boldsymbol{Q}_j = \sum_i \hat{\mathbf{q}}_v^{ij} \hat{\mathbf{q}}_v^{ij^T}$ is the sum of the outer product of the points seen by the jth sensor. The velocity is fully observable from a single lidar frame if and only if $\boldsymbol{C}^T\boldsymbol{C}$ is full rank, or equivalently that the nullspace of $\boldsymbol{C}^T\boldsymbol{C}$ has dimension zero (Barfoot, 2024). In the following, we assume $\boldsymbol{Q}_j$ to be full rank (best case scenario), meaning that the unit velocities seen by the jth sensor are not all contained in a line or a plane. In the case of a 3D lidar sensor, $\boldsymbol{Q}_j$ will always be full rank regardless of the environment geometry.

Lemma 1 *Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be two symmetric positive semidefinite matrices. Then, we have*

$$\mathrm{null}\left(\boldsymbol{A}+\boldsymbol{B}\right) = \mathrm{null}\left(\boldsymbol{A}\right) \cap \mathrm{null}\left(\boldsymbol{B}\right).$$

Assume $\boldsymbol{x} \in \mathrm{null}\left(\boldsymbol{A}\right) \cap \mathrm{null}\left(\boldsymbol{B}\right)$. It is straightforward to see that $\boldsymbol{x} \in \mathrm{null}\left(\boldsymbol{A}+\boldsymbol{B}\right)$, hence $\mathrm{null}\left(\boldsymbol{A}+\boldsymbol{B}\right) \supseteq \mathrm{null}\left(\boldsymbol{A}\right) \cap \mathrm{null}\left(\boldsymbol{B}\right)$. Then, assume $\boldsymbol{x} \in \mathrm{null}\left(\boldsymbol{A}+\boldsymbol{B}\right)$, we have

$$\boldsymbol{x}^T(\boldsymbol{A}+\boldsymbol{B})\boldsymbol{x} = \underbrace{\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}}_{\geq 0} + \underbrace{\boldsymbol{x}^T\boldsymbol{B}\boldsymbol{x}}_{\geq 0} = 0$$

Recall that for any symmetric PSD matrix $\boldsymbol{M}$, we have $\boldsymbol{M} = \boldsymbol{S}^T\boldsymbol{S}$. As such, $\boldsymbol{x}^T\boldsymbol{M}\boldsymbol{x} = 0 \Leftrightarrow \left(\boldsymbol{S}\boldsymbol{x}\right)^T \boldsymbol{S}\boldsymbol{x} = 0 \Rightarrow \boldsymbol{x} \in \mathrm{null}\left(\boldsymbol{S}\right) \Rightarrow \boldsymbol{x} \in \mathrm{null}\left(\boldsymbol{M}\right)$. Therefore, $\boldsymbol{x} \in \mathrm{null}\left(\boldsymbol{A}\right) \cap \mathrm{null}\left(\boldsymbol{B}\right)$ and $\mathrm{null}\left(\boldsymbol{A}+\boldsymbol{B}\right) \subseteq \mathrm{null}\left(\boldsymbol{A}\right) \cap \mathrm{null}\left(\boldsymbol{B}\right)$, which concludes the proof.

First, note that each member can be factorized as

$$\boldsymbol{C}_j^T\boldsymbol{C}_j = \underbrace{\begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \boldsymbol{t}_v^{vj\wedge^T} & \mathbf{1} \end{bmatrix}}_{\text{full rank}} \underbrace{\begin{bmatrix} \boldsymbol{Q}_j & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}}_{\text{PSD, rank=3}} \underbrace{\begin{bmatrix} \mathbf{1} & \boldsymbol{t}_v^{vj\wedge} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}}_{\text{full rank}}, \tag{A.7}$$

[1] $\mathbf{q}_f^{ij}$ are the coordinates of the ith point from sensor jth, in frame f.

thus being positive semidefinite. Using Lemma 1, we find the nullspace of $\boldsymbol{C}^T\boldsymbol{C}$ using the nullspace of each member of the sum. The nullspace of $\boldsymbol{C}_j^T\boldsymbol{C}_j$ is

$$\begin{aligned}\operatorname{null}\left(\boldsymbol{C}_j^T\boldsymbol{C}_j\right) &= \operatorname{null}\left(\begin{bmatrix}\boldsymbol{Q}_j & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{0}\end{bmatrix}\begin{bmatrix}\boldsymbol{1} & \boldsymbol{t}_v^{vj\wedge}\\ \boldsymbol{0} & \boldsymbol{1}\end{bmatrix}\right)\\ &= \left\{\begin{bmatrix}\boldsymbol{1} & \boldsymbol{t}_v^{vj\wedge}\\ \boldsymbol{0} & \boldsymbol{1}\end{bmatrix}^{-1}\begin{bmatrix}\boldsymbol{0}\\ \boldsymbol{k}\end{bmatrix}, \boldsymbol{k}\in\mathbb{R}^3\right\}\\ &= \left\{\begin{bmatrix}-\boldsymbol{t}_v^{vj\wedge}\boldsymbol{k}\\ \boldsymbol{k}\end{bmatrix}, \boldsymbol{k}\in\mathbb{R}^3\right\}.\end{aligned} \tag{A.8}$$

Thus for one sensor, $\boldsymbol{C}^T\boldsymbol{C}$ is rank deficient by 3. For two sensors, the nullspace of $\boldsymbol{C}_1^T\boldsymbol{C}_1 + \boldsymbol{C}_2^T\boldsymbol{C}_2$ is

$$\begin{aligned}&\left\{\begin{bmatrix}-\boldsymbol{t}_v^{v1\wedge}\boldsymbol{k}\\ \boldsymbol{k}\end{bmatrix}, \boldsymbol{k}\in\mathbb{R}^3\right\}\cap\left\{\begin{bmatrix}-\boldsymbol{t}_v^{v2\wedge}\boldsymbol{l}\\ \boldsymbol{l}\end{bmatrix}, \boldsymbol{l}\in\mathbb{R}^3\right\}\\ &= \begin{cases}\left\{\begin{bmatrix}\alpha\boldsymbol{t}_v^{v2\wedge}\boldsymbol{t}_v^{v1}\\ \alpha(\boldsymbol{t}_v^{v2}-\boldsymbol{t}_v^{v1})\end{bmatrix}, \alpha\in\mathbb{R}\right\} & \text{if } \boldsymbol{t}_v^{v1}\neq\boldsymbol{t}_v^{v2}\\ \left\{\begin{bmatrix}-\boldsymbol{t}_v^{v1\wedge}\boldsymbol{k}\\ \boldsymbol{k}\end{bmatrix}, \boldsymbol{k}\in\mathbb{R}^3\right\} & \text{if } \boldsymbol{t}_v^{v1}=\boldsymbol{t}_v^{v2},\end{cases}\end{aligned} \tag{A.9}$$

therefore being of dimension 1 if the two sensors are not at the same position, and dimension 3 otherwise. Adding a third sensor, we obtain

$$\begin{aligned}&\left\{\begin{bmatrix}\alpha\boldsymbol{t}_v^{v2\wedge}\boldsymbol{t}_v^{v1}\\ \alpha(\boldsymbol{t}_v^{v2}-\boldsymbol{t}_v^{v1})\end{bmatrix}, \alpha\in\mathbb{R}\right\}\cap\left\{\begin{bmatrix}-\boldsymbol{t}_v^{v3\wedge}\boldsymbol{k}\\ \boldsymbol{k}\end{bmatrix}, \boldsymbol{k}\in\mathbb{R}^3\right\} =\\ &\left\{\begin{bmatrix}\alpha\boldsymbol{t}_v^{v2\wedge}\boldsymbol{t}_v^{v1}\\ \alpha(\boldsymbol{t}_v^{v2}-\boldsymbol{t}_v^{v1})\end{bmatrix}\middle| \boldsymbol{t}_v^{v2\wedge}\boldsymbol{t}_v^{v1} = -\boldsymbol{t}_v^{v3\wedge}\left(\boldsymbol{t}_v^{v2}-\boldsymbol{t}_v^{v1}\right), \alpha\in\mathbb{R}\right\}.\end{aligned} \tag{A.10}$$

Looking at the condition, we remark that $\boldsymbol{t}_v^{v2\wedge}\boldsymbol{t}_v^{v1}, -\boldsymbol{t}_v^{v3\wedge}\left(\boldsymbol{t}_v^{v2}-\boldsymbol{t}_v^{v1}\right) \in \boldsymbol{t}_v^{v2\perp}\cap\boldsymbol{t}_v^{v3\perp}$ is necessary, where $(\cdot)^\perp$ denotes the orthogonal complement. As such, we can re-write the condition as

$$\begin{aligned}
&\begin{cases} \boldsymbol{t}_v^{v2\wedge}\boldsymbol{t}_v^{v1} & = \beta \boldsymbol{t}_v^{v2\wedge}\boldsymbol{t}_v^{v3}, \quad \beta \in \mathbb{R} \\ -\boldsymbol{t}_v^{v3\wedge}\left(\boldsymbol{t}_v^{v2} - \boldsymbol{t}_v^{v1}\right) & = \beta \boldsymbol{t}_v^{v2\wedge}\boldsymbol{t}_v^{v3} \end{cases} \\
\Leftrightarrow &\begin{cases} \boldsymbol{t}_v^{v1} & = \gamma \boldsymbol{t}_v^{v2} + \beta \boldsymbol{t}_v^{v3}, \quad \gamma \in \mathbb{R} \\ \beta & = 1 - \gamma \end{cases} \\
\Leftrightarrow &\ \boldsymbol{t}_v^{v1} = \gamma \boldsymbol{t}_v^{v2} + (1-\gamma)\boldsymbol{t}_v^{v3}.
\end{aligned} \tag{A.11}$$

The nullspace of $\boldsymbol{C}^T\boldsymbol{C}$ for three sensors has dimension 1 if all three $\boldsymbol{t}_v^{vj}$ are on the same line, and dimension 0 otherwise. As such, the full state is observable as long as the three lidar sensors form the vertices of a triangle. Note that using induction with Lemma 1, we can intuitively add more sensors and the system remains fully observable.

A.2.2 Observability Study - Single FMCW Lidar + Gyroscope

Considering now the case of one lidar with one gyroscope measurement, we show that adding the gyroscope measurement leads to a fully observable system. With one gyroscope measurement, the measurement matrix becomes

$$\boldsymbol{C}'^T = \begin{bmatrix} \hat{\mathbf{q}}_v^1 & \cdots & \hat{\mathbf{q}}_v^N & \mathbf{0} \\ \boldsymbol{t}_v^{vj\wedge^T}\hat{\mathbf{q}}_v^1 & \cdots & \boldsymbol{t}_v^{vj\wedge^T}\hat{\mathbf{q}}_v^N & \boldsymbol{R}_{sv} \end{bmatrix}, \tag{A.12}$$

leading to

$$\boldsymbol{C}'^T\boldsymbol{C}' = \underbrace{\begin{bmatrix} \mathbf{1} & \mathbf{0} \\ \boldsymbol{t}_v^{vj\wedge^T} & \mathbf{1} \end{bmatrix}}_{\text{full rank}} \underbrace{\begin{bmatrix} \boldsymbol{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}}_{\text{full rank}} \underbrace{\begin{bmatrix} \mathbf{1} & \boldsymbol{t}_v^{vj\wedge} \\ \mathbf{0} & \mathbf{1} \end{bmatrix}}_{\text{full rank}}. \tag{A.13}$$

As such, the system becomes fully observable with only one lidar sensor and one gyroscope.

Appendix B

Supplementary Material for Doppler Experiments

B.1 Vehicle speed Input Feature

Here, we discuss the speed of the vehicle as an input feature. The speed of the vehicle may seem problematic as the vehicle velocity is the quantity that we seek to estimate. To represent the vehicle speed as an input, we calculate the median radial velocity for each lidar frame. The median is simple to calculate and gives us an estimate of the speed that is robust to outliers (e.g., from other moving objects in the environment). We calculate the median using the uncompensated, biased measurements which prevents a negative feedback loop in our learning approach, while still representing the motion of the vehicle. Alternatively we can calculate a speed estimate using the uncompensated measurements and RANSAC using a little more compute.

B.2 Pseudo-variance Input Feature

We add one additional input feature for the variance model based on the relative variance of the Doppler measurements in azimuth space. Doppler measurements do not differ greatly with a small change in direction since they are simply the projection of the vehicle velocity onto the corresponding radial direction. We exploit this aspect by calculating a *pseudo-variance* of the measurements using a local azimuth neighbourhood and the sample variance equation,

$$\rho_k^{ae} = \sum_{\substack{j=a-\ell \\ j\neq a}}^{a+\ell} \frac{1}{2\ell} \left(y_k^{je} - y_k^{ae} \right)^2, \tag{B.1}$$

where y_k^{ae} is the Doppler measurement in lidar frame, k, at the azimuth bin, a, and elevation bin, e. Instead of the mean, which we cannot calculate without the groundtruth, we use the Doppler measurement at the center, y_k^{ae}. We choose not to include neighbouring elevation bins since those measurements are physically from different beams. We keep the neighbourhood small, e.g., we set $\ell = 5$. Figure 5.3 shows an example visualization of our pseudo-variance input feature.

B.3 More Qualitative Examples

In this section of the Appendix, we show additional figures from our FMCW lidar odometry experiments. These figures show more qualitative examples of the Doppler bias and variance predictions using our trained models. We show examples from both the Boreas FMCW dataset and the Aeva HQ dataset.

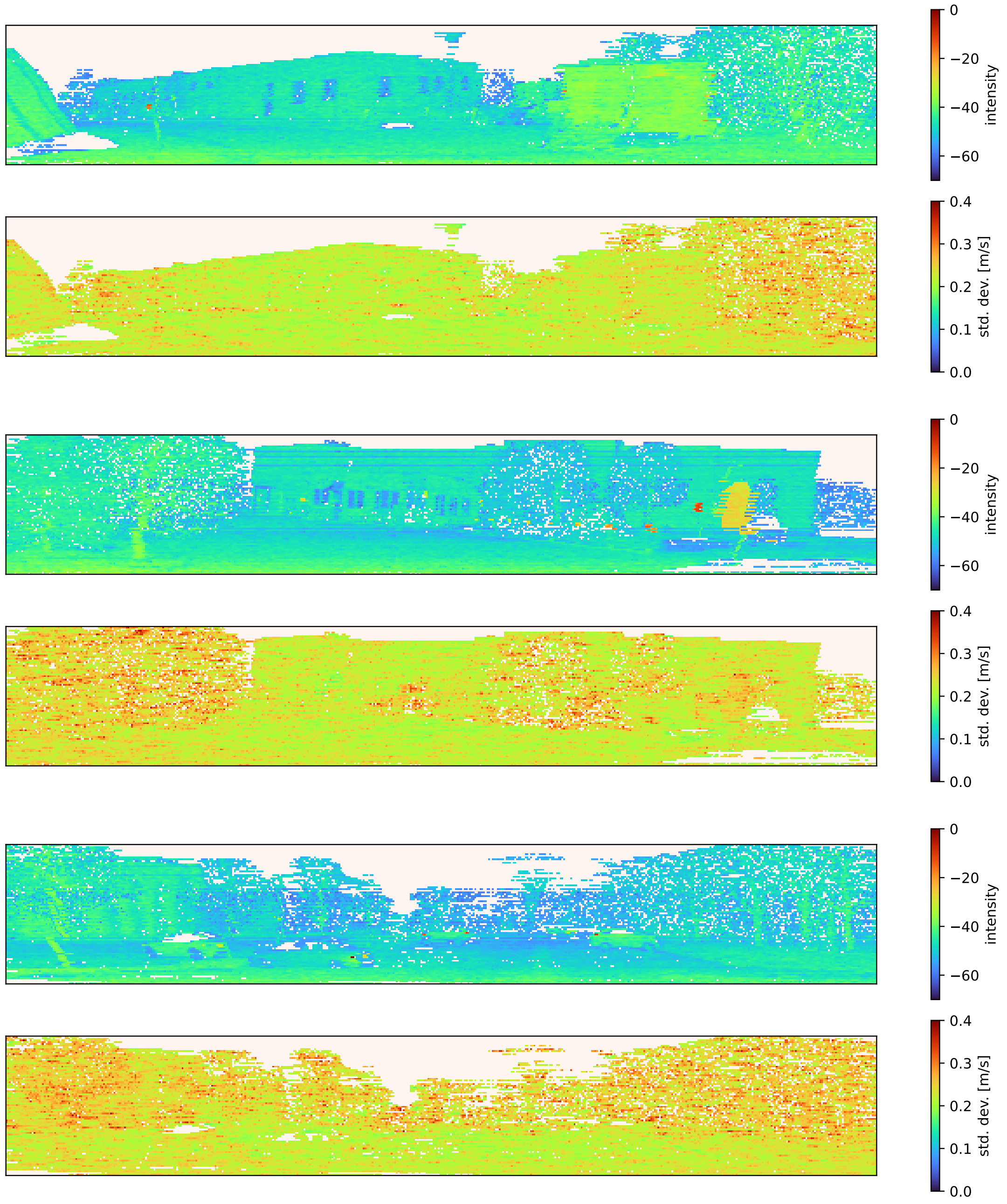


Figure B.1: Three qualitative examples that demonstrate the Doppler uncertainty prediction on the Boreas dataset. The intensity values are shown along with the prediction to provide visual context of the scene. Doppler error tends to be higher on unstructured surfaces such as the foliage on trees and lower on flat surfaces such as the ground.

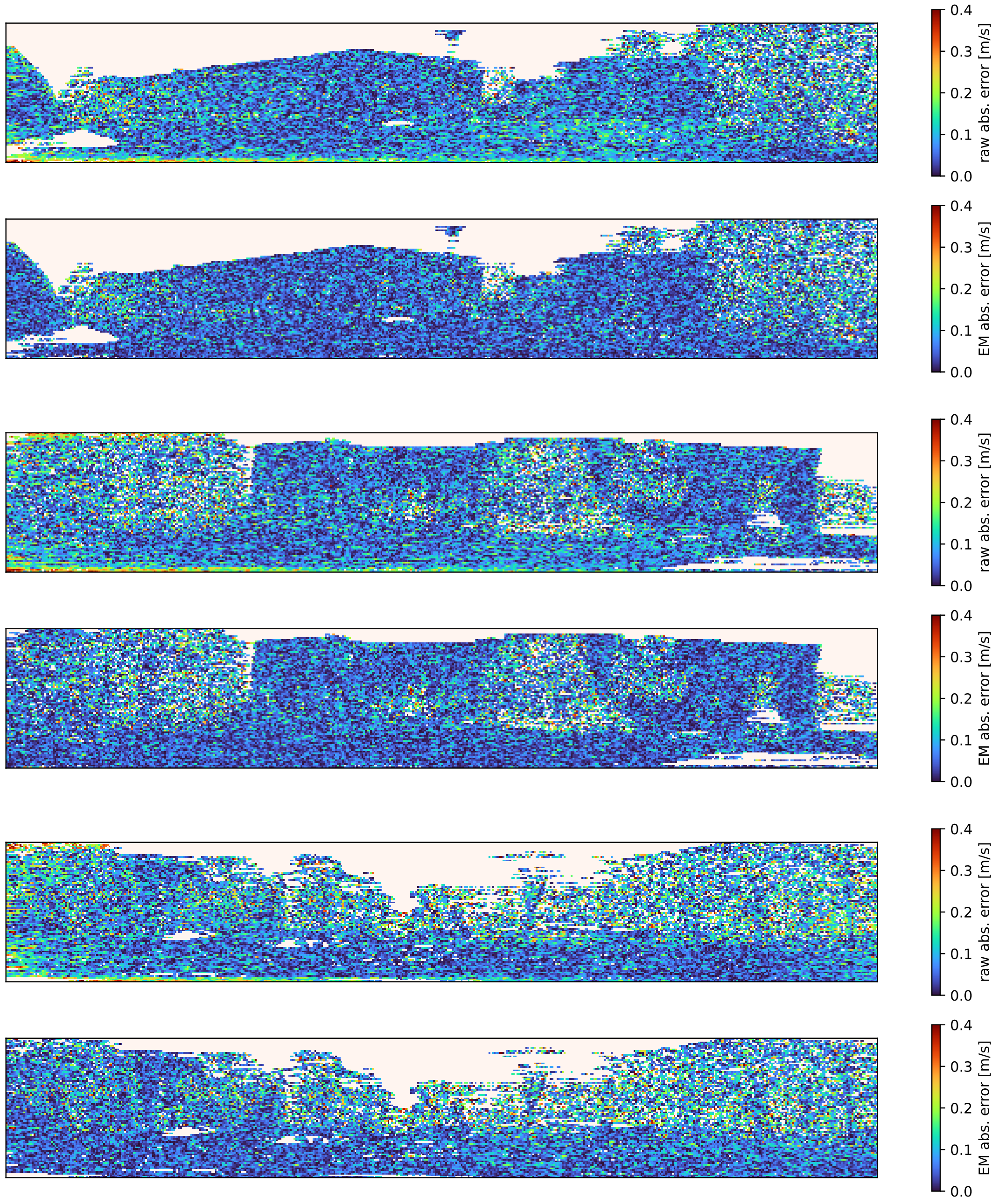


Figure B.2: Three qualitative examples that demonstrate the Doppler bias prediction on the Boreas FMCW dataset. The raw Doppler errors are shown along with the prediction to provide a visual comparison. Even without training with supervision from the groundtruth trajectory, we are able to adequately estimate the measurement biases.

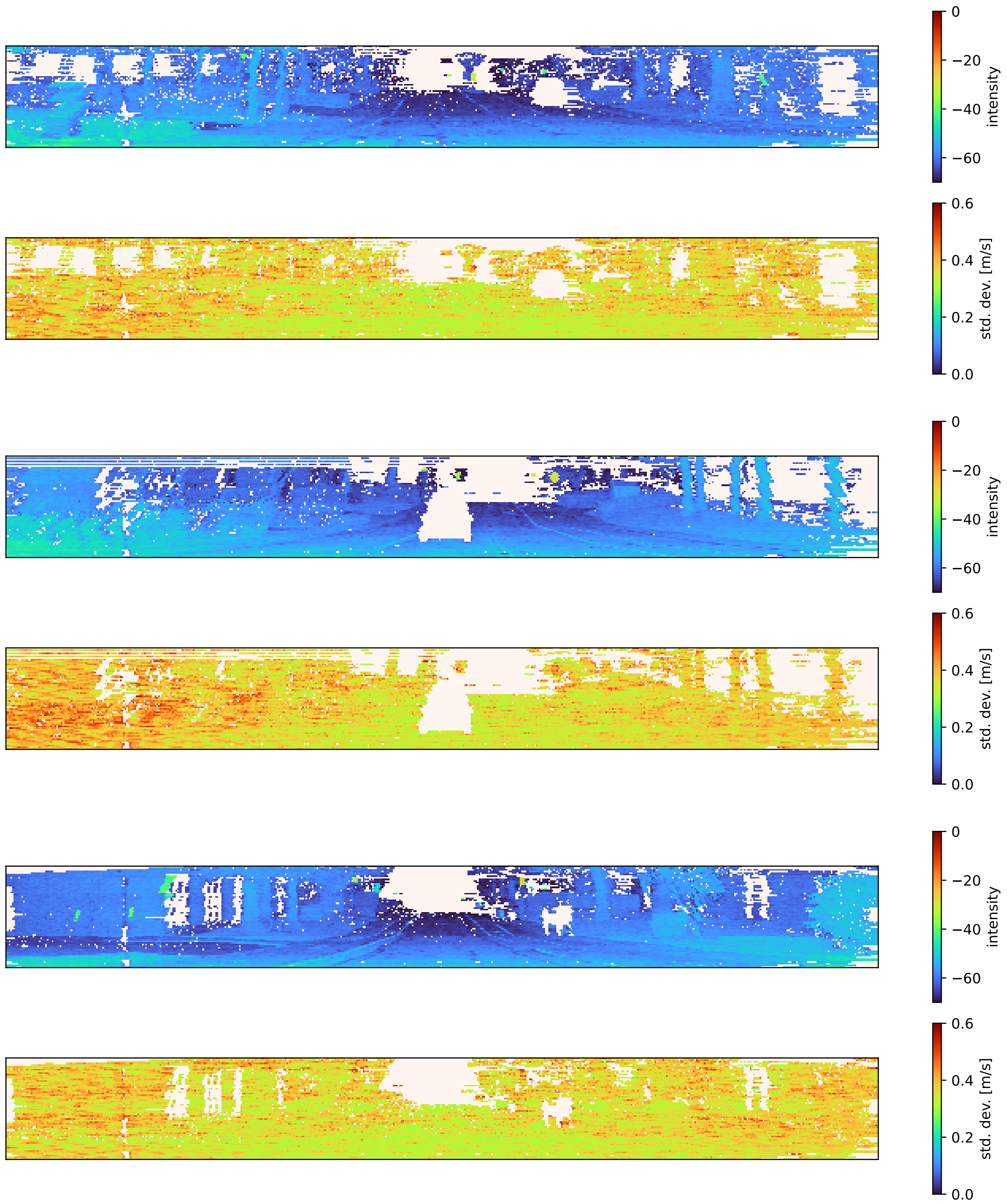


Figure B.3: Three qualitative examples that demonstrate the Doppler uncertainty prediction on the Aeva HQ dataset. The intensity values are shown along with the prediction to provide visual context of the scene. Similar to the Boreas FMCW dataset, Doppler error tends to be higher on unstructured surfaces such as the foliage on trees and lower on flat surfaces such as the ground.

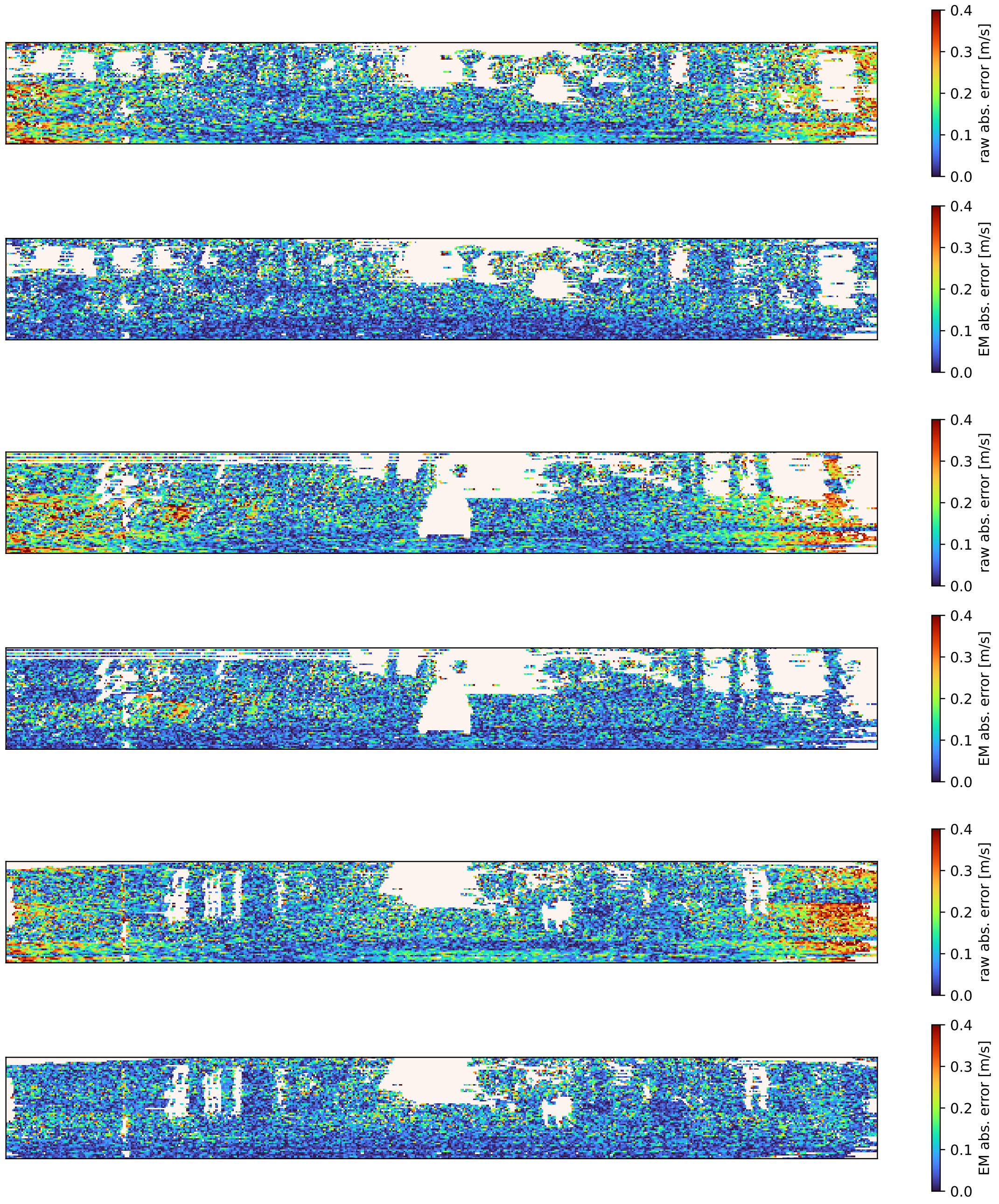


Figure B.4: Three qualitative examples that demonstrate the Doppler bias prediction on the Aeva HQ dataset. The raw Doppler errors are shown along with the prediction to provide a visual comparison. Even without training with supervision from the groundtruth trajectory, we are able to adequately estimate the measurement biases.

Bibliography

Agishev, R., Petříček, T., and Zimmermann, K. (2023). Self-supervised depth correction of lidar measurements from map consistency loss. *IEEE Robotics and Automation Letters*. 64

Ala-Luhtala, J., Särkkä, S., and Piché, R. (2015). Gaussian filtering and variational approximations for bayesian smoothing in continuous-discrete stochastic dynamic systems. *Signal Processing*, 111:124–136. 14, 18, 28, 33

Anderson, S. and Barfoot, T. D. (2013). Towards relative continuous-time slam. In *ICRA*, pages 1033–1040. 74, 100

Anderson, S. and Barfoot, T. D. (2015). Full STEAM ahead: Exactly sparse Gaussian process regression for batch continuous-time trajectory estimation on SE(3). In *IROS*. 55, 75, 98, 109

Arasaratnam, I. and Haykin, S. (2009). Cubature kalman filters. *IEEE Transactions on Automatic Control*, 54(6):1254–1269. 24, 25

Barfoot, T., Hay Tong, C., and Sarkka, S. (2014). Batch Continuous-Time Trajectory Estimation as Exactly Sparse Gaussian Process Regression. In *RSS*. 39, 71, 98, 100

Barfoot, T. D. (2024). *State Estimation for Robotics*. Cambridge University Press, 2nd edition. 1, 2, 7, 10, 14, 18, 31, 32, 36, 40, 64, 72, 117, 129, 131

Barfoot, T. D., Forbes, J. R., and Yoon, D. J. (2020). Exactly sparse gaussian variational inference with application to derivative-free batch nonlinear state estimation. *International Journal of Robotics Research (IJRR)*. 2, 3, 13, 45, 73, 115

Barnes, D., Gadd, M., Murcutt, P., Newman, P., and Posner, I. (2020). The oxford radar robotcar dataset: A radar extension to the oxford robotcar dataset. In *ICRA*. 116, 121, 122

Barnes, D. and Posner, I. (2020). Under the radar: Learning to predict robust keypoints for odometry estimation and metric localisation in radar. In *ICRA*. 111, 113

Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., and Gall, J. (2019). SemanticKITTI: A dataset for semantic scene understanding of lidar sequences. In *Int. Conf. on Comp. Vision*. 100

Behley, J. and Stachniss, C. (2018). Efficient surfel-based slam using 3d laser range data in urban environments. In *RSS*. 100, 118

Besl, P. and McKay, N. D. (1992). A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256. 100

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer. 2, 9, 13, 14, 28, 29, 33, 69

Bloesch, M., Czarnowski, J., Clark, R., Leutenegger, S., and Davison, A. J. (2018). CodeSLAM-learning a compact, optimisable representation for dense visual SLAM. In *CVPR*. 98

Bourmaud, G. (2016). Online variational bayesian motion averaging. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 126–142. 13

Brossard, M., Barrau, A., and Bonnabel, S. (2020a). AI-IMU dead-reckoning. *IEEE Transactions on Intelligent Vehicles*, 5(4):585–595. 48

Brossard, M., Bonnabel, S., and Barrau, A. (2020b). A new approach to 3D ICP covariance estimation. *RAL*. 101, 120

Broussolle, F. (1978). State estimation in power systems: detecting bad data through the sparse inverse matrix method. *IEEE Transactions on Power Apparatus and Systems*, (3):678–682. 14

Brown, D. C. (1958). A solution to the general problem of multiple station analytical stereotriangulation. RCA-MTP Data Reduction Technical Report No. 43 (or AFMTC TR 58-8), Patrick Airforce Base, Florida. 13

Buchanan, R., Agrawal, V., Camurri, M., Dellaert, F., and Fallon, M. (2022). Deep imu bias inference for robust visual-inertial odometry with factor graphs. *IEEE Robotics and Automation Letters*, 8(1):41–48. 64

Burnett, K., Yoon, D. J., Schoellig, A. P., and Barfoot, T. D. (2021). Radar odometry combining probabilistic estimation and unsupervised feature learning. In *Robotics: Science and Systems (RSS)*. 4, 97, 110, 111, 116

Burnett, K., Yoon, D. J., Wu, Y., Li, A. Z., Zhang, H., Lu, S., Qian, J., Tseng, W.-K., Lambert, A., Leung, K. Y., Schoellig, A. P., and Barfoot, T. D. (2023). Boreas: A multi-season autonomous driving dataset. *The International Journal of Robotics Research*, 42(1-2):33–42. 81, 82

Chen, K., Nemiroff, R., and Lopez, B. T. (2023). Direct lidar-inertial odometry: Lightweight lio with continuous-time motion correction. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3983–3989. 65

Chen, X., Milioto, A., Palazzolo, E., Giguère, P., Behley, J., and Stachniss, C. (2019). SuMa++: Efficient lidar-based semantic slam. In *IROS*. 101, 118

Cho, Y., Kim, G., and Kim, A. (2020). Unsupervised geometry-aware deep lidar odometry. In *ICRA*. 101, 116, 117, 118

Cools, R. (1997). Constructing cubature formulae: The science behind the art. *Acta Numerica*, 6:1–54. 24

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314. 96

Czarnowski, J., Laidlow, T., Clark, R., and Davison, A. J. (2020). Deepfactors: Real-time probabilistic dense monocular SLAM. *RAL*. 98

De Maio, A. and Lacroix, S. (2022). Deep bayesian ICP covariance estimation. In *International Conference on Robotics and Automation*. 48

Dellenbach, P., Deschaud, J.-E., Jacquet, B., and Goulette, F. G. (2022). Ct-icp: Real-time elastic lidar odometry with loop closure. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 5580–5586. 77, 100

DeTone, D., Malisiewicz, T., and Rabinovich, A. (2018). Self-improving visual odometry. *arXiv preprint arXiv:1812.03245*. 99

Dong, J., Mukadam, M., Dellaert, F., and Boots, B. (2016). Motion planning as probabilistic inference using gaussian processes and factor graphs. In *Robotics: Science and Systems*, volume 12, page 4. 13

Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localisation and mapping (SLAM): Part I the essential algorithms. *IEEE Robotics and Automation Magazine*, 11(3):99–110. 13

Erisman, A. M. and Tinney, W. F. (1975). On computing certain elements of the inverse of a sparse matrix. *Commununications of the ACM*, 18(3):177–179. 14, 23

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395. 2, 41, 75, 98

Forster, C., Carlone, L., Dellaert, F., and Scaramuzza, D. (2016). On-manifold preintegration for real-time visual–inertial odometry. *IEEE Transactions on Robotics*, 33(1):1–21. 66

Furgale, P., Barfoot, T. D., and Sibley, G. (2012). Continuous-time batch estimation using temporal basis functions. In *ICRA*. 100

García-Fernández, Á. F., Svensson, L., Morelande, M. R., and Särkkä, S. (2015). Posterior linearization filter: Principles and implementation using sigma points. *IEEE transactions on signal processing*, 63(20):5561–5573. 14, 28

Gašperin, M. and Juričić, D. (2011). Application of unscented transformation in nonlinear system identification. *IFAC Proceedings Volumes*, 44(1):4428–4433. 14, 15

Geiger, A., Lenz, P., and Urtasun, R. (2012). Are we ready for autonomous driving? The KITTI vision benchmark suite. In *CVPR*. 83, 100, 116, 118, 120, 121, 122

Ghahramani, Z. and Hinton, G. E. (1996). Parameter estimation for linear dynamical systems. Technical Report CRG-TR-96-2, University of Toronto. 97

Ghahramani, Z. and Roweis, S. T. (1999). Learning nonlinear dynamical systems using an em algorithm. In *Advances in neural information processing systems*, pages 431–437. 29, 97

Golub, H. and Van Loan, C. F. (1996). Matrix computations, johns hopkins uni. *Press, London*. 127

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`. 96

Goudar, A., Zhao, W., Barfoot, T. D., and Schoellig, A. P. (2022). Gaussian variational inference with covariance constraints applied to range-only localization. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 2872–2879. IEEE. 42

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *CVPR*, pages 770–778. 112

Hexsel, B., Vhavle, H., and Chen, Y. (2022). DICP: Doppler Iterative Closest Point Algorithm. In *RSS*. 65

Hidalgo-Carrió, J., Hennes, D., Schwendner, J., and Kirchner, F. (2017). Gaussian process estimation of odometry errors for localization and mapping. In *ICRA*, pages 5696–5701. IEEE. 64

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366. 96

Ito, K. and Xiong, K. (2000). Gaussian filters for nonlinear filtering problems. *IEEE Transactions on Automatic Control*, 45(5):910–927. 24, 25

Jatavallabhula, K. M., Iyer, G., and Paull, L. (2020). gradSLAM: Dense slam meets automatic differentiation. In *ICRA*. 98

Jensen, J. L. W. V. (1906). Sur les fonctions convexes et les inégalités entre les valeurs moyennes. *Acta mathematica*, 30:175–193. 26

Julier, S. and Uhlmann, J. (1996). A general method for approximating nonlinear transformations of probability distributions. Technical report, Robotics Research Group, University of Oxford. 23, 25

Kaess, M. and Dellaert, F. (2009). Covariance recovery from a square root information matrix for data association. *Robotics and Autonomous Systems*, 57(12):1198–1210. 14

Kellner, D., Barjenbruch, M., Klappstein, J., Dickmann, J., and Dietmayer, K. (2013). Instantaneous ego-motion estimation using doppler radar. In *International IEEE Conference on Intelligent Transportation Systems*, pages 869–874. 66, 75

Kellner, D., Barjenbruch, M., Klappstein, J., Dickmann, J., and Dietmayer, K. (2014). Instantaneous ego-motion estimation using multiple doppler radars. In *ICRA*, pages 1592–1597. 66

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. In *Int. Conf. on Learning Representations*. 81, 105, 117

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. In *Int. Conf. on Learning Representations*. 98

Ko, J. and Fox, D. (2009). Gp-bayesfilters: Bayesian filtering using gaussian process prediction and observation models. *Autonomous Robots*, 27(1):75–90. 99

Ko, J. and Fox, D. (2011). Learning gp-bayesfilters via gaussian process latent variable models. *Autonomous Robots*, 30(1):3–23. 99

Kokkala, J., Solin, A., and Särkkä, S. (2014). Expectation maximization based parameter estimation by sigma-point and particle smoothing. In *17th International Conference on Information Fusion (FUSION)*, pages 1–8. IEEE. 14, 15

Kokkala, J., Solin, A., and Särkkä, S. (2016). Sigma-point filtering and smoothing based parameter estimation in nonlinear dynamic systems. *Journal of Advances in Information Fusion*, 11(1):15–30. 14, 15, 24, 25, 99

Kramer, A., Stahoviak, C., Santamaria-Navarro, A., Agha-Mohammadi, A.-A., and Heckman, C. (2020). Radar-inertial ego-velocity estimation for visually degraded environments. In *ICRA*, pages 5739–5746. 66

Kullback, S. and Leibler, R. A. (1951). On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86. 14, 15

Kümmerle, J. and Kühner, T. (2020). Unified intrinsic and extrinsic camera and lidar calibration under uncertainties. In *ICRA*, pages 6028–6034. IEEE. 64

Laconte, J., Deschênes, S.-P., Labussière, M., and Pomerleau, F. (2019). Lidar measurement bias estimation via return waveform modelling in a context of 3d mapping. In *ICRA*, pages 8100–8106. IEEE. 64

Landry, D., Pomerleau, F., and Giguere, P. (2019). CELLO-3D: Estimating the covariance of ICP in the real world. In *ICRA*. 48, 101

Li, Q., Chen, S., Wang, C., Li, X., Wen, C., Cheng, M., and Li, J. (2019). LO-Net: Deep real-time lidar odometry. In *CVPR*. 100, 101, 116, 117, 118, 120

Li, R., Hwangbo, J., Chen, Y., and Di, K. (2011). Rigorous photogrammetric processing of HiRISE stereo imagery for mars topographic mapping. *IEEE Transactions on Geoscience and Remote Sensing*, 49(7):2558–2572. 13

Liu, K., Ok, K., Vega-Brown, W., and Roy, N. (2018). Deep inference for covariance estimation: Learning Gaussian noise models for state estimation. In *ICRA*. 48, 79, 112

Maddern, W., Pascoe, G., Linegar, C., and Newman, P. (2017). 1 Year, 1000km: The Oxford RobotCar Dataset. *International Journal of Robotics Research*. 116, 121, 122

Magnus, J. R. and Neudecker, H. (2019). *Matrix differential calculus with applications in statistics and econometrics*. John Wiley & Sons. 68

McGarey, P., MacTavish, K. A., Pomerleau, F., and Barfoot, T. D. (2017). Tslam: Tethered simultaneous localization and mapping for mobile robots. *International Journal of Robotics Research (IJRR)*, 36(12):1363–1386. 42

Mehra, R. (1970). On the identification of variances and adaptive kalman filtering. *IEEE Transactions on automatic control*, 15(2):175–184. 48

Meurant, G. (1992). A review on the inverse of symmetric tridiagonal and block tridiagonal matrices. *SIAM Journal of Matrix Analysis and Applications*, 13(3):707–728. 14

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. (2021). Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106. 128

Mukadam, M., Dong, J., Yan, X., Dellaert, F., and Boots, B. (2018). Continuous-time gaussian process motion planning via probabilistic inference. *The International Journal of Robotics Research*, 37(11):1319–1340. 13

Neal, R. M. and Hinton, G. E. (1998). A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer. 29

Nocedal, J. and Wright, S. J. (2006). *Numerical Optimization*. Springer, 2nd edition. 17

Opper, M. and Archambeau, C. (2009). The variational gaussian approximation revisited. *Neural computation*, 21(3):786–792. 14, 16

Palieri, M., Morrell, B., Thakur, A., Ebadi, K., Nash, J., Chatterjee, A., Kanellakis, C., Carlone, L., Guaragnella, C., and Agha-Mohammadi, A.-a. (2020). Locus: A multi-sensor lidar-centric solution for high-precision odometry and 3d mapping in real-time. *IEEE RAL*, pages 421–428. 65

Pan, Y., Xiao, P., He, Y., Shao, Z., and Li, Z. (2021). Mulls: Versatile lidar slam via multi-metric linear least square. In *ICRA*. 100

Park, Y. S., Shin, Y.-S., Kim, J., and Kim, A. (2021). 3d ego-motion estimation using low-cost mmwave radars via radar velocity factor for pose-graph slam. *IEEE RAL*, 6(4):7691–7698. 66

Paton, M., MacTavish, K., Warren, M., and Barfoot, T. D. (2016). Bridging the appearance gap: Multi-experience localization for long-term visual teach and repeat. In *IROS*, pages 1918–1925. 1

Peretroukhin, V. and Kelly, J. (2017). Dpc-net: Deep pose correction for visual localization. *IEEE Robotics and Automation Letters*, 3(3):2424–2431. 64

Peretroukhin, V., Vega-Brown, W., Roy, N., and Kelly, J. (2016). PROBE-GK: Predictive robust estimation using generalized kernels. In *International Conference on Robotics and Automation*, pages 817–824. 48, 49

Pineda, L., Fan, T., Monge, M., Venkataraman, S., Sodhi, P., Chen, R. T., Ortiz, J., DeTone, D., Wang, A., Anderson, S., et al. (2022). Theseus: A library for differentiable nonlinear optimization. *Advances in Neural Information Processing Systems*, 35:3801–3818. 98

Pomerleau, F., Colas, F., and Siegwart, R. (2015). *A Review of Point Cloud Registration Algorithms for Mobile Robotics*. 100

Pradeep, V., Konolige, K., and Berger, E. (2014). Calibrating a multi-arm multi-sensor robot: A bundle adjustment approach. In *Experimental robotics*, pages 211–225. Springer. 13

Qin, C., Ye, H., Pranata, C. E., Han, J., Zhang, S., and Liu, M. (2020). Lins: A lidar-inertial state estimator for robust and efficient navigation. *ICRA*, pages 8899–8905. 65

Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA. 14

Rauch, H. E., Tung, F., and Striebel, C. T. (1965). Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, 3(8):1445–1450. 13

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. In *Int. Conf. on Medical Image Computing and Comp.-Assisted Intervention*. 111, 112

Russell, R. L. and Reale, C. (2021). Multivariate uncertainty in deep learning. *IEEE Transactions on Neural Networks and Learning Systems*. 48

Särkkä, S. (2013). *Bayesian Filtering and Smoothing*. Cambridge University Press. 13, 24, 25, 28

Särkkä, S. (2013). *Bayesian filtering and smoothing*. Cambridge University Press. 114

Särkkä, S., Hartikainen, J., Svensson, L., and Sandblom, F. (2016). On the relation between gaussian process quadratures and sigma-point methods. *Journal of Advances in Information Fusion*, 11(1):31–46. 24, 25

Sarmavuori, J. and Särkkä, S. (2012). Fourier-hermite kalman filter. *IEEE Transactions on Automatic Control*, 57(6):1511–1515. 24

Schön, T. B., Wills, A., and Ninness, B. (2011a). System identification of nonlinear state-space models. *Automatica*, 47(1):39–49. 14, 15

Schön, T. B., Wills, A., and Ninness, B. (2011b). System identification of nonlinear state-space models. *Automatica*, 47(1):39–49. 99

Schwall, M., Daniel, T., Victor, T., Favaro, F., and Hohnhold, H. (2020). Waymo public road safety performance data. *arXiv preprint arXiv:2011.00038*. 1

Shan, T., Englot, B., Meyers, D., Wang, W., Ratti, C., and Rus, D. (2020). Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. In *IROS*, pages 5135–5142. 65

Sibley, G., Sukhatme, G., and Matthies, L. (2006). The iterated sigma point kalman filter with applications to long-range stereo. In *Proceedings of Robotics: Science and Systems*, Philadelphia, USA. 28, 32

Snelson, E. and Ghahramani, Z. (2006). Sparse gaussian processes using pseudo-inputs. In *Advances in neural information processing systems*, pages 1257–1264. 99

Stein, C. M. (1981). Estimation of the mean of a multivariate normal distribution. *Annals of Statististics*, 9(6):1135–1151. 17

Stengel, R. F. (1994). *Optimal control and estimation*. 48

Tagliabue, A., Tordesillas, J., Cai, X., Santamaria-Navarro, A., How, J. P., Carlone, L., and Agha-mohammadi, A.-a. (2021). Lion: Lidar-inertial observability-aware navigator for vision-denied environments. In *Experimental Robotics: The 17th International Symposium*, pages 380–390. Springer. 65

Takahashi, K., Fagan, J., and Chen, M.-S. (1973). A sparse bus impedance matrix and its application to short circuit study. In *Proceedings of the PICA Conference*. 14, 20, 21, 37, 127

Tang, C. and Tan, P. (2018). Ba-net: Dense bundle adjustment network. In *Int. Conf. on Learning Representations*. 98

Tang, T. Y., Yoon, D. J., Pomerleau, F., and Barfoot, T. D. (2018). Learning a bias correction for lidar-only motion estimation. In *Proceedings of the 15th Conference on Computer and Robot Vision (CRV)*, pages 166–173. IEEE. 64

Thomas, H., Goulette, F., Deschaud, J.-E., Marcotegui, B., and LeGall, Y. (2018). Semantic classification of 3d point clouds with multiscale spherical neighborhoods. In *Int. Conf. on 3D Vision*. 113, 115

Thomas, H., Qi, C. R., Deschaud, J.-E., Marcotegui, B., Goulette, F., and Guibas, L. J. (2019). KPConv: Flexible and deformable convolution for point clouds. *Int. Conf. on Comp. Vision*. 110, 111, 112, 128

Torroba, I., Sprague, C. I., Bore, N., and Folkesson, J. (2020). PointNetKL: Deep inference for GICP covariance estimation in bathymetric SLAM. *IEEE Robotics and Automation Letters*, 5(3):4078–4085. 48

Triggs, W., McLauchlan, P., Hartley, R., and Fitzgibbon, A. (2000). Bundle adjustment: A modern synthesis. In Triggs, W., Zisserman, A., and Szeliski, R., editors, *Vision Algorithms: Theory and Practice*, LNCS, pages 298–375. Springer Verlag. 14

Turner, R., Deisenroth, M., and Rasmussen, C. (2010). State-space inference and learning with gaussian processes. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 868–875. 99

Vega-Brown, W., Bachrach, A., Bry, A., Kelly, J., and Roy, N. (2013). CELLO: A fast algorithm for covariance estimation. In *International Conference on Robotics and Automation*, pages 3160–3167. 48

Vega-Brown, W. and Roy, N. (2013). CELLO-EM: Adaptive sensor models without ground truth. In *International Conference on Intelligent Robots and Systems*, pages 1907–1914. IEEE. 48

Vizzo, I., Guadagnino, T., Mersch, B., Wiesmann, L., Behley, J., and Stachniss, C. (2023). KISS-ICP: In Defense of Point-to-Point ICP – Simple, Accurate, and Robust Registration If Done the Right Way. *IEEE RAL*, 8(2):1029–1036. 100

von Stumberg, L., Wenzel, P., Khan, Q., and Cremers, D. (2020). Gn-net: The gauss-newton loss for multi-weather relocalization. *RAL*. 98

Wang, Y. and Solomon, J. M. (2019). Deep closest point: Learning representations for point cloud registration. In *Int. Conf. on Comp. Vision*, pages 3523–3532. 113

Wong, J. N., Yoon, D. J., Schoellig, A. P., and Barfoot, T. D. (2020a). A data-driven motion prior for continuous-time trajectory estimation on SE(3). *RAL*. 52, 56, 58, 100

Wong, J. N., Yoon, D. J., Schoellig, A. P., and Barfoot, T. D. (2020b). Variational inference with parameter learning applied to vehicle trajectory estimation. *IEEE Robotics and Automation Letters (RAL)*, 5(4):5291–5298. 3, 48, 52, 61

Wu, Y., Hu, D., Wu, M., and Hu, X. (2006). Gaussian filters for nonlinear filtering problems. *IEEE Transactions on Signal Processing*, 54(8):2910–2921. 24, 25

Wu, Y., Yoon, D. J., Burnett, K., Kammel, S., Chen, Y., Vhavle, H., and Barfoot, T. D. (2023). Picking up speed: Continuous-time lidar-only odometry using doppler velocity measurements. *IEEE Robotics and Automation Letters (RAL)*, 8(1):264–271. 4, 63, 65, 70, 75, 88, 95, 100, 119, 129

Xu, W., Cai, Y., He, D., Lin, J., and Zhang, F. (2022). Fast-lio2: Fast direct lidar-inertial odometry. *IEEE Transactions on Robotics*, 38(4):2053–2073. 65

Ye, H., Chen, Y., and Liu, M. (2019). Tightly coupled 3d lidar inertial odometry and mapping. In *ICRA*, pages 3144–3150. 65, 100

Yoon, D. J. and Barfoot, T. D. (2023). Towards consistent batch state estimation using a time-correlated measurement noise model. In *International Conference on Robotics and Automation (ICRA)*. 128

Yoon, D. J., Burnett, K., Laconte, J., Chen, Y., Vhavle, H., Kammel, S., Reuther, J., and Barfoot, T. D. (2023). Need for speed: Fast correspondence-free lidar-inertial odometry using doppler velocity. In *International Conference on Intelligent Robots and Systems (IROS)*. 4, 63, 65, 70, 88, 93, 95, 130

Yoon, D. J., Zhang, H., Gridseth, M., Thomas, H., and Barfoot, T. D. (2021). Unsupervised learning of lidar features for use in a probabilistic trajectory estimator. *IEEE Robotics and Automation Letters (RAL)*, 6(2):2130–2138. 4, 97, 110, 123

Zhang, J. and Singh, S. (2017). Low-drift and real-time lidar odometry and mapping. *Autonomous Robots*. 100, 118, 120

Zhang, Z. (1997). Parameter estimation techniques: A tutorial with application to conic fitting. *Image and vision Computing*, 15(1):59–76. 2, 98, 115

Zhao, S., Zhang, H., Wang, P., Nogueira, L., and Scherer, S. (2021). Super odometry: Imu-centric lidar-visual-inertial estimator for challenging environments. *IROS*, pages 8729–8736. 65