Towards Radar-Based Mapping and Localization in All Weather Conditions

by

Keenan Burnett

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy

Institute for Aerospace Studies
University of Toronto

Towards Radar-Based Mapping and Localization in
All Weather Conditions

Keenan Burnett
Doctor of Philosophy

Institute for Aerospace Studies
University of Toronto
2025

Abstract

Radar is an important sensor in autonomous driving due to its inherent robustness to dust, fog, rain, and snow. This thesis aims to demonstrate that radar can serve as a viable alternative to lidar for mapping and localization. To achieve this, we developed a data collection platform called Boreas, which includes a 128-beam lidar, a spinning mechanical radar, and a GNSS/IMU. Over one year, we gathered 350km of driving data under varying seasons and weather conditions. This dataset enabled the experiments conducted throughout this thesis and has been adopted by the wider research community. Our first contribution was to quantify the importance of motion distortion and Doppler effects on radar localization and to propose a lightweight method to compensate for them. Subsequently, we demonstrated a self-supervised radar odometry pipeline that combined a deep-learned front-end with a classical probabilistic back-end. We then developed a radar-based mapping and localization pipeline using the Visual Teach and Repeat paradigm. We conducted a thorough comparison of radar-to-radar, radar-to-lidar, and lidar-to-lidar localization showing that our radar-based localization is sufficiently accurate to enable autonomous navigation. The prospect of localizing radar to lidar maps is promising, given that many autonomous driving companies already utilize lidar maps. Additionally, using radar to localize on lidar maps can harness most of the advantages of radar. To further improve the performance of our radar odometry, we investigated fusing pointcloud measurements with an inertial measurement unit. This investigation led us to compare treating an IMU as an input to a motion model versus as a measurement of the state. We addressed this research question by performing an analysis on a 1D simulation, a 3D lidar-inertial simulation, and the Newer College Dataset. Our continuous-time lidar-inertial odometry, based on a Singer prior, achieved state-of-the-art results. This research informed the design of our radar-inertial odometry, based on a white-noise-on-acceleration prior, where we demonstrate results competitive with the state of the art on the Boreas dataset.

This thesis is dedicated to Sam and my family.
Thank you for all your support and encouragement throughout the years.

Acknowledgements

I want to express my utmost gratitude to Tim Barfoot for his mentorship and guidance throughout my time as a graduate student. Tim's hands-on feedback and leadership in picking research topics were invaluable. Tim also continually raised the bar to ensure that research was conducted at a high standard. I also want to thank Tim for allowing me to lead the University of Toronto AutoDrive team and subsequently build the Boreas autonomous driving research platform. I thank Angela Schoellig for her consistent support during my research. Angela supervised our first attendance (and win!) at the SAE AutoDrive Challenge and helped to convince me to pursue a PhD. Angela taught me how to promote my research and make the most out of conferences. General Motors donated the Buick, which formed the basis of our Boreas platform. The construction of the Boreas data collection platform would not have been possible without Goran Basic, who was contracted to design and assemble the roof rack. Goran went above and beyond to ensure the roof rack met our needs. The Amazon Open Data Sponsorship program supports this work by hosting the Boreas dataset. I would also like to thank Andrew Lambert and Keith Leung at Applanix for their help in integrating the Applanix POSLV system and providing post-processed GNSS data and tools to extract raw IMU data. I also want to thank my colleagues, David Yoon, Yuchen Wu, and Daniil Lisus, for collaborating on several publications. I thank my numerous other colleagues and collaborators for their comradery. I thank the Robotics Institute and Vector Institute for providing space to work and opportunities to network with other students and researchers. The Vector Institute also provided me with an annually recurring unrestricted grant. This work was partially financially supported by a Natural Sciences and Engineering Research Council (NSERC) grant and an Ontario Research Fund: Research Excellence grant.

Contents

List of Tables

List of Figures

Notation

$\mathbf{T}_{vi} \in SE(3)$ denotes a transformation matrix, where

$$\mathbf{T}_{vi} = \begin{bmatrix} \mathbf{C}_{vi} & \mathbf{r}_v^{iv} \\ \mathbf{0}^T & 1 \end{bmatrix}, \tag{1}$$

$\mathbf{C}_{vi} \in SO(3)$ is a rotation matrix, and $\mathbf{r}_v^{iv} \in \mathbb{R}^3$ is the translation vector from the vehicle frame v to the inertial frame i as measured in the vehicle frame. $\mathbf{T}_{vi}$ transforms points from the inertial frame to the vehicle frame. $\mathbf{u}^\wedge$ is the skew-symmetric operator,

$$\mathbf{u}^\wedge = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix}^\wedge = \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix}. \tag{2}$$

We overload this operator by re-using it on 6×1 vectors such that

$$\mathbf{x}^\wedge = \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix}^\wedge = \begin{bmatrix} \mathbf{v}^\wedge & \mathbf{u} \\ \mathbf{0}^T & 0 \end{bmatrix}. \tag{3}$$

$(\cdot)^\vee$ is then the inverse of this operator. A rotation matrix can be constructed using the exponential map with

$$\mathbf{C} = \exp(\boldsymbol{\phi}^\wedge) = \sum_{n=0}^{\infty} \frac{1}{n!} (\boldsymbol{\phi}^\wedge)^n. \tag{4}$$

Similarly, a transformation can also be constructed using an exponential map with

$$\mathbf{T} = \exp(\boldsymbol{\xi}^\wedge) = \sum_{n=0}^{\infty} \frac{1}{n!} (\boldsymbol{\xi}^\wedge)^n, \tag{5}$$

where

$$\boldsymbol{\xi} = \begin{bmatrix} \boldsymbol{\rho} \\ \boldsymbol{\phi} \end{bmatrix}, \tag{6}$$

$\boldsymbol{\rho}$ is a translational component, and $\boldsymbol{\phi}$ is a rotational component. The inverse of the exponential map is the logarithmic map, $\ln(\cdot)^\vee$. In this thesis, we make frequent use of body-centric velocity, $\boldsymbol{\varpi} = \boldsymbol{\varpi}_v^{vi}$, where

$$\boldsymbol{\varpi}_v^{vi} = \begin{bmatrix} \boldsymbol{\nu}_v^{vi} \\ \boldsymbol{\omega}_v^{vi}, \end{bmatrix}, \tag{7}$$

$\boldsymbol{\nu}_v^{vi} = \dot{\mathbf{r}}_v^{vi}$ is the body-centric linear velocity, and $\boldsymbol{\omega}_v^{vi}$ is the angular velocity that would be measured by a gyroscope in the vehicle frame. Poisson's equation is given by

$$\dot{\mathbf{C}}_{vi} = -(\boldsymbol{\omega}_v^{vi})^{\wedge}\mathbf{C}_{vi}. \tag{8}$$

The $(\cdot)^{\odot}$ operator is used to reverse the order of the operands when Jacobians are being derived,

$$\mathbf{x}^{\wedge}\mathbf{p} = \mathbf{p}^{\odot}\mathbf{x} \tag{9a}$$

$$\mathbf{p}^{\odot} = \begin{bmatrix}\boldsymbol{\rho}\\ \eta\end{bmatrix}^{\odot} = \begin{bmatrix}\eta\mathbf{1} & -\boldsymbol{\rho}^{\wedge}\\ \mathbf{0}^T & \mathbf{0}^T\end{bmatrix}, \tag{9b}$$

where $\mathbf{x} \in \mathbb{R}^6$, and $\mathbf{p}$ is a homogeneous point. $\mathrm{Ad}(\mathbf{T})$ represents the adjoint of a member of $SE(3)$,

$$\mathrm{Ad}(\mathbf{T}) = \boldsymbol{\mathcal{T}} = \begin{bmatrix}\mathbf{C} & (\mathbf{J}\boldsymbol{\rho})^{\wedge}\mathbf{C}\\ \mathbf{0} & \mathbf{C}\end{bmatrix}. \tag{10}$$

Another operator we use is

$$\mathbf{x}^{\curlywedge} = \begin{bmatrix}\mathbf{u}\\ \mathbf{v}\end{bmatrix}^{\curlywedge} = \begin{bmatrix}\mathbf{v}^{\wedge} & \mathbf{u}^{\wedge}\\ \mathbf{0} & \mathbf{v}^{\wedge}\end{bmatrix}. \tag{11}$$

For more details on notation, we refer the reader to [16].

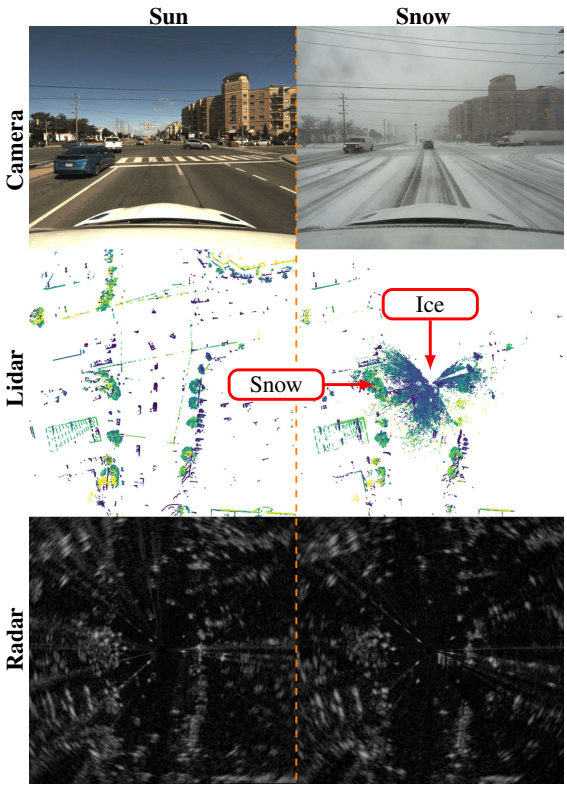
Chapter 1

Introduction

Autonomous vehicles have the potential to transform personal mobility. It is anticipated that with increased autonomous vehicle capabilities, annual fatalities due to traffic accidents may be reduced. Many traffic fatalities are the result of human error, whether that be driving under the influence of alcohol, speeding, distracted driving, or drowsy driving [111, 110, 109]. These are all cases where autonomous driving technology has the potential to intervene and minimize loss of life. At the same time, there has been a proliferation of radar in the context of robotics, with applications including autonomous vehicles [1], mining [4], and disaster response [113]. However, most autonomous vehicles today rely primarily on cameras and lidar for perception, and lidar has emerged as a dominant sensor for mapping and localization. Although cameras and lidars have been shown to achieve sufficient performance under nominal conditions, adverse weather remains an open problem. Radar sensors may provide a solution.

Thanks to its longer wavelength, radar is robust to small particles such as dust, fog, rain, and snow, which can negatively impact cameras and lidar. Figure 1.1 depicts our autonomous driving data collection platform, Boreas, alongside a qualitative comparison of camera, lidar, and radar data taken during a sunny day and a snowstorm. During the snowstorm, the lidar pointcloud becomes littered with detections associated with snowflakes and is partially blocked by a build-up of ice, while the radar data appears relatively unperturbed. Furthermore, radar tends to have a more extended detection range. Radar can also transmit through certain materials, allowing it to see beyond the line of sight of visual sensors like cameras and lidars. These features make radar particularly well-suited for inclement weather. However, radar typically has a coarser spatial resolution than lidar and produces noisier measurements. Radar also has a larger beam divergence than lidar. This large beam divergence and long wavelength mean that radar measurements notably depend on the sensed objects' macroscopic geometry and material properties. For these reasons, detecting and matching features between radar scans is challenging. In Table 1.1, we provide a high-level comparison of lidar, vision, and radar in the context of mapping and localization for autonomous vehicles. Figure 1.2 further illustrates some of the common sources of noise present in radar, which make it challenging to work with.

(a) Boreas data-collection platform



(b) Sun vs. snow in the Boreas dataset

Figure 1.1: (a) depicts our Boreas data-collection platform, which includes a Navtech radar, Velodyne 128-beam lidar, FLIR Blackfly S camera, and an Applanix POS LV GNSS/IMU. (b) provides a qualitative comparison of data collected on a sunny day and during a snowstorm.

Table 1.1: Comparison of mapping and localization sensors for autonomous vehicles.

	Lidar	Vision	Radar
Pros	✓ Accurate ✓ Dense pointcloud	✓ Works irrespective of environment geometry	✓ Inherently robust to weather ✓ Long Range ✓ Multiple returns per measurement
Cons	✗ Weather	✗ Brittle to large appearance changes (lighting) ✗ Weather	✗ Noisy ✗ Sparse

In this thesis, we demonstrate radar-based odometry and metric localization that approach lidar's performance and can operate even under extreme weather conditions where lidar is expected to fail. One of this thesis's main contributions is comparing radar-to-radar, radar-to-lidar, and lidar-to-lidar localization performance across varying seasonal and weather conditions. We show that our radar-to-lidar localization is sufficient to enable autonomous vehicle operation; this was not conclusively demonstrated in the prior literature. The remainder of this thesis centers around the various challenges and aspects of working with radar, such as motion distortion, Doppler effects,

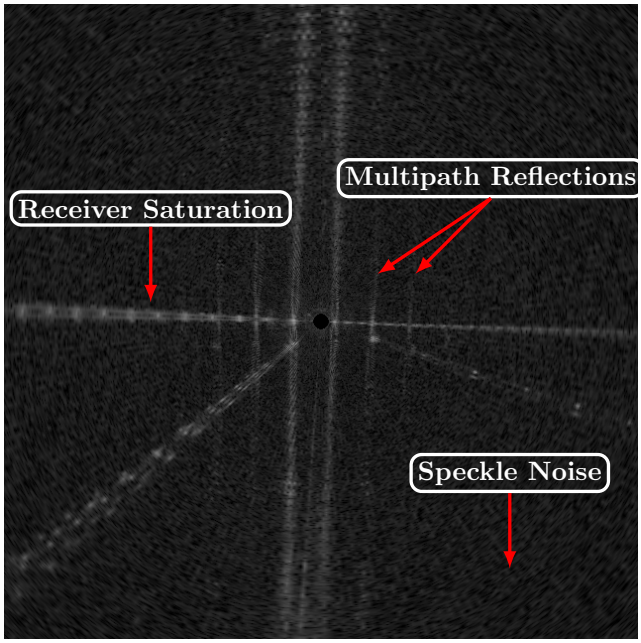


Figure 1.2: This figure depicts some of the common sources of noise present in radar. This data was collected while driving in a tunnel.

building a comprehensive dataset, and fusing inertial measurements to reduce the gap between radar and lidar-based odometry.

In our work, we present results using a mechanical spinning radar produced by Navtech as depicted in Figure 1.1. The Navtech radar's range and azimuth resolutions are competitive with other radar sensors. The main advantage to using the Navtech is that it provides a 360° field of view. This field of view could also be achieved by panelling several automotive radars around a robot. However, this would require calibrating the extrinsic transformations between multiple radars and ensuring accurate temporal synchronization. The Navtech radar also provides the raw power vs. range spectrum for each scanned azimuth; this is something that many automotive radar suppliers still do not offer. By having access to the raw data, we have the freedom to tune our feature extraction for each application. Another key difference between the Navtech radar and automotive radar is that Doppler velocity measurements are not currently supported by the Navtech radar. Nevertheless, the results presented in this thesis can also apply to a comparable combination of automotive radar sensors.

Figure 1.3 depicts a high-level timeline of publications in radar-based mapping and localization for robotics. This list is by no means exhaustive, and its purpose is merely to highlight some of the important works over the years that have influenced the developments in this thesis. We include a selection of our publications in this timeline to provide the reader with a sense of how the contributions of this thesis fit in with the broader work conducted in this field. There continues to be plenty of research in radar-based odometry, localization, and SLAM. However, we omit some of these recent works for the sake of brevity.

1998 Clark and Durrant-Whyte [46] First radar-based localization using retroreflectors

2001 Dissanayake et al. [51] First radar-based SLAM

2004 Jose and Adams [77] Proposed a new radar SLAM approach without retroreflectors

2006 Chandran and Newman [38] Radar motion estimation from map quality

2009 Rouveure et al. [128] First radar SLAM using correlative scan matching

2010 Checchin et al. [41] Radar SLAM using Fourier-Mellin transform

2011 Mullane et al. [102] Random finite set approach to radar SLAM

2011 Callmer et al. [35] First radar SLAM using visual feature matching

2013 Kellner et al. [78] First instantaneous ego-motion estimation using Doppler radar

2013 Vivet et al. [150] First Doppler-enabled spinning radar ego-motion estimation

2016 Schuster et al. [133] Radar SLAM using automotive radar on small scale

2018 Cen and Newman [36] Large-scale radar odometry using spinning radar

2020 Barnes et al. [20] Radar odometry using deep-learned correlative scan matching

2019 Holder et al. [69] Pose graph SLAM using automotive radar

2020 Barnes et al. [19] The Oxford Radar RobotCar dataset

2020 Barnes and Posner [18] Deep-learned feature matching radar odometry trained with poses

2020 Săftescu et al. [131] The first radar place recognition paper

2020 Tang et al. [146] First to demonstrate ground-based radar localization to satellite imagery

2020 Yin et al. [158] One of the first to demonstrate radar-to-lidar localization

2020 Hong et al. [70] Demonstrated radar SLAM in all weather conditions

2021 Ng et al. [106] B-spline continuous-time radar-inertial odometry using automotive radar

2021 Burnett et al. [28] (**Chapter 5**) Quantified motion distortion and Doppler effects

2021 Burnett et al. [29] (**Chapter 6**) Self-supervised learning-based radar odometry

2021 Adolfsson et al. [3] Demonstrated accurate radar odometry, rivalling lidar

2022 Burnett et al. [30] (**Chapter 7**) Compared radar to lidar localization across weather

2023 Burnett et al. [31] (**Chapter 4**) The Boreas autonomous driving dataset

2025 Burnett et al. [32] (**Chapter 9**) Continuous-time radar-inertial odometry using a Gaussian process prior

t

Figure 1.3: A brief history of radar for mapping and localization in robotics. This list is by no means exhaustive but rather highlights some of the relevant prior work that influenced this thesis and includes some of the publications produced in accordance with this thesis.

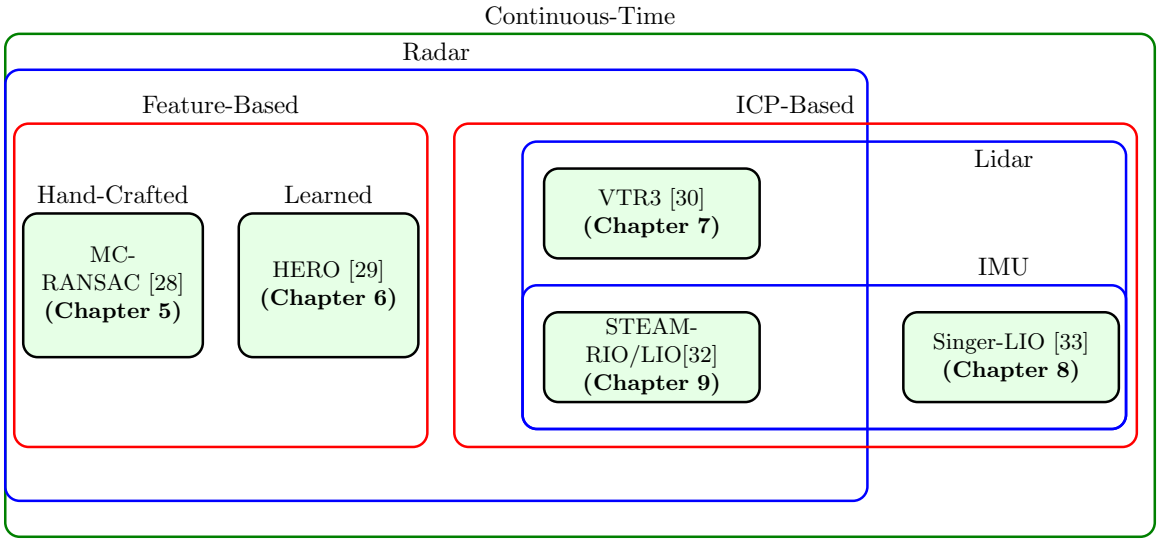


Figure 1.4: Conceptual map of the published works associated with this thesis.

1.1 Overview of Thesis

In Figure 1.4, we present a conceptual map of the published papers presented in this thesis, excluding our dataset paper. In general, this thesis centres around radar-based mapping and localization. The primary algorithmic difference between our published works is how we obtain correspondences between radar (or lidar) scans. For feature-based approaches, we first detect keypoints and then match descriptors. In **Chapter 5**, we used hand-crafted descriptors for this purpose, and in **Chapter 6** we use deep-learned features which were trained in a self-supervised fashion. Another way of obtaining correspondences is to perform pointcloud registration using iterative closest point (ICP), where an initial guess is required since this is an iterative solver. We use an ICP-like solver in **Chapter 7**, **Chapter 8**, and **Chapter 9**. Since we often have a good initial guess, thanks to motion priors or other sensors such as an IMU, these approaches based on ICP are quite robust. However, if one needs to localize without a prior initial guess, then the feature-based approaches will be of more use. This thesis also frequently presents results for lidar-based approaches to compare and contrast with radar. We also have two publications using inertial measurement units (IMUs). One unique aspect of this thesis is that all of the papers in this conceptual map involve some aspect of continuous-time state estimation. In most of the chapters, we provide radar odometry results. **Chapter 7** is the exception where we instead provide metric localization results. In Figure 1.5, we provide an example of a lidar map and radar map generated using the approach we present in **Chapter 9**. In the following, we provide a brief description of each chapter and the associated publications.

Chapter 2: Related Work We provide a review of relevant literature here.

Chapter 3: Continous-Time State Estimation using Gaussian Processes We provide a brief review of prior work in the area of continuous-time state estimation using Gaussian processes.

Chapter 4: Boreas: A Multi-Season Autonomous Driving Dataset

Associated publication: "Boreas: A multi-season autonomous driving dataset" K. Burnett et al. (The International Journal of Robotics Research (IJRR), 2023) [31]

Prior to this thesis, no single dataset could be used to compare radar localization to lidar localization across varying seasonal and weather conditions. The Boreas dataset was created to address this need. Our dataset includes over 350km of driving data collected over one year. I was the sole first author of a journal publication [31] based on this dataset and did most of the work to collect and process the dataset. The novel contributions of this dataset include:

- Data collected on a repeated route over one year, including multiple weather conditions such as snowstorms.
- A unique high-quality sensor configuration including a 128-beam lidar and a 360° radar.
- An online leaderboard for radar odometry, metric localization, and object detection.[1]
- A Python development kit for working with the dataset.[2]

Chapter 5: Motion Distortion and Doppler Effects in Spinning Radar Navigation
Associated publication: "Do we need to compensate for motion distortion and Doppler effects in spinning radar navigation?" K. Burnett, A. P. Schoellig, T. D. Barfoot (IEEE Robotics and Automation Letters (RA-L), 2020) (presented at the IEEE Conference on Robotics and Automation (ICRA), 2021) [28]

When working with the Navtech radar, two important distortion effects need to be considered: motion distortion, which results from the scanning-while-moving nature of the sensor, and Doppler distortion, which results from the ego-motion of the vehicle corrupting range measurements. As the second novel contribution of this thesis, we were the first to quantify the importance of these distortion effects in radar odometry and radar-based mapping and localization. We showed that motion distortion is important for radar odometry, but Doppler distortion can be neglected. However, it is important to compensate for both distortion effects for radar-based mapping and localization. I was the sole first author of the publication based on this work [28]. We make our code for this project publicly available[3].

Chapter 6: Combining Probabilistic Estimation and Self-Supervised Feature Learning
Associated publication: *"Radar odometry combining probabilistic estimation and unsupervised feature learning" K. Burnett, D. J. Yoon, A. P Schoellig, T. D. Barfoot (presented at Robotics: Science and Systems 2021) [29] *Equal contribution was shared with my co-author David Yoon where I focused on the front-end feature detection and David focused on the back-end state estimation.

It is challenging to detect and match keypoints between radar scans reliably. In [28], we experimented with matching ORB features [130] and compared this to a radial statistics descriptor [36]. We observed that bad feature matches have a negative effect on radar odometry performance. To address this, we investigated learning-based radar odometry [29]. As the third novel contribution of this thesis, we were the first to demonstrate self-supervised radar odometry combining a classic probabilistic back-end with a deep learning front-end. Self-supervised methods are important in the context of robotics because labeling datasets is both expensive and time-consuming. We can achieve the best of both worlds by combining deep learning with probabilistic state estimation.

[1] https://www.boreas.utias.utoronto.ca
[2] https://github.com/utiasASRL/pyboreas
[3] https://github.com/keenan-burnett/yeti_radar_odometry

Deep learning can be leveraged to process rich sensor data, while classical estimators can deal with probabilities and out-of-distribution samples through outlier rejection schemes. At the time of publication, our method was the top learning-based point-wise radar odometry method on the Oxford Radar Robotcar Dataset [19], outperforming a comparable supervised method, UnderTheRadar [18]. This work was published as a conference paper where I contributed equally with my co-author, David Yoon [29]. We make our code for this project publicly available[4].

Chapter 7: Are We Ready for Radar to Replace Lidar?

Associated publication: *"Are We Ready for Radar to Replace Lidar in All-Weather Mapping and Localization?" K. Burnett, Y. Wu, D. J. Yoon, A. P. Schoellig, T. D. Barfoot (IEEE Robotics and Automation Letters (RA-L), 2022) (presented at the International Conference on Intelligent Robots and Systems (IROS), 2022) *Equal contribution was shared with my co-author Yuchen Wu. I led research on this project. The implementation was a tight collaboration with my co-author.

Another strategy for working with radar data is to extract keypoints and register radar pointclouds using some variation of Iterative Closest Point (ICP). ICP relies on having a good initial guess as it is a local solver. This way, learning-based features can be more robust in localizing across a broader range of initial guesses. As the fourth novel contribution of this thesis, we were the first to provide a detailed comparison between radar and lidar localization across varying seasonal and weather conditions [30]. We implemented topometric localization using the visual tech and repeat framework. We compared lidar-to-lidar, radar-to-radar, and radar-to-lidar localization. Performing cross-modal localization between radar data and a lidar map allows us to take advantage of radar's robust sensing capabilities while using existing lidar maps that many autonomous driving companies already have. A surprising result of this work was that we observed that lidar-based localization can still outperform radar-based localization under moderate levels of precipitation. However, we also showed that our radar-based localization is sufficiently accurate to enable autonomous driving. Furthermore, we expect lidar-based localization to cease functioning under more extreme weather conditions, whereas radar should still function normally. This work was published as a journal paper where I shared equal contribution with my co-author, Yuchen Wu [30]. We make our code for this project publicly available[5].

Chapter 8: IMU as an Input versus a Measurement

Associated publication: "IMU as an Input versus a Measurement of the State in Inertial-Aided State Estimation" K. Burnett, A. P. Schoellig, T. D. Barfoot (Robotica, 2024) [33]

To address some of the shortcomings of radar and to reduce the gap between radar and lidar localization, we investigated combining radar with an inertial measurement unit (IMU). Radars and IMUs are high-rate sensors, and the problem of how best to fuse these measurements led us to consider why IMU measurements are often treated as an input to a motion model. The fifth contribution of this thesis is a comparison between treating an IMU as input to a motion model vs. a measurement of the state. The novel contributions are as follows:

- We provide a detailed comparison of treating an IMU as an input to a motion model vs. a measurement of the state on a 1D simulation problem. Such a comparison has not been

[4] `https://github.com/utiasASRL/hero_radar_odometry`
[5] `https://github.com/utiasASRL/vtr3`

previously presented in the literature.

- We show how to perform preintegration using heterogeneous factors (a combination of binary and unary factors) using continuous-time state estimation. To our knowledge, this has not been shown before in the literature.
- We present our novel approach to lidar-inertial odometry using a Singer prior, which includes body-centric acceleration in the state. We also provide experimental results in simulation and on the Newer College Dataset.

I was the sole first author of a journal paper based on this work [33]. We make our code for this project publicly available[6].

Chapter 9: Continuous-Time Radar-Inertial Odometry

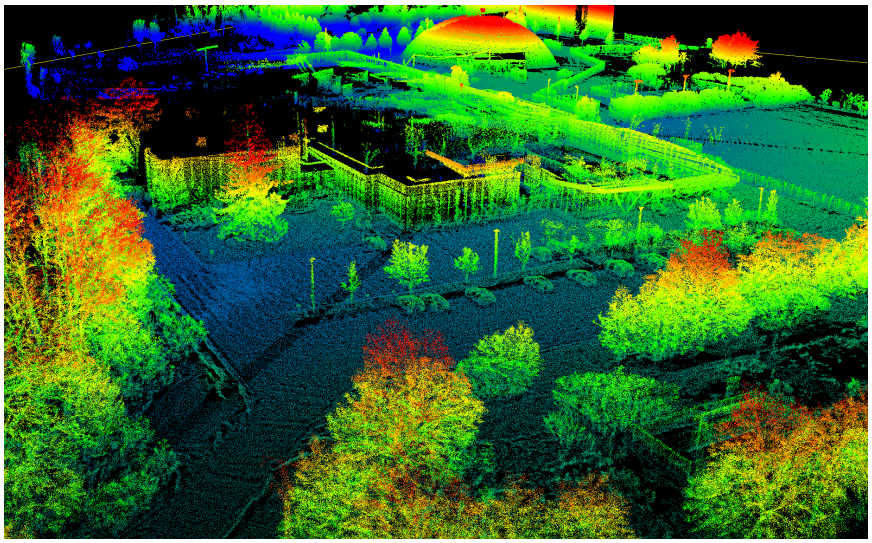
Associated publication: "Continuous-Time Radar-Inertial and Lidar-Inertial Odometry using a Gaussian Process Motion Prior" K. Burnett, A. P. Schoellig, T. D. Barfoot (IEEE Transactions on Robotics (T-RO), 2024) [32]

As the sixth contribution of this thesis, we demonstrate continuous-time radar-inertial and lidar-inertial odometry using a Gaussian process motion prior. The novel contributions are summarized as follows,

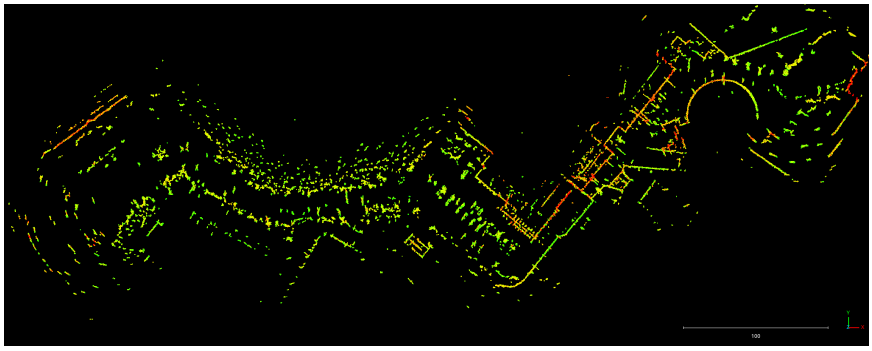
- We demonstrate continuous-time lidar-inertial and radar-inertial odometry using a Gaussian process motion prior where the preintegration cost is linear in the number of estimation times.
- We provide experimental results of our real-time approach on three datasets: KITTI-raw [61], Boreas [31] and the Newer College Dataset [123].
- We demonstrate radar-inertial odometry with a spinning mechanical radar using gyroscope and accelerometer measurements. To our knowledge, this has not been previously demonstrated in the literature.
- We provide a detailed comparison of lidar-inertial and radar-inertial odometry performance across varying seasonal and weather conditions.

I was the sole first author of a journal paper based on this work [32]. We make our code for this project publicly available[7].

[6] `https://github.com/utiasASRL/steam_icp`
[7] `https://github.com/utiasASRL/steam_icp`



(a) Lidar map of UTIAS (coloured by height)



(b) Radar map of UTIAS (coloured by intensity)

Figure 1.5: This figure depicts lidar and radar maps of the University of Toronto Institute for Aerospace Studies.

Chapter 2

Related Work

In this chapter, we review relevant work related to this thesis. The central theme of this thesis is radar-based mapping and localization. Another important theme is the application of continuous-time estimation techniques. We will review relevant work on overarching topics here while reserving some literature review for individual chapters where the discussion of the relevant work is isolated to that chapter.

2.1 Related Work

2.1.1 Lidar Odometry

Lidar odometry methods can be broadly classified into feature-based approaches, which seek to extract and match sparse geometric features, and direct methods, which work directly with raw lidar pointclouds. Direct methods usually rely on a variation of iterative closest point (ICP) to match pairs of pointclouds [121]. Due to the large number of points produced by modern lidar sensors (∼100k points per scan), these methods incur a heavy computational load. Recent methods rely on coarse voxelization and efficient data structures for map storage and retrieval [48, 151, 156, 44] to enable real-time operation. Care must also be taken to tune ICP parameters, such as the maximum point-to-point matching distance, to ensure reliable and fast convergence. There are also several ICP variants to choose from, such as point-to-point, point-to-plane, and Generalized ICP [121, 134]. Examples of feature-based methods include LOAM [161], which matches edge and plane features, and SuMa++ [45], which matches surfels using ICP aided by semantic segmentation labels from a neural network. Feature-based methods tend to work well in structured environments but may experience a drop in performance in unstructured environments. For a more detailed literature review on lidar odometry and lidar-inertial odometry, we refer readers to the recent survey by Lee et al. [89].

2.1.2 Lidar-Inertial Odometry

Prior works have leveraged inertial measurement units (IMUs) to address several shortcomings of lidar-only odometry. Firstly, IMU measurements can be used to compensate for the motion-distortion effect of lidar sensors [161]. Furthermore, IMUs can enable lidar-inertial odometry to

tackle trajectories with more aggressive motion and potentially degenerate geometry. Previous lidar-inertial odometry literature can then be sorted based on the degree of integration between the lidar and IMU modalities, ranging from loosely coupled to tightly coupled. Tightly coupled methods directly incorporate IMU data into the pointcloud alignment optimization.

LOAM [161] is an example of a loosely coupled approach where an IMU is used to undistort lidar pointclouds for use in ICP within a discrete-time state estimation framework where the IMU preintegration may be used as an initial guess for ICP. LIO-SAM [135] and LION [141] are examples of loosely coupled approaches that undistort lidar data using an IMU and then include both relative pose estimates from ICP and preintegrated IMU measurements in a factor graph. DLIO [44] is a recent example of a loosely coupled approach where an IMU is used to undistort lidar data. Preintegrated IMU measurements are then combined with pose estimates from ICP using a hierarchical geometric observer. LIOM [157] is an example of a tightly coupled lidar-inertial odometry using a factor graph. FAST-LIO2 [156] is another tightly coupled approach that uses an iterated extended Kalman filter.

Even after incorporating an IMU, some challenges remain, such as handling harsh environmental conditions such as dust, fog, rain, and snow that can adversely affect lidar data. Figure 1.1 depicts an example where lidar data is affected by snow and ice build-up. Radar is being investigated as a potential alternative to lidar in order to tackle these problems

2.1.3 Radar-Based Mapping and Localization

Radar mapping and localization is a long-standing research area in robotics. The first methods to demonstrate radar-based localization relied on reflective beacons installed in the environment [46, 51]. These reflective beacons were easy to discriminate from background noise however they severely limit the area in which radar-based localization could be deployed. Without such beacons, traditional radar filtering techniques such as Constant False Alarm Rate (CFAR) [125] have proven to be difficult to tune for radar-based localization. Setting the threshold too high results in insufficient features, which can cause localization to fail. Setting the threshold too low results in a noisy radar pointcloud and a registration process susceptible to local minima. Due to the noise inherent to radar, applying methods designed for lidar or vision is challenging.

Several methods have been proposed to deal with high noise in radar measurements. Jose and Adams [77] proposed to estimate the probability of target presence and to include radar cross-section in their simultaneous localization and mapping (SLAM) setup with limited success. Chandran and Newman [38] maximized an estimate of map quality to recover both the vehicle motion and radar map. These two prior works were limited to parking-lot-scale areas and the performance was partially limited by the quality of the radar sensors that they had access to at the time.

Rouveure et al. [128] and Checchin et al. [41] eschewed sparse feature extraction entirely by matching dense radar scans using 3D (x-y-yaw) cross-correlation and the Fourier-Mellin transform, respectively. Dense correlation remains a popular approach to radar odometry but less so for mapping and localization so far. Maps constructed using dense correlation will often require more storage space. In addition, it is challenging to account for motion distortion using this approach. Nevertheless, given the availability of high-performance parallel computing (GPUs) on many robotics platforms, this approach and its variants continue to hold promise. Recent publications in this area include the work by Park et al. [114] where they use the Fourier Mellin Transform on Cartesian and

log-polar radar images to estimate rotation and translation sequentially. Barnes et al. [20] demonstrated a fully differentiable, correlation-based radar odometry pipeline. Their approach learns a binary mask to remove distractor features before using brute force search to find the pose with the minimum cross-correlation.

Mullane et al. [102] proposed to use a random-finite-set formulation of SLAM in situations of high clutter and data association ambiguity. However, their approach incurs a heavy computational load for little added benefit. Vivet et al. [149] and Kellner et al. [78] proposed to use the relative Doppler velocity measurements to estimate the instantaneous egomotion. Indeed, Doppler velocity measurements can be used for odometry on their own, or can serve as useful motion priors in ICP-like estimators or even dense correlation. Callmer et al. [35] proposed to leverage features originally designed for vision to enable landmark-based SLAM. Schuster et al. [133] subsequently refined this approach by designing bespoke radar feature descriptors. Schuster et al. primarily worked with automotive radar where only a sparse set of targets is provided. Due to this data sparsity, it then becomes critical to obtain the correct correspondences between features using bespoke descriptors. Rapp et al. [124] used Normalized Distributions Transform (NDT) to perform probabilistic egomotion estimation with radar.

In their seminal work, Cen and Newman demonstrated accurate large-scale radar odometry using a spinning mechanical radar [36]. The accuracy and scale demonstrated in their work exceeded that which was previously demonstrated. Their work presented a new method to extract stable keypoints and perform scan matching using graph matching. Further research in this area has been spurred by the introduction of the Oxford Radar RobotCar Dataset [19], which includes lidar, vision, and radar data from a Navtech radar.

Odometry has recently been a central focus of radar-based navigation research. Components of an odometry pipeline can be repurposed for mapping and localization, which is the ultimate goal of this research. In [37], Cen et al. present an update to their radar odometry pipeline with improved keypoint detection, descriptors, and a new graph-matching strategy leading to improved odometry performance. Aldera et al. [5] train a focus of attention policy to downsample the measurements given to data association, thus speeding up the odometry pipeline. Aldera et al. [6] train a classifier on the principal eigenvector of their graph matching problem to predict and correct for failures in radar odometry. In [18], Barnes and Posner present a deep-learning-based keypoint detector and descriptor that are learned directly from radar data using differentiable point matching and pose estimation. In [70, 71], Hong et al. demonstrate a radar-SLAM pipeline capable of handling extreme weather. More recent work in radar-based localization has focused on improving aspects of radar odometry [85, 3, 7] and developing better SLAM pipelines [69].

The popularization of the Oxford dataset as well as MulRan [79] and our own Boreas dataset [31] have enabled more standardized comparisons between competing methods, enabling the community to better measure progress. For example, when Barnes and Posner published their work in 2020 [18], the state of the art in radar odometry was around 2% translational drift (KITTI metric) on the Oxford dataset. Since then, the state of the art is now around 0.5% translational drift on the Boreas dataset. There is not yet a single popular dataset for assessing radar odometry using automotive radar, so the results presented in the area are somewhat disjointed but it appears that progress is being made [164] [68]. The use of low-cost radar sensors has recently become a salient area of research for indoor positioning systems under conditions unfavourable to vision [82] [94] [99] [72].

Other research in radar-based localization focuses on topological localization (also known as place recognition), which can be used by downstream metric mapping and localization systems to identify loop closures. Săftescu et al. [131] learn a metric space embedding for radar scans using a convolutional neural network. Nearest-neighbour matching is then used to recognize locations at test time. Gadd et al. [58] improve this place recognition performance by integrating a rotationally invariant metric space embedding into a sequence-based trajectory matching system previously applied to vision [101]. De Martini et al. [47] proposed a two-stage system to integrate topological localization with metric pose estimation. A related avenue of research has been to localize radar scans using existing satellite imagery [146] [144] [145] or the pre-built lidar maps [158, 159].

Currently, the state of the art for radar odometry (without machine learning) with a spinning mechanical radar is CFEAR, which extracts only the k strongest detections on each scanned azimuth and subsequently matches the live radar scan to a sliding window of keyframes in a manner similar to ICP [4]. At the recent 2024 Radar in Robotics competition, the winning entry was CFEAR++ [91], which is built on top of CFEAR while also using semantic segmentation to focus on radar points associated with buildings as well as an IMU for a rotation prior.

Automotive radar sensors offer range and azimuth resolutions approximately on par with mechanically actuated radar. It is possible to replace a single 360-degree rotating radar with several automotive radars panelled around a vehicle [34]. Each target will then enjoy a relative Doppler velocity measurement, which can be used to estimate ego-motion [78]. However, recent work [85, 59] indicates that the target extraction algorithms built into automotive radar may not be optimal for mapping and localization. Thus, sensors that expose the underlying signal data offer greater flexibility since the feature extraction algorithm can be tuned for the desired application. Furthermore, many automotive radar sensors do not provide access to the raw data cube but instead provide access to a sparse set of targets resulting from feature extraction via Constant False Alarm Rate (CFAR) and Fast Fourier Transforms (FFTs). When the raw data is unavailable, we cannot tune the feature extraction process for our desired application (mapping and localization). Other single-chip radar sensors or cascaded radar sensors produced by Texas Instruments provide access to the raw sensor data. However, they tend to provide angular resolutions significantly worse than that provided by either the Navtech radar or automotive radar. From a researcher's perspective, the Navtech sensor is also convenient because it provides 360-degree coverage with a single sensor. It removes the need for engineering effort in extrinsic calibration and temporal synchronization of multiple sensors. Recent works have benefited from advancements in radar sensors where frequency-modulated continuous wave (FMCW) radar sensors now possess target elevation in 3D. For a more detailed review of radar-based localization, we refer readers to the survey by Harlow et al. [65].

2.1.4 Continuous-Time State Estimation

There are two main classes of continuous-time approaches: parametric approaches that rely on temporal basis functions and non-parametric approaches such as Gaussian processes. Two popular examples of parametric approaches are linear interpolation and cubic B-splines.

Linear interpolation is often performed in the Lie algebra between pairs of discrete trajectory samples. This approach assumes a constant velocity between pairs of poses and relies on sampling the trajectory at a sufficiently high rate to support dynamic motions. These approaches sometimes upsample the estimated trajectory using splines to remove motion distortion from pointclouds.

Smoothness factors may be included to penalize acceleration between pairs of poses. Examples of linear interpolation approaches include [23, 108, 162]. CT-ICP is an example of linear interpolation where their innovation was to parametrize each lidar scan as a pair of poses at the start and end of the scan and to model the motion during a scan with constant velocity while allowing trajectory discontinuities between scans [49].

Parametric approaches represent the trajectory using a finite set of temporal basis functions. Previous examples of parametric approaches include [57, 119]. Recent examples of parametric approaches applied to lidar odometry and lidar-inertial odometry include [53, 122, 112, 96, 86], all of which use B-splines.

Non-parametric approaches such as Gaussian processes (GPs) seek to model a continuous-time trajectory implicitly given a set of measurements of the state. The state at a set of estimation times can then be determined by performing Gaussian process regression. These estimation times may be chosen independently of the measurement times. The posterior GP may then be queried at any time of interest. In prior work, it was shown that for vector spaces, a linear time-varying stochastic differential equation can be interpreted as a Gaussian process, and batch continuous-time trajectory estimation can be performed efficiently thanks to the exact sparsity of the inverse kernel matrix owing to the Markovian nature of the state [11]. This approach enables linear time complexity instead of the usual cubic time complexity for Gaussian process regression. Furthermore, this work showed that posterior interpolation could be performed as an O(1) operation. Subsequently, Anderson and Barfoot extended this approach to work with $SE(3)$ where the trajectory is divided into a sequence of local GPs [10]. Recently, Le Gentil and Vidal-Calleja employed Gaussian processes to model the linear acceleration and angular velocities in continuous time given a set of IMU measurements [88, 87]. They then used their estimated GP to upsample IMU measurements towards undistorting pointclouds and to provide improved preintegration measurements for inertial-aided state estimation. One appealing aspect of our approach is that we start from a physically motivated prior: white noise on acceleration or constant velocity. Furthermore, compared to the linear interpolation approaches, the Gaussian process prior provides a principled manner to construct motion priors and perform interpolation. In addition, the hyper-parameters of the GP can be learned from a training set using maximum likelihood, enabling a data-driven approach to fine-tune the GP for each application. Determining the spacing of control points is an important engineering challenge in using B-splines, which can be avoided by using Gaussian processes instead. For a comparison of splines and Gaussian processes, we refer the reader to Johnson et al. [75]. We refer the reader to Talbot et al. [142] for an up-to-date literature review of continuous-time state estimation.

Chapter 3

Continous-Time State Estimation using Gaussian Processes

In this chapter, we review prior work that demonstrated continuous-time trajectory estimation as exactly sparse Gaussian process regression [10, 11, 16]. We first consider states that can be described using a vector space representation and then present prior work for handling Lie groups such as $SE(3)$. Here, we consider systems with a Gaussian process (GP) prior and a linear measurement model:

$$\mathbf{x}(t) \sim \mathcal{GP}(\check{\mathbf{x}}(t), \check{\mathbf{P}}(t,t')), \tag{3.1a}$$

$$\mathbf{y}_k = \mathbf{C}_k \mathbf{x}(t_k) + \mathbf{n}_k, \tag{3.1b}$$

where $\mathbf{x}(t)$ is the state, $\check{\mathbf{x}}(t)$ is the mean function, $\check{\mathbf{P}}(t,t')$ is the covariance function, and $\mathbf{y}_k$ are measurements corrupted by zero-mean Gaussian noise $\mathbf{n}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k)$. In this section, we restrict our attention to a class of GP priors resulting from linear time-invariant (LTI) stochastic differential equations (SDEs) of the form

$$\begin{aligned} \dot{\mathbf{x}}(t) &= \boldsymbol{A}\mathbf{x}(t) + \boldsymbol{B}\mathbf{u}(t) + \boldsymbol{L}\mathbf{w}(t), \\ \mathbf{w}(t) &\sim \mathcal{GP}(\mathbf{0}, \boldsymbol{Q}\delta(t-t')), \end{aligned} \tag{3.2}$$

where $\mathbf{w}(t)$ is a white-noise Gaussian process, $\boldsymbol{Q}$ is a power spectral density matrix, and $\mathbf{u}(t)$ is a known exogenous input. The general solution to this differential equation is

$$\mathbf{x}(t) = \boldsymbol{\Phi}(t,t_0)\mathbf{x}(t_0) + \int_{t_0}^{t} \boldsymbol{\Phi}(t,s)(\boldsymbol{B}\mathbf{u}(s) + \boldsymbol{L}\mathbf{w}(s))ds, \tag{3.3}$$

where $\boldsymbol{\Phi}(t,s) = \exp(\mathbf{A}(t-s))$ is the transition function. The mean function is

$$\check{\mathbf{x}}(t) = E[\mathbf{x}(t)] = \boldsymbol{\Phi}(t,t_0)\check{\mathbf{x}}_0 + \int_{t_0}^{t} \boldsymbol{\Phi}(t,s)\boldsymbol{B}\mathbf{u}(s)ds. \tag{3.4}$$

Over a sequence of estimation times, $t_0 < t_1 < \cdots < t_K$, the mean function can be written as

$$\check{\mathbf{x}}(t_k) = \boldsymbol{\Phi}(t_k, t_0)\check{\mathbf{x}}_0 + \sum_{n=1}^{k} \boldsymbol{\Phi}(t_k, t_n)\mathbf{B}_n\mathbf{u}_n, \tag{3.5}$$

assuming piecewise-constant input $\mathbf{u}_n$. This can be rewritten in a lifted form as

$$\check{\mathbf{x}} = \mathbf{A}\mathbf{B}\mathbf{u}, \tag{3.6}$$

where $\mathbf{A}$ is the lifted lower-triangular transition matrix, the inverse of which is

$$\mathbf{A}^{-1} = \begin{bmatrix} \mathbf{1} & & & \\ -\boldsymbol{\Phi}(t_1, t_0) & \ddots & & \\ & \ddots & \mathbf{1} & \\ & & -\boldsymbol{\Phi}(t_K, t_{K-1}) & \mathbf{1} \end{bmatrix}, \tag{3.7}$$

$\mathbf{B} = \text{diag}(\mathbf{1}, \boldsymbol{B}_1, \cdots, \boldsymbol{B}_K)$, and $\mathbf{u} = [\check{\mathbf{x}}_0^T \ \mathbf{u}_1^T \ \cdots \ \mathbf{u}_K^T]^T$. See [16] for further details on the formulations above. The covariance function is then

$$\begin{aligned} \check{\mathbf{P}}(t, t') &= E[(\mathbf{x}(t) - E[\mathbf{x}(t)])(\mathbf{x}(t') - E[\mathbf{x}(t')])^T] \\ &= \boldsymbol{\Phi}(t, t_0)\check{\mathbf{P}}_0\boldsymbol{\Phi}(t', t_0)^T + \int_{t_0}^{\min(t,t')} \boldsymbol{\Phi}(t, s)\boldsymbol{L}\boldsymbol{Q}\boldsymbol{L}^T\boldsymbol{\Phi}(t', s)^T ds. \end{aligned} \tag{3.8}$$

The covariance can also be rewritten in a lifted form using the same set of estimation times as above,

$$\check{\mathbf{P}} = \mathbf{A}\mathbf{Q}\mathbf{A}^T, \tag{3.9}$$

where $\mathbf{Q} = \text{diag}(\check{\mathbf{P}}_0, \mathbf{Q}_1, \cdots, \mathbf{Q}_K)$, and

$$\mathbf{Q}_k = \int_0^{\Delta t_k} \exp(\boldsymbol{A}(\Delta t_k - s))\boldsymbol{L}\boldsymbol{Q}\boldsymbol{L}^T \exp(\boldsymbol{A}(\Delta t_k - s))^T ds. \tag{3.10}$$

Our prior over the entire trajectory can then be written as

$$\mathbf{x} \sim \mathcal{N}(\check{\mathbf{x}}, \check{\mathbf{P}}), \tag{3.11}$$

where $\check{\mathbf{P}}$ is the kernel matrix. Note that the inverse kernel matrix $\check{\mathbf{P}}^{-1}$ is block-tridiagonal thanks to the Markovian nature of the state. This sparsity property also holds for linear time-varying (LTV) SDEs, provided that they are also Markovian [10]. The exact sparsity of $\check{\mathbf{P}}^{-1}$ is what allows us to perform efficient Gaussian process regression. This fact can be observed more easily by inspecting the following linear system of equations

$$\underbrace{\left(\check{\mathbf{P}}^{-1} + \mathbf{C}^T\mathbf{R}^{-1}\mathbf{C}\right)}_{\hat{\mathbf{P}}^{-1}} \hat{\mathbf{x}} = \mathbf{A}^{-T}\mathbf{Q}^{-1}\mathbf{B}\mathbf{u} + \mathbf{C}^T\mathbf{R}^{-1}\mathbf{y}, \tag{3.12}$$

where the Hessian is on the left-hand side, $\hat{\mathbf{P}}^{-1}$, is block-tridiagonal since $\check{\mathbf{P}}^{-1}$ is block-tridiagonal, and $\mathbf{C}^T\mathbf{R}^{-1}\mathbf{C}$ is block-diagonal. Thus, this linear system of equations can be solved in $O(K)$ time

using a sparse Cholesky solver. The exact sparsity of $\check{\mathbf{P}}^{-1}$ also enables us to perform efficient Gaussian process interpolation. The standard GP interpolation formulas are given by

$$\hat{\mathbf{x}}(\tau) = \check{\mathbf{x}}(\tau) + \check{\mathbf{P}}(\tau)\check{\mathbf{P}}^{-1}(\hat{\mathbf{x}} - \check{\mathbf{x}}), \tag{3.13a}$$

$$\hat{\mathbf{P}}(\tau,\tau) = \check{\mathbf{P}}(\tau,\tau) + \check{\mathbf{P}}(\tau)\check{\mathbf{P}}^{-1}\left(\hat{\mathbf{P}} - \check{\mathbf{P}}\right)\check{\mathbf{P}}^{-T}\check{\mathbf{P}}(\tau)^T, \tag{3.13b}$$

where

$$\check{\mathbf{P}}(\tau) = \begin{bmatrix}\check{\mathbf{P}}(\tau,t_0) & \check{\mathbf{P}}(\tau,t_1) & \cdots & \check{\mathbf{P}}(\tau,t_K)\end{bmatrix}. \tag{3.14}$$

The key to performing efficient interpolation relies on the sparsity of

$$\check{\mathbf{P}}(\tau)\check{\mathbf{P}}^{-1} = \begin{bmatrix}\mathbf{0} & \cdots & \mathbf{0} & \boldsymbol{\Lambda}(\tau) & \boldsymbol{\Psi}(\tau) & \mathbf{0} & \cdots & \mathbf{0}\end{bmatrix}, \tag{3.15}$$

where

$$\boldsymbol{\Psi}(\tau) = \mathbf{Q}_\tau \boldsymbol{\Phi}(t_{k+1},\tau)^T \mathbf{Q}_{k+1}^{-1}, \tag{3.16a}$$

$$\boldsymbol{\Lambda}(\tau) = \boldsymbol{\Phi}(\tau,t_k) - \boldsymbol{\Psi}(\tau)\boldsymbol{\Phi}(t_{k+1},t_k), \tag{3.16b}$$

are the only nonzero block-columns at indices $k+1$ and k, respectively. Thus, each interpolation query of the posterior trajectory is an $O(1)$ operation.

3.1 Continuous-Time Estimation on $SE(3)$

This section reviews continuous-time estimation on $SE(3)$ using the white-noise-on-acceleration prior as first described in [10]. We begin with the following nonlinear time-varying stochastic differential equation,

$$\dot{\mathbf{T}}(t) = \boldsymbol{\varpi}(t)^\wedge \mathbf{T}(t) \tag{3.17a}$$

$$\dot{\boldsymbol{\varpi}}(t) = \mathbf{w}'(t), \quad \mathbf{w}'(t) \sim \mathcal{GP}(\mathbf{0}, \boldsymbol{Q}'\delta(t-t')) \tag{3.17b}$$

where $\mathbf{T}(t) \in SE(3)$ is the pose, $\boldsymbol{\varpi}(t) = [\boldsymbol{\nu}^T \ \boldsymbol{\omega}^T]^T \in \mathbb{R}^6$ is the body-centric velocity consisting of a linear $\boldsymbol{\nu}(t)$ and angular $\boldsymbol{\omega}(t)$ component, and $\mathbf{w}'(t)$ is a white-noise Gaussian process where $\boldsymbol{Q}'$ is the symmetric positive-definite power-spectral density matrix. We refer to this as white-noise-on-acceleration due to white noise being injected on the body-centric acceleration $\dot{\boldsymbol{\varpi}}(t)$. The above nonlinear time-varying stochastic differential equation is then approximated using a sequence of local linear time-invariant stochastic differential equations [10]. Between pairs of estimation times, t_k and $t_{k+1}, k = 0 \ldots K-1$, local pose variables are defined in the Lie algebra $\boldsymbol{\xi}_k(t) \in \mathfrak{se}(3)$ such that

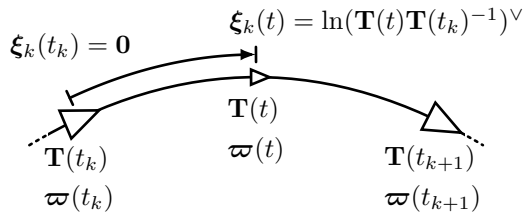


Figure 3.1: This figure illustrates the definition of the local variable $\boldsymbol{\xi}_k(t)$ which is in the tangent space of the pose at time t_k. The larger triangles denote the state at estimation times while the smaller triangle in the middle denotes the interpolated state at time t.

$$\mathbf{T}(t) = \exp(\boldsymbol{\xi}_k(t)^\wedge)\mathbf{T}(t_k). \tag{3.18}$$

The local kinematic equations are then defined as

$$\ddot{\boldsymbol{\xi}}_k(t) = \mathbf{w}_k(t), \quad \mathbf{w}_k(t) \sim \mathcal{GP}(\mathbf{0}, \boldsymbol{Q}\delta(t - t')). \tag{3.19}$$

This approximation of (3.17) holds so long as the process noise is small and the rotational motion between pairs of estimation times is also small. The local Markovian state variables are defined as

$$\boldsymbol{\gamma}_k(t) = \begin{bmatrix} \boldsymbol{\xi}_k(t) \\ \dot{\boldsymbol{\xi}}_k(t) \end{bmatrix}. \tag{3.20}$$

The local LTI SDE defined by (3.19), (3.20) is then stochastically integrated to arrive at the following local GP:

$$\boldsymbol{\gamma}_k(t) \sim \mathcal{GP}(\boldsymbol{\Phi}(t, t_k)\check{\boldsymbol{\gamma}}_k(t_k)), \boldsymbol{\Phi}(t, t_k)\check{\mathbf{P}}(t_k)\boldsymbol{\Phi}(t, t_k)^T + \mathbf{Q}_k), \tag{3.21}$$

where

$$\boldsymbol{\Phi}(t, t_k) = \begin{bmatrix} \mathbf{1} & (t - t_k)\mathbf{1} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \tag{3.22}$$

is the transition function,

$$\mathbf{Q}_k = \begin{bmatrix} \frac{1}{3}(t - t_k)^3\boldsymbol{Q} & \frac{1}{2}(t - t_k)^2\boldsymbol{Q} \\ \frac{1}{2}(t - t_k)^2\boldsymbol{Q} & (t - t_k)\boldsymbol{Q} \end{bmatrix} \tag{3.23}$$

is the covariance between two times, t, t_k, and $\check{\boldsymbol{\gamma}}_k(t_k)$, $\check{\mathbf{P}}(t_k)$ are the initial mean and covariance at $t = t_k$, the starting point of the local variable. Over a sequence of estimation times, $t_0 < t_1 < \cdots < t_K$, the kernel matrix can be written as

$$\check{\mathbf{P}} = \text{cov}(\delta\mathbf{x}) = \mathbf{A}\mathbf{Q}\mathbf{A}^T, \tag{3.24}$$

where, as before, $\mathbf{A}$ is the lifted transition matrix and $\mathbf{Q} = \text{diag}(\check{\mathbf{P}}_0, \mathbf{Q}_1, \cdots, \mathbf{Q}_K)$. Again, even though the kernel matrix is dense, the inverse kernel matrix $\check{\mathbf{P}}^{-1} = \mathbf{A}^{-T}\mathbf{Q}^{-1}\mathbf{A}^{-1}$ is block-tridiagonal. The exact sparsity of the inverse kernel matrix is what allows us to perform batch trajectory estimation as exactly sparse Gaussian process regression. As a result, the computation for batch trajectory estimation scales linearly with the number of estimation times.

To convert our continuous-time formulation into a factor graph, we construct a sequence of

motion prior factors between pairs of estimation times,

$$J_{v,k} = \frac{1}{2}\mathbf{e}_{v,k}^T \mathbf{Q}_k^{-1} \mathbf{e}_{v,k}, \tag{3.25a}$$

$$\mathbf{e}_{v,k} = \boldsymbol{\gamma}_k(t_{k+1}) - \check{\boldsymbol{\gamma}}_k(t_{k+1}) - \boldsymbol{\Phi}(t_{k+1}, t_k)(\boldsymbol{\gamma}_k(t_k) - \check{\boldsymbol{\gamma}}_k(t_k)), \tag{3.25b}$$

where J denotes a cost factor, $\mathbf{e}$ denotes an error function, and $\check{\boldsymbol{\gamma}}_k(t) = E[\boldsymbol{\gamma}_k(t)]$ is the prior mean. In the absence of exogenous control inputs, $\check{\boldsymbol{\gamma}}_k(t) = \boldsymbol{\Phi}(t, t_k)\check{\boldsymbol{\gamma}}_k(t_k)$ and so (3.25b) simplifies to

$$\mathbf{e}_{v,k} = \boldsymbol{\gamma}_k(t_{k+1}) - \boldsymbol{\Phi}(t_{k+1}, t_k)\boldsymbol{\gamma}_k(t_k). \tag{3.26}$$

To translate this prior factor, which is defined in terms of the local variables, into the global variables, we first rearrange (3.18) as

$$\boldsymbol{\xi}_k(t) = \ln(\mathbf{T}(t)\mathbf{T}(t_k)^{-1})^\vee. \tag{3.27}$$

We then use the following conversion for body-centric velocity:

$$\dot{\boldsymbol{\xi}}_k(t) = \boldsymbol{\mathcal{J}}(\boldsymbol{\xi}_k(t))^{-1}\boldsymbol{\varpi}(t). \tag{3.28}$$

From (3.27), (3.28), we can then define the local Markovian variable in terms of the global variables with

$$\boldsymbol{\gamma}(t) = \begin{bmatrix} \ln(\mathbf{T}(t)\mathbf{T}(t_k)^{-1})^\vee \\ \boldsymbol{\mathcal{J}}\left(\ln(\mathbf{T}(t)\mathbf{T}(t_k)^{-1})^\vee\right)^{-1}\boldsymbol{\varpi}(t) \end{bmatrix}. \tag{3.29}$$

The motion prior factors can then be written in terms of the global variables,

$$\mathbf{e}_{v,k} = \begin{bmatrix} \ln(\mathbf{T}_{k+1}\mathbf{T}_k^{-1})^\vee - (t_{k+1} - t_k)\boldsymbol{\varpi}_k \\ \boldsymbol{\mathcal{J}}\left(\ln(\mathbf{T}_{k+1}\mathbf{T}_k^{-1})^\vee\right)^{-1}\boldsymbol{\varpi}_{k+1} - \boldsymbol{\varpi}_k \end{bmatrix}, \tag{3.30}$$

where we observe that the motion prior is penalizing the state estimates from deviating from a constant velocity.

After performing batch trajectory estimation using these motion prior factors, the sparsity of the prior allows Gaussian process interpolation to be performed as an $O(1)$ operation where

$$\begin{aligned} \hat{\mathbf{T}}(\tau) = &\ \exp\left((\boldsymbol{\Lambda}_1(\tau)\hat{\boldsymbol{\gamma}}_k(t_k) + \boldsymbol{\Psi}_1(\tau)\hat{\boldsymbol{\gamma}}_k(t_{k+1}))^\wedge\right)\hat{\mathbf{T}}_k, \\ \hat{\boldsymbol{\varpi}}(\tau) = &\ \boldsymbol{\mathcal{J}}(\ln(\hat{\mathbf{T}}(\tau)\hat{\mathbf{T}}_k^{-1})^\vee)(\boldsymbol{\Lambda}_2(\tau)\hat{\boldsymbol{\gamma}}_k(t_k) + \boldsymbol{\Psi}_2(\tau)\hat{\boldsymbol{\gamma}}_k(t_{k+1})) \end{aligned} \tag{3.31}$$

are the interpolation equations involving only the two states bracketing the desired interpolation time: $t_k < \tau < t_{k+1}$. $\boldsymbol{\Psi}(\tau) = [\boldsymbol{\Psi}_1(\tau)^T\ \boldsymbol{\Psi}_2(\tau)^T]^T$ and $\boldsymbol{\Lambda}(\tau) = [\boldsymbol{\Lambda}_1(\tau)^T\ \boldsymbol{\Lambda}_2(\tau)^T]^T$ are the interpolation matrices that result from the standard GP interpolation formula in (3.16). When performing continuous-time trajectory estimation, we use the posterior interpolation formulas to build measurement factors at times between our desired estimation times.

Chapter 4

Boreas: A Multi-Season Autonomous Driving Dataset

Autonomous vehicle research and development to date has focused on achieving sufficient reliability in ideal conditions such as the sunny climates observed in San Francisco, California, or Phoenix, Arizona. Adverse weather conditions, such as rain and snow, remain outside the operational envelope for many of these systems. Additionally, most self-driving vehicles rely on highly accurate maps for localization and perception. These maps are costly to maintain and may degrade due to seasonal changes. These shortcomings must be addressed so that self-driving vehicles can be deployed safely.

To encourage research in this area, we have created the Boreas dataset, a large multi-modal dataset collected by driving a repeated route over one year. The dataset features over 350km of driving data with stark seasonal variations and multiple sequences with adverse weather, such as rain and falling snow. Our data-taking platform, shown in Figure 1.1, includes a 128-beam lidar, a 5 MP camera, and a 360° scanning radar. Globally consistent centimetre-accurate ground-truth poses are obtained by post-processing global navigation satellite system (GNSS), inertial measurement unit (IMU), wheel encoder data, and a secondary correction subscription. Our dataset supports benchmarks for odometry, metric localization, and 3D object detection.

This dataset may be used to study the effects of seasonal variation on long-term localization. It also enables comparisons of vision, lidar, and radar-based mapping and localization pipelines. Comparisons may include the robustness of individual sensing modalities to adverse weather or the resistance to map degradation.

Table 4.1: Related datasets. Lead: public leaderboard. Size: For perception datasets, size is given as the number of annotated frames and the number of annotations (3D boxes). GT: ground truth pose source. (A): automotive radar. (N): 360° Navtech radar. RTK (Real-Time Kinematic) uses a global positioning system (GPS) base station and differential measurements to improve GPS accuracy. RTX uses data from a global network of tracking stations to calculate corrections. This can be used to achieve cm-level accuracy without a base station [12]. †Waymo's Mid-Range, Short-Range proprietary 3D lidar. ‡The Oxford Robotcar dataset contains one sequence with snow on the ground but that sequence has no falling snow.

Name	Lead	Size	Camera	Lidar	Radar	GT	Night	Rain	Snow	Seasons
Perception										
ApolloScape [73]	✓	144k 70k boxes	2x9.2MP	1x64C	✗	GPS/IMU	✓	✓	✗	✗
Argoverse [39]	✓	22k 993k boxes	7x2.3MP +2x5MP	2x32C	✗	GPS/IMU	✓	✗	✗	✗
CADC [120]	✗	7.5k 372k boxes	8x1.3MP	1x32C	✗	GPS/IMU + RTK	✗	✗	✓	✗
KITTI (Object) [61]	✓	15k 200k boxes	4x1.4MP	1x64C	✗	GPS/IMU + RTK	✗	✗	✗	✗
nuScenes [34]	✓	40k 1.4M boxes	6x1.4MP	1x32C	✓(A)	GPS/IMU + Lidar Loc	✓	✓	✗	✗
RADIATE [136]	✗	44k 200k boxes	2x0.25MP	1x32C	✓(N)	GPS/IMU	✓	✓	✓	✗
Waymo OD [140]	✓	230k 12M boxes	5x2.5MP	1(MR†) 4(SR†)	✗	GPS/IMU	✓	✓	✗	✗
Boreas-Objects-V1	✓	7.1k 320k boxes	1x5MP	1x128C	✓(N)	GPS/IMU	✗	✗	✗	✗
Localization										
KITTI (Odometry) [61]	✓	39km 22 seqs	4x1.4MP	1x64C	✗	GPS/IMU + RTK	✗	✗	✗	✗
Complex Urban [74]	✗	451km 40 seqs	2x1.9MP	2x16C + 2x1C	✗	SLAM	✗	✗	✗	✗
Oxford RobotCar [97]	✗	1000km 100 seqs	3x1.2MP +3x1MP	1x4C + 2x1C	✗	GPS/IMU + RTK	✓	✓	✗‡	✓
Oxford Radar [19]	✗	280km 32 seqs	3x1.2MP +3x1MP	2x32C + 2x1C	✓(N)	GPS/IMU + VO	✗	✓	✗	✗
MulRan [79]	✗	124km 12 seqs	✗	1x64C	✓(N)	SLAM	✗	✗	✗	✗
Boreas	✓	350km 44 seqs	1x5MP	1x128C	✓(N)	GPS/IMU + RTX	✓	✓	✓	✓

4.1 Related Work

Many published autonomous driving datasets focus on perception, particularly 3D object detection and semantic segmentation of images and lidar pointclouds. However, these datasets tend to lack

variation in weather and season. Further, many of these datasets do not provide radar data. Thanks to their longer wavelength, automotive radar sensors are robust to precipitation, dust, and fog. For this reason, radar may play a key role in enabling autonomous vehicles to operate in adverse weather. The Boreas dataset addresses these shortcomings by including a 360° scanning radar and data taken during various weather conditions (sun, cloud, rain, night, snow) and seasons.

Another significant fraction of datasets focus on the problem of localization, usually odometry. The Boreas dataset includes a high-density lidar (128-beam) and a 360° scanning radar. The combination of these sensors and the significant weather variation in this dataset enables detailed comparisons between the localization capabilities of these two sensing modalities; this is something that previous datasets could not support due to either not having a radar sensor or insufficient weather variation. Furthermore, our post-processed ground-truth poses are sufficiently accurate to support a public leaderboard for odometry and metric localization. Another dataset that focuses on adverse weather is RADIATE [136]. Whereas RADIATE focuses on perception, our dataset focuses on localization. Our dataset is larger and includes repeated traversals of a route with higher-quality localization ground truth. Furthermore, our dataset provides higher-resolution radar, lidar, and camera data. For a detailed comparison of related datasets, see Table 4.1.

4.2 Data Collection

Most of the Boreas dataset was collected by driving a repeated route near the University of Toronto over one year. Figure 4.1 illustrates the seasonal variations observed over this time. Figure 4.2 compares camera, lidar, and radar measurements in three distinct weather conditions: falling snow, rain, and sun. The primary repeated route will be referred to as the Glen Shields route and is depicted in Figure 4.4. Additional routes were collected as a single standalone sequence or a small number of repeated traversals. The Glen Shields route can be used for research related to long-term localization, while the other routes allow for experiments that test for generalization to previously unseen environments. The frequency of different metadata tags is displayed in Figure 4.5.

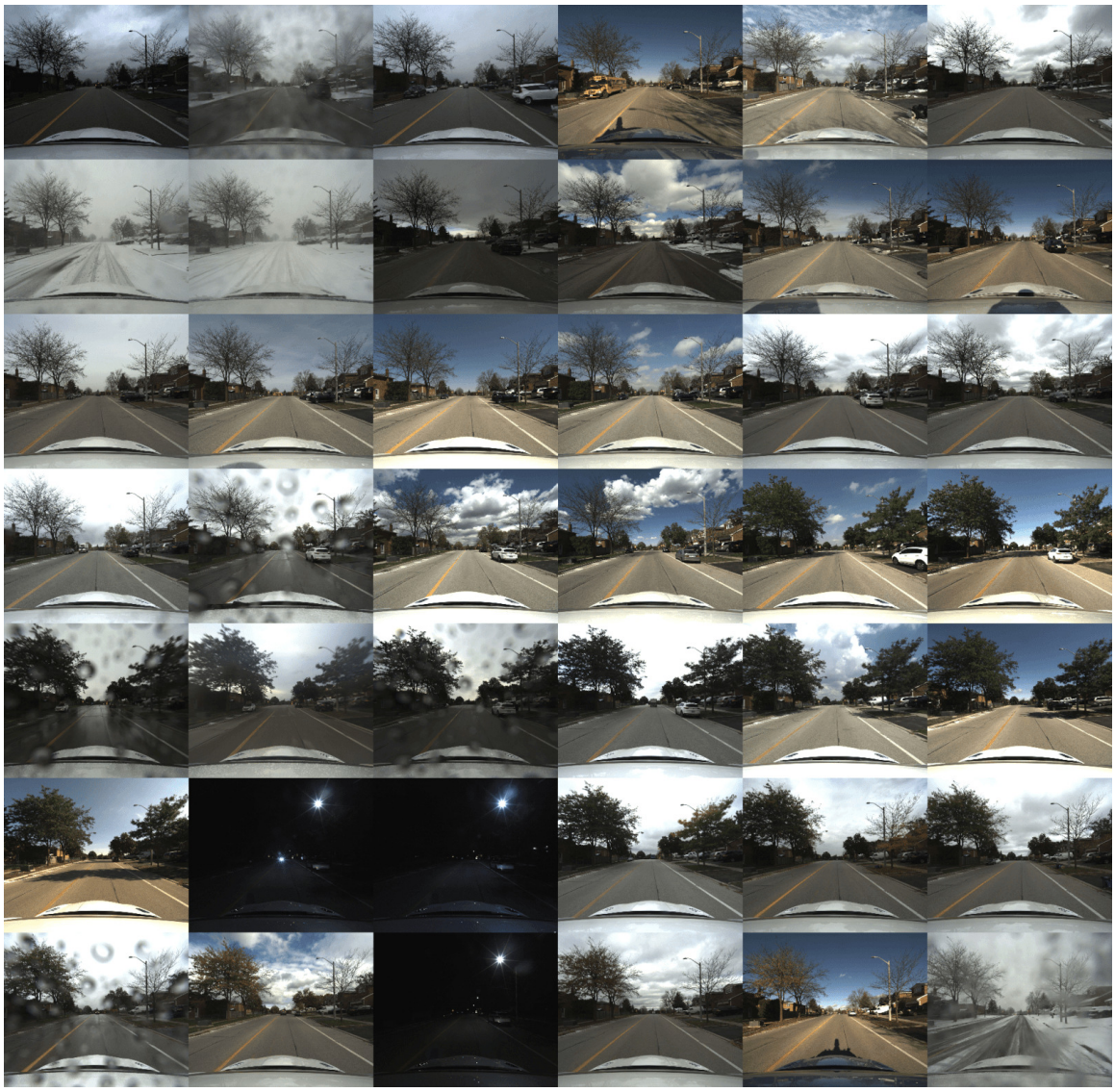


Figure 4.1: This figure depicts one year of seasonal changes in the Boreas dataset. Each image represents a camera image that was taken on a different day. The sequences are sorted in chronological order from left to right and top to bottom, starting in November, 2020 and finishing in November, 2021. Note that the sequences are not evenly spaced in time.

4.3 Sensors

Table 4.2 provides detailed specifications for the sensors used in this dataset. Figures 4.6 and 4.7 illustrate the placement of the different sensors on Boreas.

4.4 Dataset Format

4.4.1 Data Organization

The Boreas dataset is divided into *sequences*, which include all sensor data and ground-truth poses from a single drive. Sequences are identified by the date and time they were collected with the format `boreas-YYYY-MM-DD-HH-MM`. The data for each sequence is organized as shown in Figure 4.8.

4.4.2 Timestamps

The name of each file corresponds to its timestamp. These timestamps are given as UNIX epoch times in microseconds. All sensor timestamps were synchronized to the coordinated universal time (UTC) time reported by the Applanix POS LV. The Velodyne lidar was synchronized using a standard hardwired connection to the Applanix POS LV carrying a pulse-per-second (PPS) signal and NMEA messages. The camera was configured to emit a square-wave pulse where the rising edge of each pulse corresponds with the start of a new camera exposure event. The Applanix POS LV was then configured to receive and timestamp these event signals. Camera timestamps were then corrected in post using the recorded event times and exposure values: $t_{\text{camera}} = t_{\text{event}} + \frac{1}{2}\text{exposure}(t_{\text{event}})$.

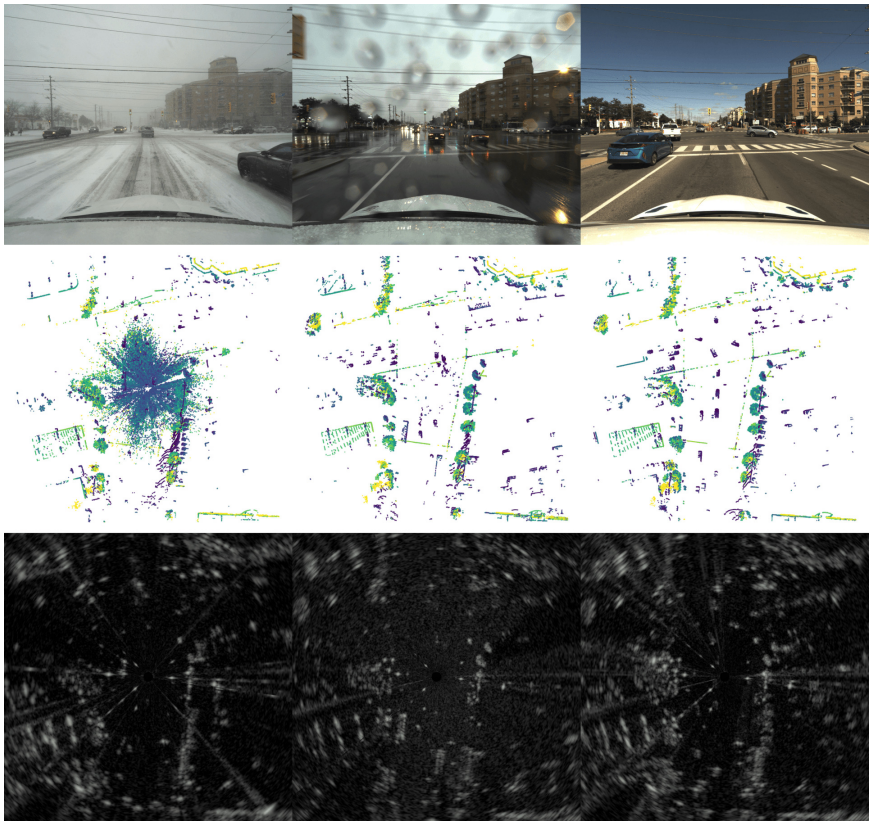


Figure 4.2: Weather variation in the Boreas dataset. Note that the lidar pointcloud becomes littered with detections associated with snowflakes during falling snow and that the radar data remains relatively unperturbed across the weather conditions.

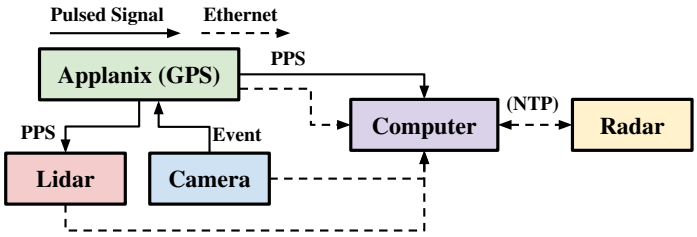


Figure 4.3: Time synchronization of sensors on Boreas.

The data-recording computer was synchronized to UTC in a fashion similar to the Velodyne, using an RS-232 serial cable carrying a PPS signal and NMEA messages. The Navtech radar synchronizes its local clock using network time protocol (NTP). Since the data-recording computer publishing the NTP time is synchronized to UTC, the radar is thereby also synchronized to UTC.

For lidar pointclouds, the timestamp corresponds to the temporal middle of the scan. Each lidar point also has a timestamp associated with it. These point times are given in seconds relative to the middle of the scan. For radar scans, the timestamp also corresponds to the middle of the scan: $\lfloor \frac{M}{2} \rfloor - 1$ where M is the number of azimuths. Each scanned radar azimuth is timestamped in the same format as the filename, a UNIX epoch time. Figure 4.3 shows a diagram of our synchronization setup.

4.4.3 File Formats

Camera images are rectified and anonymized by default. We use Anonymizer to blur license plates and faces [148]. Images are stored in the commonly-used `png` format. Lidar pointclouds are stored in a binary format to minimize storage requirements. Our devkit provides methods for working with these binary formats in C++ and Python. Each point has six fields: $[x, y, z, i, r, t]$ where (x, y, z) is the position of the point with respect to the lidar, i is the intensity of the reflected infrared signal, r is the ID of the laser that made the measurement, and t the point timestamp explained in Section 4.4.2. Raw radar scans are stored as 2D polar images: M azimuths x R range bins. We follow Oxford's convention and embed timestamp and encoder information into the first eleven columns (bytes) of each scanned azimuth. Each scanned azimuth is timestamped individually. Each range bin stores the reflected radar power in dB half-steps. Each row of the polar image corresponds to a scanned azimuth. The first eight columns / bytes of each azimuth / row represent a 64-bit integer, the UNIX epoch timestamp of each azimuth in microseconds. The next two columns represent a 16-bit unsigned integer, the rotational encoder value. The next column is unused but preserved for compatibility with the Oxford format. See [19] for further details on the Navtech sensor and this file format. The polar radar scans can be readily converted into a top-down Cartesian representation, as shown in Figure 4.2.

Figure 4.4: The Glen Shields route in Toronto, Ontario, Canada. Mapbox satellite data was used to generate this figure.

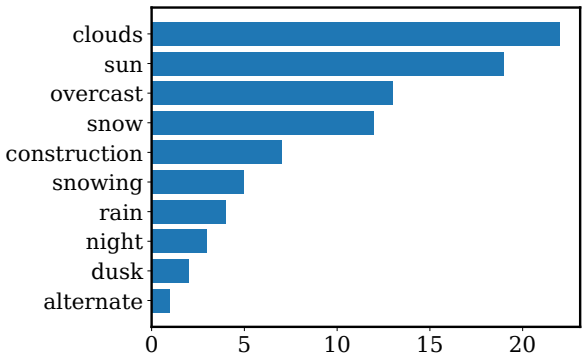


Figure 4.5: Frequency of metadata tags in the Boreas dataset. Snow: snow is on the ground, snowing: it is actively snowing, alternate: a route other than Glen Shields.

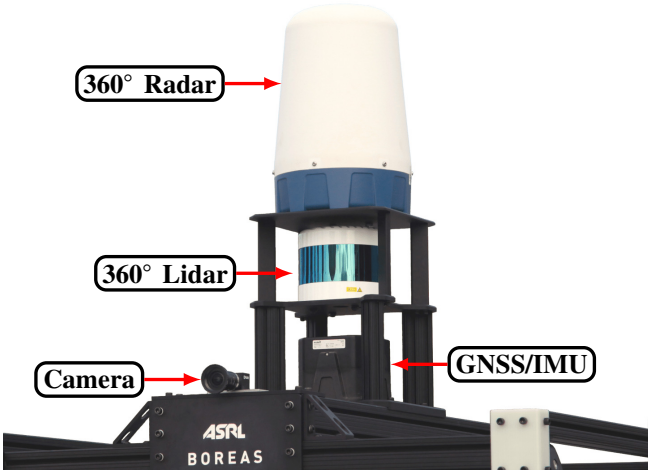


Figure 4.6: A close-up view of Boreas' sensor configuration.

Note that measurements are not synchronous as in other datasets (KITTI [61], CADC [120]), which means that measurements with the same index do not have the same timestamp. However, given the timestamps and relative pose information, different sensor measurements can still be fused together. Lidar pointclouds are not motion-corrected, but we provide methods for removing motion distortion in our devkit. Navtech radar scans suffer from motion distortion and Doppler distortion, [28] and [29] provide methods to compensate for these effects.

Table 4.2: Sensor specifications. †Position accuracy changes over time as a function of the number of visible satellites. †These numbers represent expected accuracy in nominal conditions. ‡Our Navtech radar's firmware was upgraded partway through the project, older sequences have a range resolution of 0.0596m, and a range of 200m.

Sensor	Specifications
Applanix POS LV 220	• 2-4cm RTX accuracy (RMS)† • 200 Hz
Navtech CIR304-H Radar	• 0.0438m range solution‡ • 0.9° horizontal resolution • 250m range‡ • 4 Hz
FLIR Blackfly S Camera (BFS-U3-51S5C)	• 2448x2048 (5 MP) • 81° HFOV x 71° VFOV • 10 Hz
Velodyne Alpha-Prime Lidar	• 128 beams • 0.1° vertical resolution (variable) • 0.2° horizontal resolution • 360° HFOV x 40° VFOV • 300m range (10% reflectivity) • $\sim$ 2.2M points/s • 10 Hz

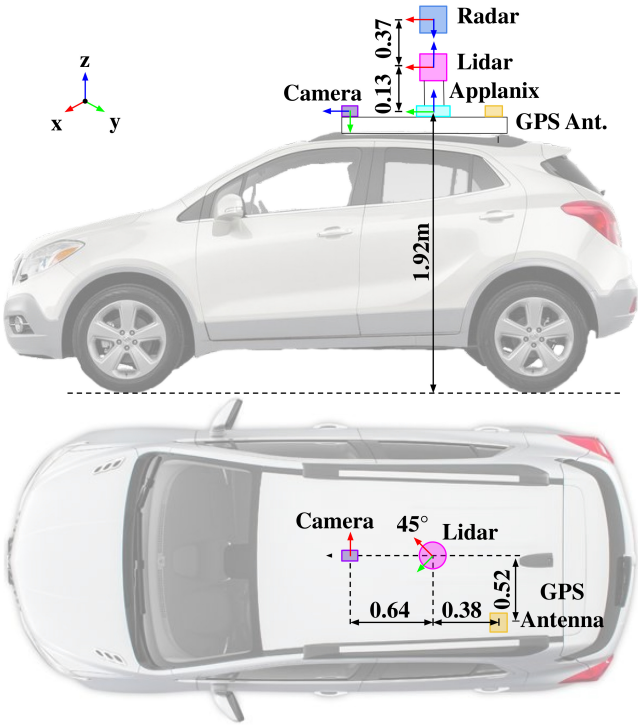


Figure 4.7: Boreas sensor placement. Distances are given in metres. Measurements shown are approximate. Refer to the calibrated extrinsics contained in the dataset for precise measurements.

```
boreas-YYYY-MM-DD-HH-MM
    applanix
        camera_poses.csv
        imu.csv
        gps_post_process.csv
        lidar_poses.csv
        radar_poses.csv
    calib
        camera0_intrinsics.yaml
        P_camera.txt
        T_sens1_sens2.txt
    camera
        <timestamp>.png
    lidar
        <timestamp>.bin
    radar
        <timestamp>.png
    route.html
    video.mp4
```

Figure 4.8: Data organization for a single Boreas sequence.

4.5 Ground-Truth Poses

Ground-truth poses are obtained by post-processing GNSS, IMU, and wheel encoder measurements, along with corrections obtained from an RTX subscription using Applanix's POSPac software suite. Positions and velocities are given with respect to a fixed East-North-Up frame ENU_{ref}. The position

of ENU_{ref} is aligned with the first pose of the first sequence (`boreas-2020-11-26-13-58`), but the orientation is defined to be tangential to the geoid as defined in the WGS-84 convention such that x points East, y points North, and z points up. `applanix/gps_post_process.csv` contains the post-processed ground truth in the Applanix frame at 200Hz for each sequence. We follow the convention used by [16] for describing rotations and 4×4 homogeneous transformation matrices. Each sensor frame's ground truth is stored as a row in `applanix/<sensor>_poses.csv` with the following format: $[t, x, y, z, v_x, v_y, v_z, r, p, y, \omega_z, \omega_y, \omega_x]$ where t is the epoch timestamp in microseconds that matches the filename, $\mathbf{r}_e^{se} = [x \ y \ z]^T$ is the position of the sensor s with respect to ENU_{ref} as measured in ENU_{ref}, $\mathbf{v}_e^{se} = [v_x \ v_y \ v_z]^T$ is the velocity of the sensor with respect to ENU_{ref}, (r, p, y) are the roll, pitch, and yaw angles, which can be converted into a rotation matrix between the sensor frame and ENU_{ref}. $\boldsymbol{\omega}_s^{se} = [\omega_x \ \omega_y \ \omega_z]^T$ are the angular velocities of the sensor with respect to ENU_{ref} as measured in the sensor frame. The pose of the sensor frame is then: $\mathbf{T}_{es} = \begin{bmatrix} \mathbf{C}_{es} & \mathbf{r}_e^{se} \\ \mathbf{0}^T & 1 \end{bmatrix} \in SE(3)$ where $\mathbf{C}_{es} = \mathbf{C}_1(\text{roll})\mathbf{C}_2(\text{pitch})\mathbf{C}_3(\text{yaw})$ [16]. We also provide post-processed IMU measurements in `applanix/imu.csv` at 200Hz in the Applanix frame, including linear acceleration and angular velocity.

The residual root mean square (RMS) position error reported by Applanix is typically less than 5cm in nominal conditions but can be as high as 20-40cm in urban canyons. Figure 4.9 shows the residual RMS errors resulting from the post-processing conducted by the Applanix POSPac software. The estimated error can change depending on the visibility of satellites. Note that these values represent global estimates, and that relative pose estimates are more accurate over short time horizons.

4.6 Calibration

4.6.1 Camera Intrinsics

Camera intrinsics are calibrated using MATLAB's camera calibrator [98] and are recorded in `camera0_intrinsics.yaml`. Images under `camera/` have already been rectified. The rectified camera matrix $\mathbf{P}$ is stored in `P_camera.txt`. To project lidar points onto a camera image, we use the pose of the camera $\mathbf{T}_{ec}$ at time t_c and the pose of the lidar $\mathbf{T}_{el}$ at time t_l to compute a transform from the lidar frame to the camera frame given by $\mathbf{T}_{cl} = \mathbf{T}_{ec}^{-1}\mathbf{T}_{el}$. Each point in the lidar frame is then transformed into the camera frame with $\mathbf{x}_c = \mathbf{T}_{cl}\mathbf{x}_l$, where $\mathbf{x}_l = [x \ y \ z \ 1]^T$. The projected image coordinates are then obtained using [16]:

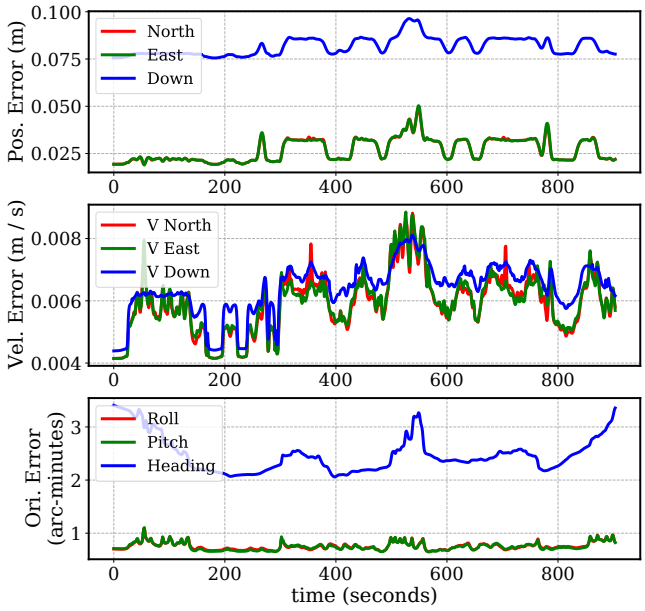


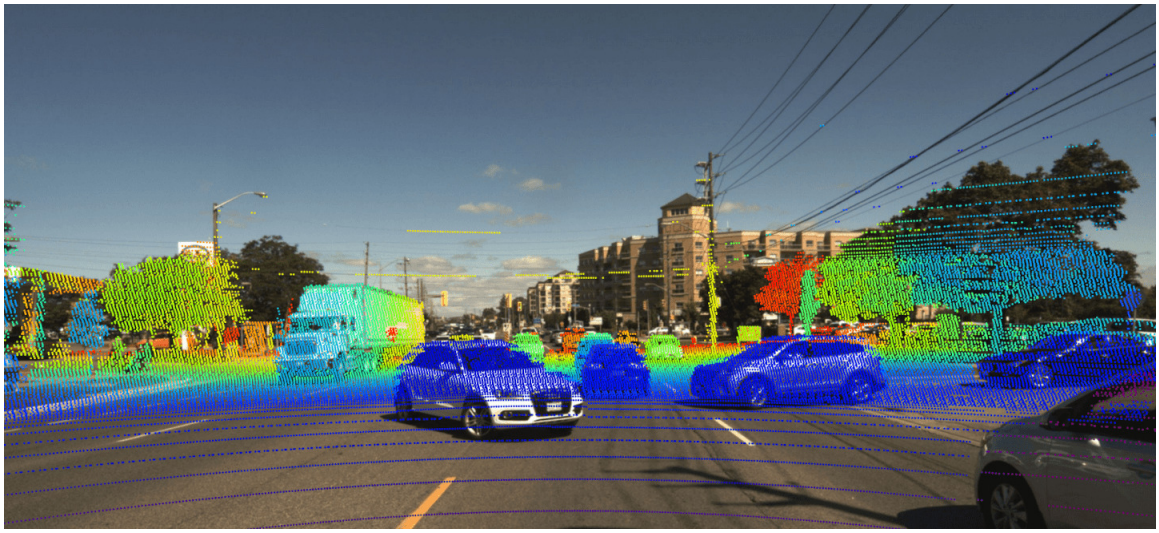
Figure 4.9: Post-processed RMS position, velocity, and orientation residual error vs. time reported by Applanix's POSPac software for a sequence collected on 2021-09-07. Note: an arc-minute is $\frac{1}{60}$ of one degree.

$$\begin{bmatrix} u \\ v \end{bmatrix} = \mathbf{D}\ \mathbf{P}\ \frac{1}{z}\ \mathbf{x}_c \tag{4.1}$$

$$\text{where } \mathbf{D} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \mathbf{P} = \begin{bmatrix} f_u & 0 & c_u & 0 \\ 0 & f_v & c_v & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.2}$$

4.6.2 Sensor Extrinsics

The extrinsic calibration between the camera and lidar is obtained using MATLAB's camera-to-lidar calibrator [98]. The results of this calibration are illustrated in Figure 4.10. We use correlative scan matching via the Fourier Mellin transform [41] to calibrate the radar-to-lidar rotation. Several lidar-radar pairs were collected while the vehicle was stationary at different locations. The final rotation estimate is obtained by averaging the results from several measurement pairs [27]. The translation between the lidar and radar is obtained from the computer-assisted design (CAD) model of the roof rack. The results of the radar-to-lidar calibration are shown in Figure 4.11. The extrinsics between the lidar and the Applanix reference frame were obtained using Applanix's in-house calibration tools. Their tool outputs this relative transform as a by-product of a batch optimization aiming to estimate the most likely vehicle path given a sequence of lidar pointclouds and post-processed GNSS/IMU measurements. All extrinsic calibrations are provided as 4x4 homogeneous transformation matrices under the `calib/` folder.



(a) Perspective View

(b) Coloured Pointcloud

Figure 4.10: Lidar points projected onto a camera image using the camera-lidar calibration. (a) Lidar points are coloured based on their longitudinal distance from the vehicle. (b) Lidar points are given RGB colour values based on their projected location on the camera image.

4.7 3D Annotations

We provide a set of 3D bounding box annotations for a subset of the Boreas dataset obtained in sunny weather. We refer to this as the Boreas-Objects-V1 dataset. Annotations were obtained using the Scale.ai data annotation service [132]. 7111 lidar frames were annotated at 5Hz, resulting in 326,180 unique 3D box annotations. Since the lidar data was collected at 10Hz, the annotations may be interpolated between frames to double the number of annotated frames at a slightly lower fidelity. The data is divided into 53 continuous scenes, each 20-70 seconds long. The scenes are divided into 37 training scenes and 16 test scenes where the ground truth labels have been withheld for the benchmark. Figure 4.12 displays two statistics for our annotations.

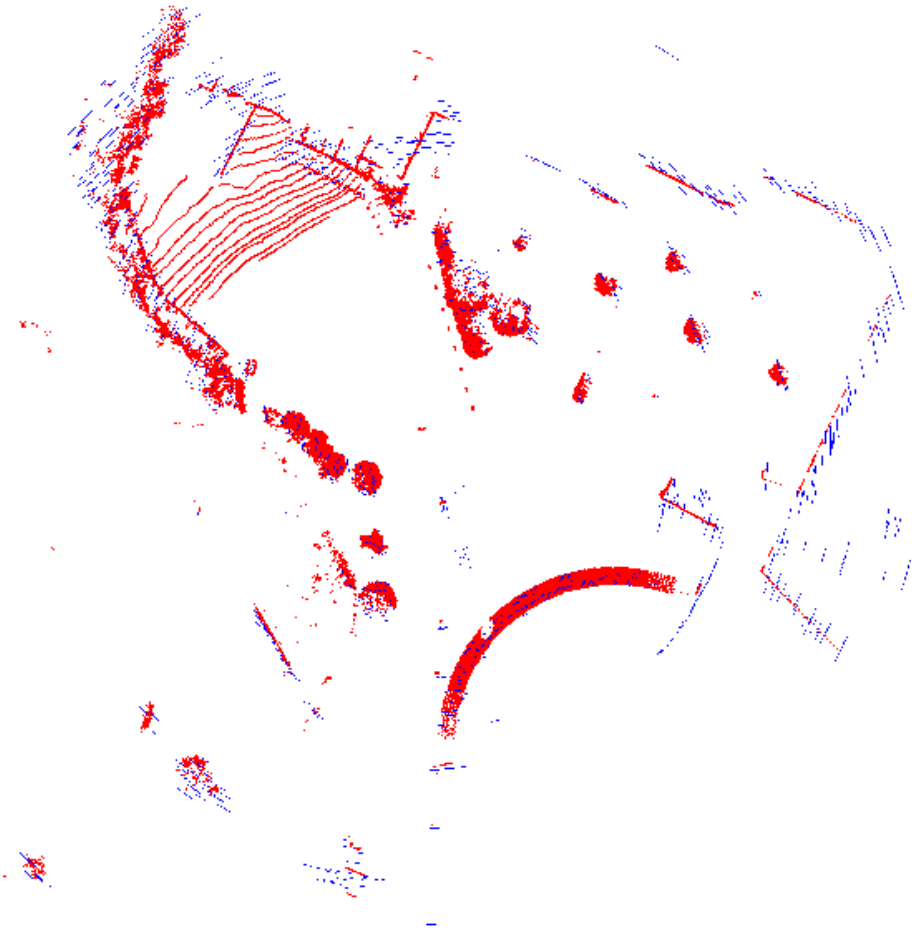


Figure 4.11: Lidar measurements are drawn in red using a bird's eye view projection with the ground plane removed. Radar targets are first extracted from the raw radar data and then are drawn as blue pixels. The two sensors have been aligned using the radar-to-lidar calibration.

We use the same folder structure as in Figure 4.8 but with an additional folder, `labels/`. Similar to KITTI, annotations for a particular frame are stored in a text file with the same filename (timestamp) as the lidar frame. Each row of a label file corresponds to a different 3D box annotation with the format: $[\text{uuid}, \text{type}, d_x, d_y, d_z, x, y, z, \text{yaw}]$. The uuid is a unique ID for a particular object track consistent across frames within a particular scene. The type is the semantic class for an object that can be one of: {Car, Cyclist, Pedestrian, Misc}. The Car class includes coupes, sedans, SUVs, vans, pick-up trucks, and ambulances. The Cyclist class includes people riding motorcycles but excludes parked bicycles. The Misc class includes buses, industrial trucks, streetcars, and trains. Objects are labelled within a rectangular area centred on the lidar +/- 75m in both dimensions. Bounding box locations (x, y, z) and orientations (yaw) are given with respect to the lidar frame. (d_x, d_y, d_z) represent the bounding box dimensions (length, width, and height). Figure 4.13 shows an example of our 3D object annotations for lidar, camera, and radar.

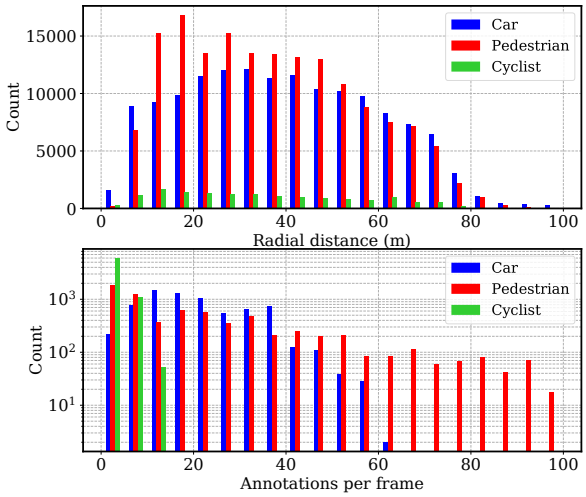


Figure 4.12: 3D annotation statistics for Boreas-Objects-V1.

Figure 4.13: Examples of 3D annotations in the Boreas-Objects-V1 dataset.

4.8 Benchmark Metrics

We support online leaderboards for odometry, metric localization, and 3D object detection. For odometry, we use the same metrics as the KITTI dataset [61]. The KITTI odometry metrics average the relative position and orientation errors over every sub-sequence of length (100m, 200m, 300m, ..., 800m); this results in two metrics, a translational drift reported as a percentage of path length and a rotational drift reported as degrees per metre travelled. For 3D object detection, we also defer to the KITTI dataset by reporting the mean average precision (mAP) on a per-class basis. For cars, a 70% overlap counts as a true positive, and for pedestrians, 50%. These ratios are used as they are the same as those used in the KITTI dataset. We do not divide our dataset based on difficulty levels.

Our metric localization leaderboard aims to benchmark mapping and localization pipelines. In this scenario, we envision a situation where one or more repeated traversals of the Glen Shields route are used to construct a map offline. Any data from the training sequences may be used to construct

a map. Then, during a test sequence, the goal would be to perform metric localization between the live sensor data and the pre-built map. Localization approaches may use temporal filtering and can leverage the IMU if desired, but GNSS information will not be available. The goal of this benchmark is to simulate localizing a vehicle in real time; as such, methods may not use future sensor information in an acausal manner. Our goal is to support both global and relative map structures. Only one of the training sequences will be specified as the map sequence used by the benchmark. For 3D localization, users must choose either the lidar or the camera as the reference sensor. For 2D localization, only the radar frames are used as a reference. For each (camera—lidar—radar) frame s_2 in the test sequence, users will specify the ID (timestamp) of the (camera—lidar—radar) frame s_1 in the map sequence that they are providing a relative pose with respect to: $\hat{\mathbf{T}}_{s_1,s_2}$. We then compute root-mean-squared error (RMSE) values for the translation and rotation as follows:

$$\mathbf{T}_e = \mathbf{T}_{a,s_1}\mathbf{T}_{s_1,s_2}\hat{\mathbf{T}}_{s_1,s_2}^{-1}\mathbf{T}_{a,s_1}^{-1} = \begin{bmatrix}\mathbf{C}_e & \mathbf{r}_e \\ \mathbf{0}^T & 1\end{bmatrix}, \tag{4.3}$$

$$\mathbf{r}_e = \begin{bmatrix}x_e & y_e & z_e\end{bmatrix}^T, \tag{4.4}$$

$$\phi_e = \arccos\left(\frac{\text{tr }\mathbf{C}_e - 1}{2}\right), \tag{4.5}$$

where $\mathbf{T}_{s_1,s_2}$ is the known ground truth pose, and $\mathbf{T}_{a,s_1}$ is the calibrated transform from the sensor frame to the Applanix frame (x-right, y-forwards, z-up). x_e, y_e, z_e are the lateral, longitudinal, and vertical errors, respectively. We calculate RMSE values for x_e, y_e, z_e, ϕ_e.

Users can also provide 6×6 covariance matrices $\mathbf{\Sigma}_i$ for each localization estimate. A pose with uncertainty is described as $\mathbf{T} = \exp(\boldsymbol{\xi}^\wedge)\overline{\mathbf{T}}$ where $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$ [16]. Given $\hat{\mathbf{T}}_i = \hat{\mathbf{T}}_{s_1,s_2}(t_i)$, we compute an average consistency score c for the localization and covariance estimates where

$$\boldsymbol{\xi}_i = \ln\left(\mathbf{T}_i\hat{\mathbf{T}}_i^{-1}\right)^\vee = \begin{bmatrix}\rho_1 & \rho_2 & \rho_3 & \psi_1 & \psi_2 & \psi_3\end{bmatrix}^T, \tag{4.6}$$

$$c = \left(\sum_{i=1}^{N}\frac{\boldsymbol{\xi}_i^T\mathbf{\Sigma}_i^{-1}\boldsymbol{\xi}_i}{N\text{dim}(\boldsymbol{\xi}_i)}\right)^{1/2}. \tag{4.7}$$

A consistency score close to 1 is ideal. $c < 1$ means that the method is over-confident, $c > 1$ means that the method is under-confident. Note that the above metrics will be averaged across the test sequences.

4.9 Development Kit

As part of this dataset, we provide a development kit for new users to get started. The primary purpose of the devkit is to act as a wrapper around the dataset to be used in Python; this allows users to query frames and the associated ground truth for either odometry, localization, or 3D object detection. We also provide convenient methods for removing motion distortion from pointclouds, working with polar radar scans, and converting to and from Lie algebra and Lie group representations. The devkit also provides several ways to visualize sensor data. We also provide introductory Jupyter notebooks tutorials, including projecting lidar onto a camera frame and visualizing 3D boxes. Evaluation scripts used by our benchmark will be stored in the devkit, allowing users to

validate their algorithms before submission to the benchmark. The development kit can be found at boreas.utias.utoronto.ca.

4.10 Conclusions

In this chapter, we presented Boreas, a multi-season autonomous driving dataset with over 350km of driving data collected over one year. The dataset provides a unique, high-quality sensor suite including a Velodyne Alpha-Prime (128-beam) lidar, a 5MP camera, a 360° Navtech radar, and accurate ground-truth poses obtained from an Applanix POS LV with an RTX subscription. We also provide 3D object labels for a subset of the Boreas data obtained in sunny weather. The primary purpose of this dataset is to enable further research into long-term localization across seasons and adverse weather conditions. Our website provides an online leaderboard for odometry, metric localization, and 3D object detection. Given this dataset, we can now focus on some of the challenges inherent to working with spinning radar, such as motion distortion and Doppler effects.

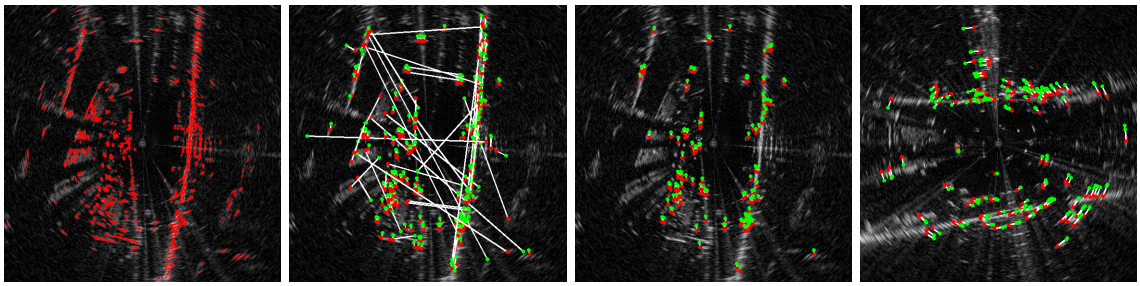
Chapter 5

Motion Distortion and Doppler Effects in Spinning Radar Navigation

Prior works made tremendous progress in applying the Navtech radar to odometry [36, 37, 5, 20, 18, 114] and place recognition [131, 58, 79]. However, most of these works made the simplifying assumption that a radar scan is collected at a single instant in time. In reality, the sensor is rotating while the vehicle is moving, causing the radar scan to be distorted in a cork-screw fashion. Range measurements of the Navtech radar are also impacted by Doppler frequency shifts resulting from the relative velocity between the sensor and its surroundings. Both distortion effects become more pronounced as the speed of the ego-vehicle increases. Most automotive radar sensors are unaffected by either distortion effect. However, the Navtech radar provides 360° coverage with accurate range and azimuth resolution, making it an appealing navigation sensor.

In this chapter, we quantify motion distortion's effect on radar-based navigation. We also revisit a lightweight estimator, Motion-Compensated RANSAC [9], which can recover the motion between a pair of scans and remove the distortion. Motion distortion and Doppler distortion were previously described by Rouveure et al. [129] who provided a method for compensating for these effects given proprioceptive sensors measuring linear and angular velocity such as a wheel odometer and a gyroscope. Doppler distortion was also briefly acknowledged in [36]. Our work is the first to quantify the impact on radar-based navigation and to provide a method to compensate for it without requiring an auxilliary sensor.

As our primary experiment to demonstrate the effects of motion distortion, we perform radar odometry on the Oxford Radar RobotCar Dataset [19]. As an additional experiment, we perform metric localization using our own data-taking platform, shown in Figure 1.1. Qualitative results of both distortion effects are also provided. Rather than focusing on achieving state-of-the-art navigation results, this chapter aims to show that motion distortion and Doppler effects are significant and can be compensated for with relative ease. Motion distortion has been treated in the literature through the use of continuous-time trajectory estimation [9, 15, 11, 10, 57] for lidars [22] and rolling-shutter cameras [66], but these tools are yet to be applied to spinning radar. This chapter focuses



(a) Feature Extraction (b) Data Association (c) MC-RANSAC Inliers (d) MC-RANSAC Inliers Rotating

Figure 5.1: This figure illustrates our feature extraction and matching process. (a) displays our raw extracted features. (b) displays the output of our ORB-descriptor-based data association. (c) and (d) show the inlier set resulting from motion-compensated RANSAC while driving straight and rotating.

on the problem of motion distortion and Doppler effects using the Navtech radar sensor, which has not received attention in these prior works.

5.1 Methodology

5.1.1 Feature Extraction

Feature detection in radar data is more challenging than in lidar or vision due to its higher noise floor and lower spatial resolution. Constant False Alarm Rate (CFAR) [126] is a simple feature detector popular for radar. CFAR is designed to estimate the local noise floor and capture relative peaks in the radar data. One-dimensional CFAR can be applied to Navtech data by convolving each azimuth with a sliding window detector.

As discussed in [36], CFAR produces many redundant keypoints, is difficult to tune, and produces false positives due to the noise artifacts present in radar. Instead, Cen et al. [36] proposed a detector that estimates a signal's noise statistics and then scales the power at each range by the probability that it is a real detection. In [37], Cen et al. proposed an alternative detector that identifies continuous regions of the scan with high intensity and low gradients. Keypoints are then extracted by locating the middle of each continuous region. We will refer to these detectors as Cen2018 and Cen2019, respectively.

The original formulations of these detectors did not lend themselves to real-time operation. As such, we made several modifications to improve the runtime. For Cen2018, we use a Gaussian filter instead of a binomial filter, and we calculate the mean of each azimuth instead of using a median filter. We do not remove multipath reflections.

Cen2019 was designed to be easier to tune and have fewer redundant keypoints. However, we found that by adjusting the probability threshold of detections, Cen2018 obtained better odometry performance when combined with our RANSAC-based scan matching. Based on these preliminary tests, we concluded that Cen2018 was the best choice for our experiments. Figure 5.1(a) shows Cen2018 features plotted on top of a Cartesian radar image.

We convert the raw radar scans output by the Navtech sensor, which are in polar form, into

Cartesian form. We then calculate an ORB descriptor [130] for each keypoint on the Cartesian image. There may be better keypoint descriptors for radar data, such as the learned feature descriptors employed in [18]. However, ORB descriptors are quick to implement, rotationally invariant, and resistant to noise.

For data association, we perform brute-force matching of ORB descriptors. We then apply a nearest-neighbor distance ratio test [93] to remove false matches. The remaining matches are sent to our RANSAC-based estimators. Figure 5.1(b) shows the result of the initial data association. Note that there are several outliers. Figure 5.1(c),(d) shows the remaining inliers after performing RANSAC.

5.1.2 Motion Distortion

The output of data association is not perfect and often contains outliers. As a result, it is common to employ an additional outlier rejection scheme during estimation. In this chapter, we use RANSAC [54] to find an outlier-free set that can then be used to estimate the desired transform. If we assume that two radar scans are taken at times $\bar{t}_1$ and $\bar{t}_2$, then the transformation between them can be estimated directly using the closed-form solution to aligning two 3D point sets with known correspondences by Arun et al. [13].

During each iteration of RANSAC, a random subset of size S is drawn from the initial matches, and a candidate transform is generated. The algorithm terminates if the number of inliers exceeds the desired threshold or a maximum number of iterations is reached. Radar scans are 2D, so we use $S = 2$. The estimation process is repeated on the largest inlier set to obtain a more accurate transform. We will refer to this approach as *rigid* RANSAC.

Our derivation of motion-compensated RANSAC follows closely from [9]. However, we are applying the algorithm to a scanning radar in 2D instead of a two-axis scanning lidar in 3D. Furthermore, our derivation is shorter and uses updated notation from [16].

The principal idea behind motion-compensated RANSAC is to estimate the velocity of the sensor instead of estimating a transformation. We make the simplifying assumption that the linear and angular velocity between a pair of scans is constant. To account for motion distortion, we remove the assumption that radar scans are taken at a single instant in time. Data association produces two sets of corresponding measurements, $\mathbf{y}_{m,1}$ and $\mathbf{y}_{m,2}$, where $m = 1...M$. Each pair of features, m, is extracted from sequential radar frames 1 and 2 at times $t_{m,1}$ and $t_{m,2}$. The temporal difference between a pair of measurements is $\Delta t_m := t_{m,2} - t_{m,1}$. The generative model for measurements is given as

$$\begin{aligned} \mathbf{y}_{m,1} &:= \mathbf{f}(\mathbf{T}_s(t_{m,1})\mathbf{p}_m) + \mathbf{n}_{m,1}, \\ \mathbf{y}_{m,2} &:= \mathbf{f}(\mathbf{T}_s(t_{m,2})\mathbf{p}_m) + \mathbf{n}_{m,2}, \end{aligned} \tag{5.1}$$

where $\mathbf{f}(\cdot)$ is a nonlinear transformation from Cartesian to cylindrical coordinates and $\mathbf{T}_s(t)$ is a 4 x 4 homogeneous transformation matrix representing the pose of the sensor frame $\boldsymbol{\mathcal{F}}_{s}$ with respect to the inertial frame $\boldsymbol{\mathcal{F}}_{i}$ at time t. $\mathbf{p}_m$ is the original landmark location in the inertial frame. We assume that each measurement is corrupted by zero-mean Gaussian noise: $\mathbf{n}_{m,1} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{m,1})$. The

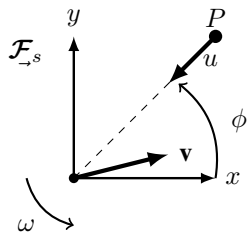


Figure 5.2: This diagram illustrates the relationship between the ego-motion $(\mathbf{v}, \omega)$ and the radial velocity u.

transformation between a pair of measurements is defined as

$$\mathbf{T}_m := \mathbf{T}_s(t_{m,2})\mathbf{T}_s(t_{m,1})^{-1}. \tag{5.2}$$

To obtain our objective function, we convert feature locations from polar coordinates into local Cartesian coordinates:

$$\mathbf{p}_{m,2} = \mathbf{f}^{-1}(\mathbf{y}_{m,2}), \tag{5.3}$$

$$\mathbf{p}_{m,1} = \mathbf{f}^{-1}(\mathbf{y}_{m,1}). \tag{5.4}$$

We then use the local transformation $\mathbf{T}_m$ to create a pseudomeasurement $\hat{\mathbf{p}}_{m,2}$,

$$\hat{\mathbf{p}}_{m,2} = \mathbf{T}_m\mathbf{p}_{m,1}. \tag{5.5}$$

The error is then defined in the sensor coordinates and summed over each pair of measurements to obtain our objective function:

$$\mathbf{e}_m = \mathbf{p}_{m,2} - \hat{\mathbf{p}}_{m,2}, \tag{5.6}$$

$$J(\boldsymbol{\varpi}) := \frac{1}{2}\sum_{m=1}^{M} \mathbf{e}_m^T \mathbf{R}_{\text{cart},m}^{-1}\mathbf{e}_m. \tag{5.7}$$

Given our constant-velocity assumption, we can convert from a velocity vector $\boldsymbol{\varpi}$ into a transformation matrix using the following formula:

$$\mathbf{T} = \exp(\Delta t\boldsymbol{\varpi}^\wedge). \tag{5.8}$$

In order to optimize our objective function $J(\boldsymbol{\varpi})$, we first need to derive the relationship between $\mathbf{T}_m$ and $\boldsymbol{\varpi}$. The velocity vector $\boldsymbol{\varpi}$ can be written as the sum of a nominal velocity $\overline{\boldsymbol{\varpi}}$ and a small perturbation $\delta\boldsymbol{\varpi}$. This lets us rewrite the transformation $\mathbf{T}_m$ as the product of a nominal transformation $\overline{\mathbf{T}}_m$ and a small perturbation $\delta\mathbf{T}_m$:

$$\mathbf{T}_m = \exp(\Delta t_m(\overline{\boldsymbol{\varpi}} + \delta\boldsymbol{\varpi})^\wedge) = \delta\mathbf{T}_m\overline{\mathbf{T}}_m. \tag{5.9}$$

Let $\mathbf{g}_m(\boldsymbol{\varpi}) := \mathbf{T}_m\mathbf{p}_{m,1}$, which is nonlinear due to the transformation. Our goal is to linearize $\mathbf{g}_m(\boldsymbol{\varpi})$

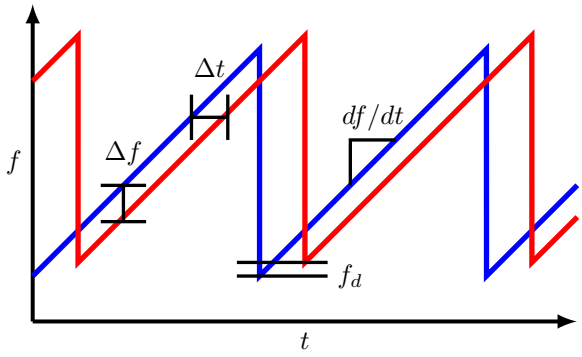


Figure 5.3: This figure depicts the sawtooth modulation pattern of an FMCW radar. The transmitted signal is blue and the received signal is red.

about a nominal operating point. We can rewrite $\mathbf{g}_m(\boldsymbol{\varpi})$ as:

$$\begin{aligned}\mathbf{g}_m(\boldsymbol{\varpi}) &= \exp(\Delta t_m \delta\boldsymbol{\varpi}^\wedge)\overline{\mathbf{T}}_m \mathbf{p}_{m,1}, \\ &\approx (\mathbf{1} + \Delta t_m \delta\boldsymbol{\varpi}^\wedge)\overline{\mathbf{T}}_m \mathbf{p}_{m,1},\end{aligned} \tag{5.10}$$

where we use an approximation for small pose changes. We swap the order of operations using the $(\cdot)^\odot$ operator [16]:

$$\begin{aligned}\mathbf{g}_m(\boldsymbol{\varpi}) &= \overline{\mathbf{T}}_m \mathbf{p}_{m,1} + \Delta t_m (\overline{\mathbf{T}}_m \mathbf{p}_{m,1})^\odot \delta\boldsymbol{\varpi} \\ &= \bar{\mathbf{g}}_m + \mathbf{G}_m \delta\boldsymbol{\varpi},\end{aligned} \tag{5.11}$$

We can now rewrite the error function from (5.6):

$$\begin{aligned}\mathbf{e}_m &\approx \mathbf{p}_{m,2} - \bar{\mathbf{g}}_m - \mathbf{G}_m \delta\boldsymbol{\varpi} \\ &= \bar{\mathbf{e}}_m - \mathbf{G}_m \delta\boldsymbol{\varpi}.\end{aligned} \tag{5.12}$$

By inserting this equation for the error function into the objective function from (5.7), and taking the derivative with respect to the perturbation and setting it to zero, $\frac{\partial J(\boldsymbol{\varpi})}{\partial \delta\boldsymbol{\varpi}^T} = \mathbf{0}$, we obtain the optimal update:

$$\delta\boldsymbol{\varpi}^\star = \left(\sum_m \mathbf{G}_m^T \mathbf{R}_{\text{cart},m}^{-1} \mathbf{G}_m\right)^{-1} \left(\sum_m \mathbf{G}_m^T \mathbf{R}_{\text{cart},m}^{-1} \bar{\mathbf{e}}_m\right), \tag{5.13}$$

where $\mathbf{R}_{\text{cart},m} = \mathbf{H}_m \mathbf{R}_{m,2} \mathbf{H}_m^T$ is the covariance in the local Cartesian frame, $\mathbf{h}(\cdot) = \mathbf{f}^{-1}(\cdot)$, and $\mathbf{H}_m = \frac{\partial \mathbf{h}}{\partial \mathbf{x}}\big|_{\bar{\mathbf{g}}_m}$. The optimal perturbation $\delta\boldsymbol{\varpi}^\star$ is used in a Gauss-Newton optimization scheme and the process repeats until $\boldsymbol{\varpi}$ converges.

This method allows us to estimate the linear and angular velocity between radar scans directly while accounting for motion distortion. These velocity estimates can then be used to remove the motion distortion from a measurement relative to a reference time using (5.8). MC-RANSAC is intended to be a lightweight method for showcasing the effects of motion distortion. A significant improvement to this pipeline would be to use the inliers of MC-RANSAC as an input to another

estimator, such as [10]. The inliers could also be used for mapping and localization or SLAM.

5.1.3 Doppler Correction

To compensate for the Doppler effects, we need to know the linear velocity of the sensor v; this can either be obtained from a GPS/IMU or using the velocity estimated by MC-RANSAC. The motion of the sensor results in a relative velocity between the sensor and its surrounding environment as depicted Figure 5.2. This relative velocity causes the received frequency to be altered according to the Doppler effect. Note that only the radial component of the velocity $u = v_x \cos(\phi) + v_y \sin(\phi)$ will result in a Doppler shift. The Radar Handbook by Skolnik [138] provides an expression for the Doppler frequency:

$$f_d = \frac{2u}{\lambda}, \tag{5.14}$$

where λ is the wavelength of the signal. Note that for an object moving towards the radar ($u > 0$) or vice versa, the Doppler frequency will be positive, resulting in a higher received frequency. For FMCW radar such as the Navtech sensor, the distance to a target is determined by measuring the change in frequency between the received signal and the carrier wave Δf:

$$r = \frac{c\Delta f}{2(df/dt)}, \tag{5.15}$$

where df/dt is the slope of the modulation pattern used by the carrier wave and c is the speed of light. FMCW radar requires two measurements to disentangle the frequency shift resulting from range and relative velocity. Since the Navtech sensor scans each azimuth only once, the measured frequency shift is the combination of both the range difference and Doppler frequency. From Figure 5.3, we can see that a positive Doppler frequency f_d will increase the received frequency and reduce the observed frequency difference Δf. Thus, a positive Doppler frequency will decrease the apparent range of a target.

The Navtech radar operates between 76 GHz and 77 GHz, resulting in a bandwidth of 1 GHz. Navtech states that they use a sawtooth modulation pattern. Given 1600 measurements per second and assuming the entire bandwidth is used for each measurement, $df/dt \approx 1.6 \times 10^{12}$.

Hence, if the forward velocity of the sensor is 1 m/s, a target positioned along the x-axis (forward) of the sensor would experience a Doppler frequency shift of 510 Hz using (5.14). This increase in the frequency of the received signal would decrease the apparent range to the target by 4.8 cm using (5.15). Naturally, this effect becomes more pronounced as the velocity increases.

Let $\beta = f_t/(df/dt)$ where f_t is the transmission frequency ($f_t \approx 76.5$ GHz). To correct for the Doppler distortion, the range of each target needs to be corrected by the following factor:

$$\Delta r_{\text{corr}} = \beta(v_x \cos(\phi) + v_y \sin(\phi)). \tag{5.16}$$

We use this simple correction in all our experiments with the velocity (v_x, v_y) coming from our motion estimator.

Table 5.1: Radar Odometry Results. *Odometry results from [18].

Method	Trans. Error (%)	% Improve (Trans.)	Rot. Error (deg/m)	% Improve (Rot.)
RO Cen* [36]	3.7168	-	0.0095	-
Rigid RANSAC	3.9777	-7.02	0.0141	-48.42
MC-RANSAC	3.6042	3.03	0.0119	-25.26
MC-RANSAC + Doppler	3.5847	3.56	0.0118	-24.21

5.2 Experimental Results

To answer the question posed by this chapter, we have devised two experiments. The first is to compare the performance of rigid RANSAC and MC-RANSAC (with or without Doppler corrections) on radar odometry using the Oxford Radar RobotCar Dataset [19]. The second experiment demonstrates the impact of both distortion effects on localization using our data. We also provide qualitative results demonstrating both distortion effects.

The Navtech radar is a frequency-modulated continuous wave (FMCW) radar. The sensor outputs the received signal power at each range bin for each azimuth. The sensor spins at 4 Hz and provides 400 measurements per rotation with a 163 m range, 4.32 cm range resolution, 1.8° horizontal beamwidth, and 0.9° azimuth resolution.

The Oxford Radar RobotCar Dataset is an autonomous driving dataset that includes two 32-beam lidars, six cameras, a GPS/IMU, and the Navtech sensor. The dataset includes thirty-two traversals equating to 280 km of driving in total.

5.2.1 Odometry

Our goal is to compare two estimators whose main difference is the compensation of distortion effects. We use the same number of maximum iterations (100) and the same inlier threshold (0.35 m) for both rigid RANSAC and MC-RANSAC. We also fix the random seed before running either estimator to ensure that the differences in performance are not due to the random selection of subsets.

For feature extraction, we use the same setup parameters for Cen2018 as in [36] except that we use a higher probability threshold, $z_q = 3.0$, and a Gaussian filter with $\sigma = 17$ range bins. We convert polar radar scans into a Cartesian image with 0.2592 m per pixel and a width of 964 pixels (250 m). We use a patch size of 21 pixels (5.4 m) for ORB descriptors. For data association, we use a nearest-neighbor distance ratio of 0.8. For Doppler corrections, we use $\beta = 0.049$. For each radar scan, extracting Cen2018 features takes approximately 35 ms. Rigid RANSAC runs in 1-2 ms and MC-RANSAC in 20-50 ms. Calculating orb descriptors takes 5 ms, and brute-force matching takes 15 ms. These processing times were obtained on a quad-core Intel Xeon E3-1505M 3.0 GHz CPU with 32 GB of RAM.

Odometry results are obtained by compounding the frame-to-frame scan matching results. Note that we do not use a motion prior or perform additional smoothing on the odometry in order to focus on frame-to-frame matching performance. Three sequences were used for parameter tuning. The remaining 29 sequences are used to provide test results.

Table 5.1 summarizes the results of the odometry experiment. We use KITTI-style odometry

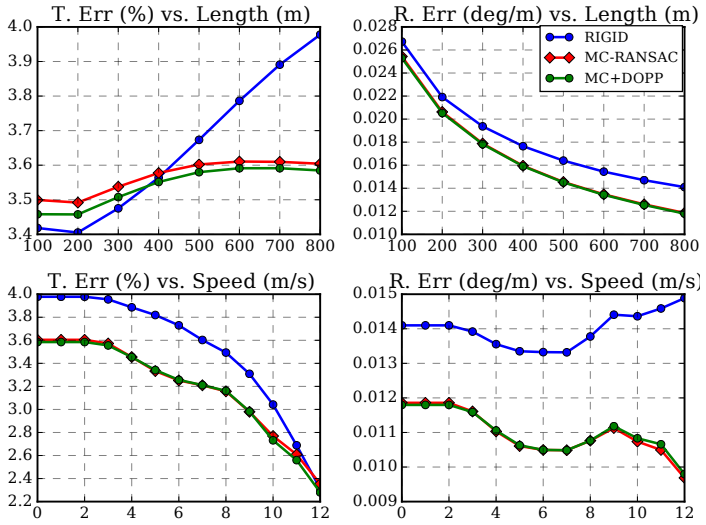


Figure 5.4: This figure provides our KITTI-style odometry results on the Oxford dataset. We provide our translational and rotational drift as a function of path length and speed. MC-RANSAC: motion-compensated RANSAC, MC+DOPP: motion and distortion compensated.

metrics [61] to quantify the translational and rotational drift as is done in [18]. The metrics are obtained by averaging the translational and rotational drifts for all subsequences of lengths (100, 200, ..., 800) metres. The table shows that motion-compensated RANSAC results in a 9.4% reduction in translational drift and a 15.6% reduction in rotational drift; this shows that compensating for motion distortion has a modest impact on radar odometry. The table also indicates that Doppler effects have a negligible impact on odometry. Our interpretation is that Doppler distortion impacts sequential radar scans in a similar manner, such that scan registration is minimally impacted. Notably, a large fraction of the Oxford dataset was collected at low speeds (0-5 m/s); this causes the motion distortion and Doppler effects to be less noticeable.

Figure 5.4 depicts each method's translational and rotational drift as a function of path length and speed. Motion compensation improves the odometry performance across most cases. Interestingly, MC-RANSAC does not increase in error as much as rigid RANSAC as the path length increases. Naturally, we expect rigid RANSAC to become much worse than MC-RANSAC at higher speeds. However, we observe that as the speed of the vehicle increases, the motion tends to become more linear (as in driving straight with less turns). When the vehicle's motion is mostly linear, the motion distortion does not impact rigid RANSAC as much. Figure 5.5 compares the odometry output of both estimators against ground truth. The results further illustrate that compensating for motion distortion improves performance.

5.2.2 Localization

This experiment aims to demonstrate the impact of motion distortion and Doppler effects on metric localization. Unlike the previous experiment, we localize between scans taken while driving in opposite directions. While most of the Oxford Radar dataset was captured at low speeds (0-10 m/s), in this experiment, we only used radar frames where the ego-vehicle's speed was above 10 m/s. For this experiment, we use our data-taking platform, shown in Figure 1.1, which includes a

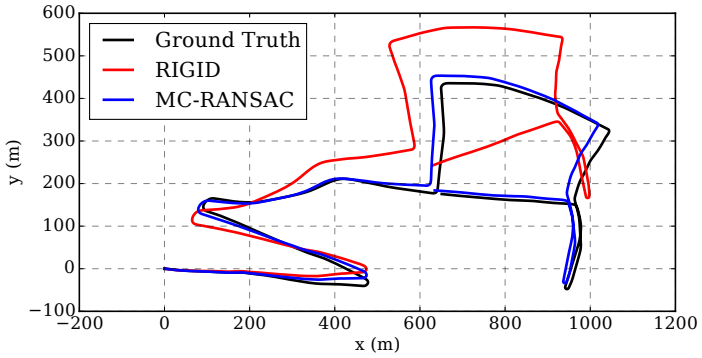


Figure 5.5: This figure highlights the impact that motion distortion can have on the accuracy of radar-based odometry. Note that motion-compensated RANSAC (MC-RANSAC) is much closer to the ground truth.

Velodyne Alpha-Prime lidar, Navtech CIR204-H radar, Blackfly S camera, and an Applanix POSLV GNSS.

Ground truth for this experiment was obtained from a 10 km drive using post-processed GNSS data provided by Applanix, which has an accuracy of 12 cm. Radar scans were initially matched by identifying pairs of proximal scans on the outgoing and return trips based on GPS data. The Navtech timestamps were synchronized to GPS time for an accurate position estimate.

Our first observation was that localizing against a drive in reverse is harder than odometry where scans are matched with roughly the same orientation. When viewed from different angles, objects have different radar cross-sections, which causes them to appear differently. Consequently, radar scans may lose or gain features when pointed in the opposite direction. This change in the radar scan's appearance prevented ORB features from matching.

As a replacement for ORB descriptors, we turned to the Radial Statistics Descriptor (RSD) described in [36], [37]. Instead of calculating descriptors based on the Cartesian radar image, RSD operates on a binary Cartesian grid derived from the detected feature locations. This grid can be thought of as a radar target occupancy grid. For each keypoint, RSD divides the binary grid into M azimuth slices and N range bins centred around the keypoint. The number of keypoints (pixels) in each azimuth slice and range bin is counted to create two histograms. In [37], a fast Fourier transform of the azimuth histogram is concatenated with a normalized range histogram to form the final descriptor.

In our experiment, we found that the range histogram was sufficient on its own, with the azimuth histogram offering only a minor improvement. It should be noted that these descriptors are more expensive to compute (60 ms) and match (30 ms) than ORB descriptors. These processing times were obtained using the same hardware as in Section 5.2.1.

The results of our localization experiment are summarized in Table 5.2. In each case, we use our RANSAC estimator from Section 6.3. The results in the table are obtained by calculating the median translation and rotation error. Compensating for motion distortion results in a 41.7% reduction in translation error. Compensating for Doppler effects results in a further 67.7% reduction in translation error. Together, compensating for both effects results in a 81.2% reduction in translation error. Note that the scan-to-scan translation error is larger than in the odometry experiment due to the increased difficulty of localizing against a reverse drive. Figure 5.6 depicts a histogram of the localization errors

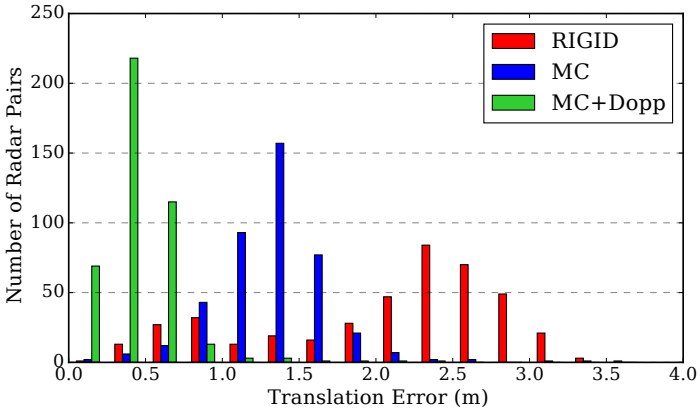


Figure 5.6: By compensating for motion distortion alone (MC) or both motion and Doppler distortion (MC + DOPP), metric localization improves.

Table 5.2: MC-RANSAC Metric Localization Results

Method	Trans. Error (m)	Rot. Error (deg)
No Compensation	2.2976	0.7110
Motion-Compensated	1.3403	1.0766
Motion-Compensated + Doppler-Compensated	0.4327	0.9984

in this experiment.

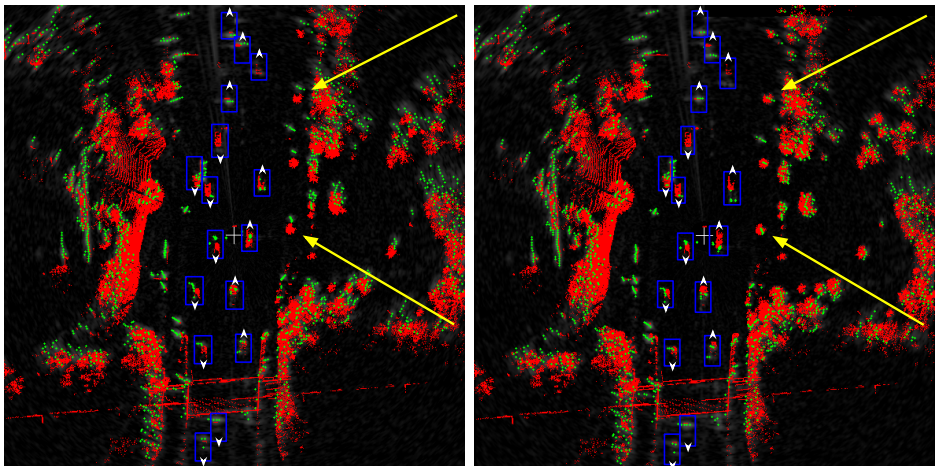
5.2.3 Qualitative Results

In this section, we present qualitative results of removing motion distortion and Doppler effects from radar data. In Figure 5.7, we have plotted points from our Velodyne lidar in red and extracted radar targets in green. In this example, the ego-vehicle is driving at 15 m/s with vehicles approaching from the opposite direction on the left-hand side of the road. To directly compare lidar and radar measurements, we have aligned each sensor's output spatially and temporally using post-processed GPS data and timestamps. Motion distortion is removed from the lidar points before the comparison is made. We use the lidar measurements to visualize the distortion in the radar scan. Figure 5.7(a) shows what the original alignment looks like when the radar scan is distorted. Figure 5.7(b) shows what the alignment looks like after compensating for motion distortion and Doppler effects. Note that static objects such as trees align much better with the lidar data in Figure 5.7(b). It is interesting that some moving vehicles (boxed in blue) are less aligned after removing distortion. Here, we do not know the other vehicles' velocities and, therefore, the true relative velocity with respect to the ego-vehicle. These results indicate that additional velocity information for each object is required to align dynamic objects. We leave this problem for future work.

5.3 Conclusions

In our odometry experiment, compensating for motion distortion had a modest impact of reducing translational drift by 9.4%. Compensating for Doppler effects had a negligible effect on odometry performance. We postulate that Doppler effects are negligible for odometry because their effects are quite similar from one frame to the next. In our localization experiment, we observed that compensating for motion distortion and Doppler effects reduced translation error by 41.7% and 67.7%, respectively, with a combined reduction of 81.2%. We also provided qualitative results demonstrating the impact of both distortion effects. We noted that each dynamic object's velocity is required to properly undistort points associated with dynamic objects. In summary, the Doppler effect can be safely ignored for the radar odometry problem, but motion distortion should be accounted for to achieve the best results. For metric localization, especially for localizing in the opposite direction from which the map was built, both motion distortion and Doppler effects need to be compensated. Accounting for these effects is computationally cheap but requires an accurate estimate of the linear and angular velocity of the sensor.

In our experiments, we observed that the performance is largely limited by the front-end feature detection and matching performance. It is challenging to tune handcrafted features that generalize across all scenes. In the following chapter, we investigate an approach where features are learned in a data-driven fashion using a deep learning front-end. MC-RANSAC also uses a simple motion-compensating estimator. In subsequent chapters, we will employ a more sophisticated estimator, STEAM, based on the work by Anderson and Barfoot [10].



(a) Distorted (b) Motion and Doppler Distortion Removed

Figure 5.7: Lidar points are shown in red, radar targets are shown in green, vehicles are boxed in blue. (a) Both motion distortion and Doppler effects are present in the radar scan. (b) Motion distortion and Doppler effects have been removed from the radar data. Note that static objects (highlighted by yellow arrows) align much better in (b). Some of the moving vehicles (boxed in blue) are less aligned after removing distortion. Here we do not know the other vehicles' velocities and therefore the true relative velocity with respect to the ego-vehicle.

Chapter 6

Combining Probabilistic Estimation and Self-Supervised Feature Learning

Previous approaches to radar odometry have either relied on hand-crafted feature extraction [36, 37, 5, 6, 28, 70, 3, 85], correlative scan matching [114, 20], or a supervised learning algorithm [20, 18] that relies on trajectory groundtruth. Barnes and Posner [18] previously showed that learned features have the potential to outperform hand-crafted features. To address these limitations, we propose a method to learn features directly from radar data without relying on groundtruth pose information.

This chapter presents a self-supervised radar odometry pipeline that approaches the state of the art as reported on the Oxford Radar RobotCar Dataset [19]. Our network parameters are trained using only the onboard radar sensor, alleviating the need for an accurate groundtruth trajectory. To our knowledge, this is the first example of a self-supervised radar odometry pipeline. We show additional experimental results on 100 km of radar data from the Boreas dataset. We also compare radar odometry performance in ideal and harsh weather conditions.

Our approach is based on the Exactly Sparse Gaussian Variational Inference (ESGVI) parameter learning framework of Barfoot et al. [17], a nonlinear batch state estimation framework that provides a family of scalable estimators from a variational objective. Model parameters can be optimized jointly with the state using a data likelihood objective. Yoon et al. [160] applied the framework to train a deep network, demonstrating feature learning for lidar odometry with only onboard lidar data. We extend their methodology and apply it to radar odometry. Our approach is modular, enabling the use of modern deep-learning tools and classical estimators with minimal interface requirements. Importantly, our method does not require the estimator to be differentiable, a limitation of some other learning-based methods [20].

The hybridization of deep learning and probabilistic state estimation allows us to achieve the best of both worlds. Deep learning can be leveraged to process rich sensor data, while classical estimators can be used to deal with probabilities and out-of-distribution samples through outlier rejection schemes, motion priors, and other estimation tools. Furthermore, classical estimators make incorporating additional sensors and constraints relatively straightforward.

6.1 Related Work

Prior works in radar odometry relied on hand-crafted feature detectors and descriptors [36] or cumbersome phase correlation techniques [20]. Barnes and Posner [18] showed that learned features can result in superior radar odometry performance. At the time of publishing, their work represented the state of the art for point-based radar odometry. Despite these results, their approach requires the estimator to be differentiable, and they require groundtruth poses as a supervisory signal. Our approach does not suffer from either of these drawbacks.

Barfoot et al. [17] presented the ESGVI framework and showed that model parameters can be jointly optimized along with the state using an Expectation-Maximization (EM) iterative optimization scheme on a data likelihood objective. In the E-step, model parameters are held fixed, and the state is optimized. In the M-step, the state distribution is held fixed, and the model parameters are optimized. This idea originates from a line of work that applied EM for linear system identification [137, 62]. Ghahramani and Roweis [63] extended the idea to simple nonlinearities approximated with Gaussian radial basis functions. Wong et al. [154] first applied EM under the ESGVI framework to large-scale trajectory estimation by learning noise models robust to outliers. Yoon et al. [160] demonstrated the first application of ESGVI parameter learning to a deep network, where lidar features for odometry were learned without groundtruth. We apply their methodology and adapt it for radar odometry.

Our framework shares similarities with the Variational Autoencoder (VAE) [80] framework, as both start from a data likelihood objective and optimize the Evidence Lower Bound (ELBO). In contrast to the VAE, which approximates latent state inference with a network, our framework applies classic state estimation (e.g., factor graph optimization). Extensions to the VAE for problems with graphical structure exist [76], but our method makes use of existing estimation tools familiar to the robotics field. Most similar to our framework is the work of DeTone et al. [50]. They present a self-supervised visual odometry framework with a deep network frontend trained according to a bundle adjustment backend. Compared to their framework, ours is based on a probabilistic objective and can handle uncertainty in the posterior estimates.

6.2 Exactly Sparse Gaussian Variational Inference Parameter Learning

This section summarizes parameter learning in the ESGVI framework as presented in prior work [17, 154, 160]. The loss function is the negative log-likelihood of the observed data,

$$\mathscr{L} = -\ln p(\mathbf{z}|\boldsymbol{\theta}), \tag{6.1}$$

where $\mathbf{z}$ is the data (e.g., radar measurements), and $\boldsymbol{\theta}$ are the parameters (e.g., network parameters). We apply the usual EM[1] decomposition after introducing the latent trajectory, $\mathbf{x}$, which is written

[1] While the acronym stays the same, we work with the negative log-likelihood and are technically applying Expectation Minimization.

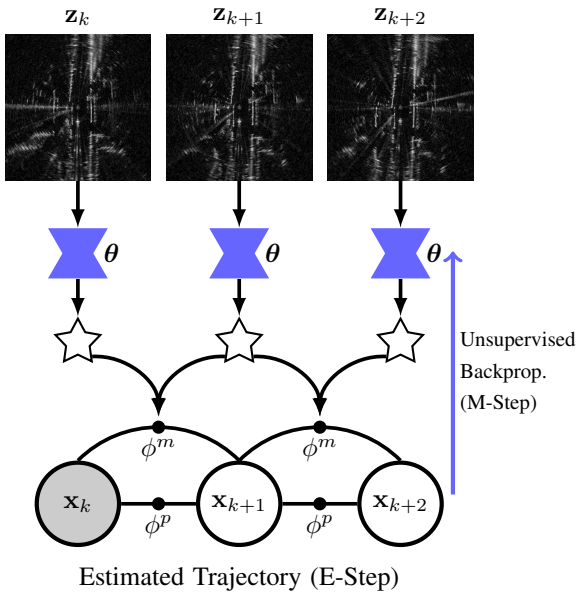


Figure 6.1: This figure depicts the factor graph of our radar odometry pipeline. $\mathbf{x}_k$ and $\mathbf{z}_k$ are defined as the state of the vehicle and the radar scan at time t_k, respectively. The vehicle trajectory is estimated over a sliding window of w frames, where $w = 3$ in this figure. The deep network parameters are denoted by $\boldsymbol{\theta}$. The output of the network is a set of feature locations (x_i, y_i), their associated inverse covariance matrices, $\mathbf{W}_i$, and their learned descriptors, $\mathbf{d}_i$, which are together represented by stars in the diagram. These features are then matched between pairs of frames using a differentiable softmax matcher. The matched features are then used to form measurement factors, ϕ^m. A white-noise-on-acceleration motion prior is applied to create prior factors, ϕ^p.

as

$$\mathscr{L} = \underbrace{\int_{-\infty}^{\infty} q(\mathbf{x}) \ln\left(\frac{p(\mathbf{x}|\mathbf{z},\boldsymbol{\theta})}{q(\mathbf{x})}\right) d\mathbf{x}}_{\leq 0} \underbrace{- \int_{-\infty}^{\infty} q(\mathbf{x}) \ln\left(\frac{p(\mathbf{x},\mathbf{z}|\boldsymbol{\theta})}{q(\mathbf{x})}\right) d\mathbf{x}}_{\text{upper bound}}, \tag{6.2}$$

where we define our posterior approximation as a multivariate Gaussian distribution, $q(\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

We work with the upper bound term, commonly referred to as the (negative) ELBO. Applying the definition of entropy of a Gaussian and dropping constants, we can rewrite the term as the ESGVI loss functional,

$$V(q|\boldsymbol{\theta}) = \mathbb{E}_q[\phi(\mathbf{x},\mathbf{z}|\boldsymbol{\theta})] + \frac{1}{2}\ln\left(|\boldsymbol{\Sigma}^{-1}|\right), \tag{6.3}$$

where $\mathbb{E}[\cdot]$ is the expectation operator, $|\cdot|$ is the matrix determinant, and we define the joint factor $\phi(\mathbf{x},\mathbf{z}|\boldsymbol{\theta}) = -\ln p(\mathbf{x},\mathbf{z}|\boldsymbol{\theta})$.

We apply Generalized EM (GEM) [104], a variation of EM that does not run the M-step to completion (convergence), to gradually optimize the data likelihood, $\mathscr{L}$. In the E-step, we hold $\boldsymbol{\theta}$ fixed and optimize $V(q|\boldsymbol{\theta})$ for $q(\mathbf{x})$. In the M-step, we hold $q(\mathbf{x})$ fixed and optimize $V(q|\boldsymbol{\theta})$ for $\boldsymbol{\theta}$. When the expectation over the posterior, $q(\mathbf{x})$, is approximated at the mean of the Gaussian, the E-step is the familiar Maximum A Posteriori (MAP) estimator [17].

6.3 Self-Supervised Deep Learning for Radar Odometry

6.3.1 Problem Definition

This subsection summarizes the sliding-window odometry formulation presented by Yoon et al. [160]. The state we estimate at time t_k is $\mathbf{x}_k = \{\mathbf{T}_{k,0}, \boldsymbol{\varpi}_k\}$, where the pose $\mathbf{T}_{k,0} \in SE(3)$ is a transformation between frames at t_k and t_0, and $\boldsymbol{\varpi}_k \in \mathbb{R}^6$ is the body-centric velocity[2]. We optimize a sliding window of w frames, $t_\tau, \ldots, t_{\tau+w-1}$, where each state has a corresponding radar scan. The first pose of the window, $\mathbf{T}_{\tau,0}$, is locked (not optimized) and treated as the reference frame for keypoint matching.

Our joint factor, $\phi(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})$, splits into motion prior factors and measurement factors. Figure 6.1 shows an example factor graph illustration. We write $\phi(\mathbf{x}, \mathbf{z}|\boldsymbol{\theta})$ as

$$\sum_{k=\tau+1}^{\tau+w-1} \left(\phi^p(\mathbf{x}_{k-1}, \mathbf{x}_k) + \sum_{\ell=1}^{L_k} \phi^m(\mathbf{z}_k^\ell, \mathbf{r}_\tau^\ell | \mathbf{x}_\tau, \mathbf{x}_k, \boldsymbol{\theta}) \right), \tag{6.4}$$

where $\mathbf{z}_k^\ell$ is the ℓth keypoint measurement in frame k, which has a total of L_k keypoints, and $\mathbf{r}_\tau^\ell$ is its matched point from frame τ. For the motion prior factors, ϕ^p, we apply a white-noise-on-acceleration prior as presented by Anderson and Barfoot [10], which is defined in (3.17). The measurement factors, ϕ^m, are of the form:

$$\phi^m(\mathbf{z}_k^\ell, \mathbf{r}_\tau^\ell | \mathbf{x}_\tau, \mathbf{x}_k, \boldsymbol{\theta}) = \frac{1}{2} {\mathbf{e}_k^\ell}^T \mathbf{W}_k^\ell \mathbf{e}_k^\ell - \ln \left| \mathbf{W}_k^\ell \right|, \tag{6.5}$$

$$\mathbf{e}_k^\ell = \mathbf{D} \left(\mathbf{z}_k^\ell - \mathbf{T}_{k,0} \mathbf{T}_{0,\tau} \mathbf{r}_\tau^{\ell_k} \right), \tag{6.6}$$

where we use the log-likelihood of a Gaussian as the factor, and $\mathbf{D}$ is a 3×4 constant projection matrix that removes the homogeneous element. The homogenous keypoint, $\mathbf{z}_k^\ell$, its point match, $\mathbf{r}_\tau^{\ell_k}$, and its inverse covariance (weight) matrix, $\mathbf{W}_k^\ell$, are quantities that depend on the network parameters, $\boldsymbol{\theta}$.

[2]Despite the radar being a 2D sensor, we formulate our problem in $SE(3)$ to be compatible with other 3D sensor modalities.

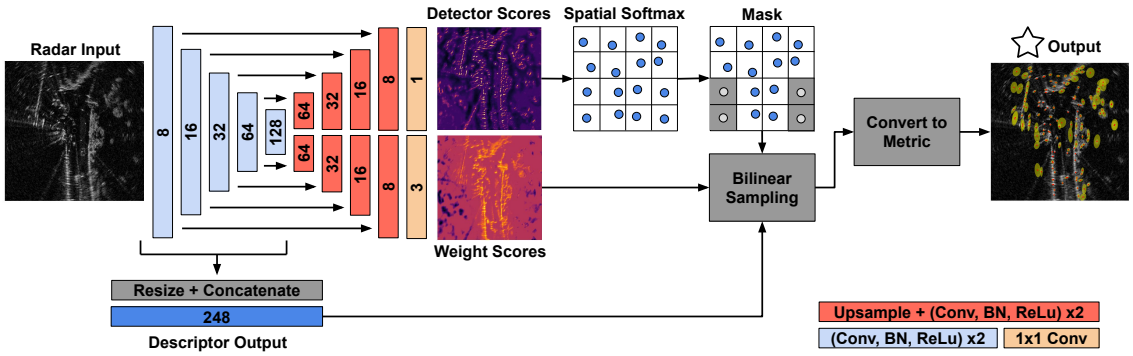


Figure 6.2: We based our network on the architecture presented by Barnes and Posner [18]. The network outputs detector scores for keypoint detection, weight scores predicting keypoint uncertainty, and descriptors for matching. The weight scores are composed into 2×2 inverse covariance matrices (see (6.9), the corresponding image is the log-determinant). 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 is 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.

6.3.2 Network

Our network is based on the architecture presented by Barnes and Posner [18], which is a U-Net [127] style convolutional encoder-multi-decoder architecture to output radar keypoints, weights, and descriptors. The input to the network is a 2D Cartesian projection of the polar radar data. Example radar scans are shown at the top of Figure 6.1.

An illustration of our network architecture is shown in Figure 6.2. A dense descriptor map is created by resizing the output of each encoder block before concatenation into a 248-channel tensor. In our approach, the weight score is a 3-channel tensor, and the detector score is a 1-channel tensor. The detector score tensor is then partitioned into ($N = 400$) equally sized cells, with each producing a candidate 2D keypoint. A spatial softmax within each cell, followed by a weighted summation of pixel coordinates, produces the image-space keypoint coordinates of each cell. The corresponding descriptor and weight score vectors are bilinearly sampled using the keypoint coordinates. The image coordinates are converted to metric coordinates using the known m/pixel resolution. Finally, we formulate the 3D homogenous keypoint, $\mathbf{z}_k^\ell$, by appending a 0 as a third coordinate and a 1 as the fourth homogenous element.

Our detector produces a candidate keypoint for each square cell in a uniformly partitioned radar image; this leads to candidate keypoints in image regions void of data (i.e., black regions of a radar image). As we are training without groundtruth, we found it necessary to mask (reject) these keypoints in practice. We threshold each azimuth of the polar radar scan by a scalar multiple, $\beta = 3$, of its mean intensity, where exceeding the threshold is considered valid. Projecting the result

into Cartesian space produces a binary mask of valid pixels. We then threshold on the ratio of valid pixels in each square cell for robustness to noise. Keypoints belonging to cells with less than 5% valid pixels are rejected.

A dense match between each keypoint descriptor and the reference descriptor tensor is applied with a softmax to preserve differentiability. We compute the dot product between each keypoint descriptor and all descriptors of the reference:

$$\mathbf{c}_k^{\ell^T} = \mathbf{d}_k^{\ell\,T} \begin{bmatrix} \mathbf{d}_\tau^1 & \cdots & \mathbf{d}_\tau^N \end{bmatrix}, \tag{6.7}$$

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

$$\mathbf{r}_\tau^{\ell_k} = \begin{bmatrix} \mathbf{p}_\tau^1 & \cdots & \mathbf{p}_\tau^N \end{bmatrix} \times \text{ softmax}(T\mathbf{c}_k^\ell), \tag{6.8}$$

where $T = 100$ is a softmax temperature constant, $\mathbf{p}_\tau^1, \ldots, \mathbf{p}_\tau^N$ are the homogeneous reference coordinates, and $\mathbf{p}_\tau^n \in \mathbb{R}^4$.

The weight score vectors of each keypoint are assembled into matrices with the following decomposition [92, 160]:

$$\mathbf{W}_k^\ell = \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0}^T & c \end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix} 1 & 0 \\ d_3 & 1 \end{bmatrix} \begin{bmatrix} \exp d_1 & 0 \\ 0 & \exp d_2 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ d_3 & 1 \end{bmatrix}^T, \tag{6.9}$$

where (d_1, d_2, d_3) is the weight score vector corresponding to keypoint $\mathbf{z}_k^\ell$, and $c = 10^4$ is a constant corresponding to the (inverse) variance of the third coordinate (which will always be 0). Visualizations of the learned keypoint covariances, $\mathbf{R}^{-1}$, are shown in Figure 6.3 as uncertainty ellipses.

6.3.3 Training and Inference

We follow the training methodology of Yoon et al. [160], which blends the GEM iterative optimization scheme required by ESGVI with Stochastic Gradient Descent (SGD) applied in conventional network training. The E-step, which optimizes for the current best posterior estimate, $q(\mathbf{x})$, is simply included in the forward propagation routine. Windows of radar scans are randomly sampled as a mini-batch of data. Forward propagation is summarized as:

1. Traditional forward propagation of the network to output radar features (see Section 6.3.2).
2. Construct the motion prior factors, ϕ^p, and measurement factors, ϕ^m, (see Section 6.3.1 and 6.3.2).
3. Batch inference for the current best posterior estimate $q(\mathbf{x})$ for each window (E-step).

We approximate the E-step with the Gauss-Newton algorithm, which involves approximations to the Hessian and approximating the expectation in (6.3) at only the mean of the posterior.

The M-step is network backpropagation on the loss functional (6.3), which only applies to the measurement factors since the motion prior factors are constant with respect to the parameters, $\boldsymbol{\theta}$. Similar to the E-step, we approximate the expectation at only the mean of the posterior. Intuitively, we are using our best-guess trajectory as a supervisory signal to then carry out standard SGD to train the feature network.

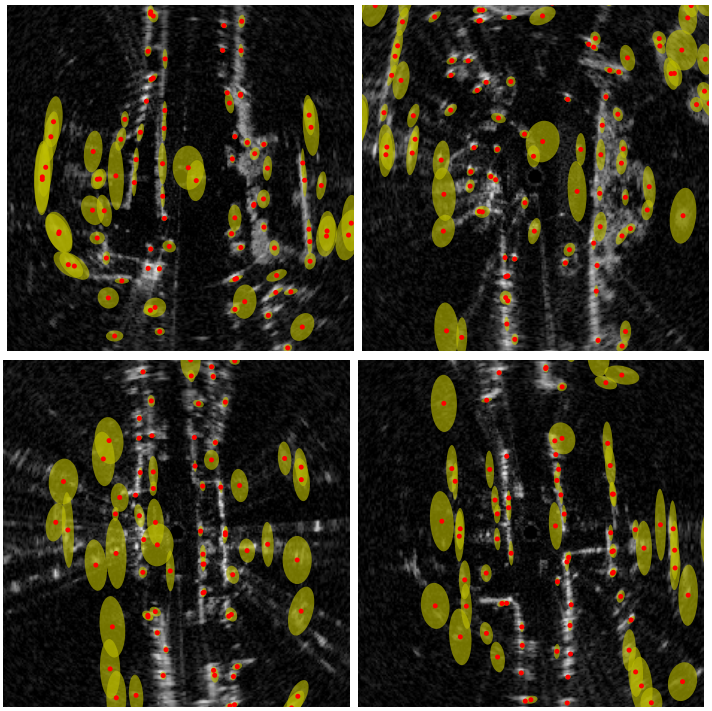


Figure 6.3: A visualization of keypoints (red) on radar images with 5 standard deviation uncertainty ellipses (yellow). Many keypoints have elongated uncertainties consistent to the scene geometry, and as expected, keypoints further away from the sensor (image centre) are more uncertain.

6.3.4 Outlier Rejection

We apply the Geman-McClure robust cost function on the measurement factors, ϕ^m, in the E-step for outlier rejection. For the M-step, we follow Yoon et al. [160] and apply a constant threshold, α, on the squared Mahalanobis distance of the measurement factor with the current best posterior estimate,

$$\mathbf{e}_k^{\ell\,T}\mathbf{W}_k^\ell\mathbf{e}_k^\ell > \alpha, \tag{6.10}$$

where $\mathbf{e}_k^\ell$ is as defined in (6.6). We do not backpropagate keypoint matches greater than $\alpha = 16$. After training, we improved odometry performance by rejecting keypoints with inverse covariances that have a log determinant, $\log|\mathbf{R}|$, less than a threshold ($\eta = 4.0$). We also use RANSAC at test time and only produce an estimate based on the inliers.

6.4 Experimental Results

6.4.1 Experiment Setup

We evaluate our approach, Hybrid-Estimate Radar Odometry (HERO), on the publicly available Oxford Radar RobotCar Dataset [19] and on the Boreas dataset. The Oxford dataset is divided into 32 sequences, each approximately 10 km long. We follow existing work by training on 24 sequences, validating on 1 sequence, and testing on the same 7 sequences as [18].

Our implementation is a hybrid between Python and C++, where network-related code is imple-

Table 6.1: Radar Odometry Results. (HC): Hand-crafted, (L): Learned.

Methods	Supervision	Translational Error (%)	Rotational Error (deg/1000m)
UnderTheRadar [18]	Supervised (L)	2.0583	6.7
Cen RO [36]	Unsupervised (HC)	3.7168	9.5
MC-RANSAC [28]	Unsupervised (HC)	3.3190	10.93
CFEAR [3]	Unsupervised (HC)	**1.76**	**5.0**
HERO (Ours)	Self-Supervised (L)	1.9879	6.524

mented in Python using PyTorch [116], and estimation-related code is implemented using STEAM[3], an open-source C++ estimation library. When training the network, we use a fixed random seed and the Adam optimizer [81] with a learning rate of 1×10^{-5} and a mini-batch size of 1 window ($w = 4$) for up to 100k iterations. The Navtech radar sensor used in the Oxford dataset has a range resolution of 0.0438 m/bin. We chose a Cartesian resolution of 0.2628 m/pixel to minimize projection errors as an integer multiple of the radar resolution. The Cartesian radar images are made to be square with a width of 640 pixels. For the spatial softmax operation, each cell is 32×32 pixels, resulting in 400 total keypoints. We use a softmax temperature of 100. For the motion prior, $\mathbf{Q}_c$ is made to be diagonal. The entries of $\mathbf{Q}_c^{-1}$ can be considered penalty terms on body-centric linear and angular acceleration. It is possible to learn $\mathbf{Q}_c$ as is done [154]. However, in our implementation, the values are hand-tuned. Similar to Barnes and Posner [18], we augment our training data with random rotations of up to 0.26 radians. Our sliding-window implementation[4] takes on average 0.07 seconds, 0.13 seconds, and 0.18 seconds for window sizes of 2, 3, and 4, respectively. Since the radar sensor spins at 4 Hz, our current implementation is real-time capable. During evaluation, we use the timestamp of each measurement to do continuous-time estimation with STEAM to compensate for motion distortion. For more information on this approach, we refer readers to the work of Anderson et al. [10] [11].

6.4.2 Oxford Radar RobotCar Dataset

Here, we compare our radar odometry results with other point-based radar odometry methods, including Cen and Newman [36] (Cen RO), Barnes and Posner [18] (Under the Radar), CFEAR [3], and Burnett et al. [28] (MC-RANSAC). Following these previous works, we report our results using the KITTI odometry metrics [60], which average the relative position and orientation errors over every sub-sequence of length (100 m, 200 m, $\cdots$, 800 m). The results in Table 6.1 show that our method, HERO, is competitive with other unsupervised radar odometry methods, surpassing the hand-crafted algorithms Cen RO and MC-RANSAC. In addition, our method exceeds the performance of Under the Radar, a learned point-based radar odometry approach, without needing any groundtruth supervision. This feature gives our method an advantage for deployment in regions where a source of high-quality groundtruth is unavailable. Sources of groundtruth, such as a GNSS/IMU system, are costly and require a clear line of sight to the sky for GPS reception. Notably, the authors of the Oxford Radar RobotCar Dataset [18] state that the accuracy of their

[3] `https://github.com/utiasASRL/steam`
[4] On an Nvidia Tesla V100 GPU and 2.2 GHz Intel Xeon CPU.

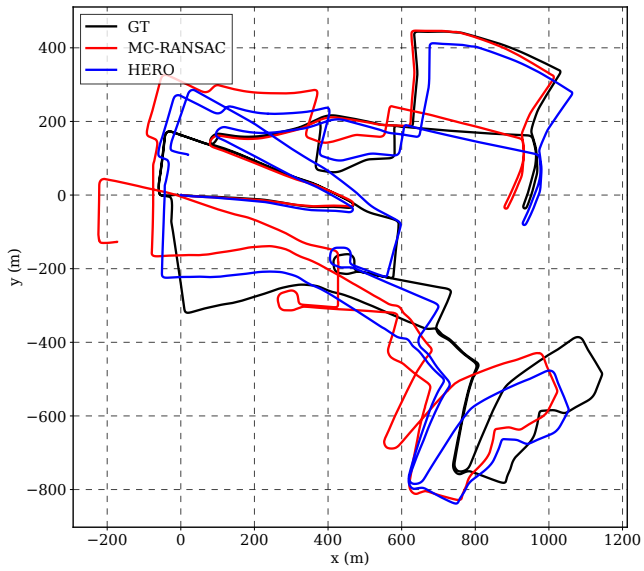


Figure 6.4: This figure depicts the results of our self-supervised Hybrid-Estimate Radar Odometry (HERO) in blue alongside the results of a hand-crafted radar odometry pipeline based on motion compensated RANSAC (MC-RANSAC) [28] in red. The groundtruth trajectory is plotted in black (GT). Sequence: 2019-01-10-14-02-34-radar-oxford-10k.

GPS/INS system varied significantly due to poor GPS reception [97]. They addressed this issue by including visual odometry and loop closures in a large-scale optimization [18]. Figure 6.4 illustrates an example sequence comparing our odometry method to MC-RANSAC.

We provide an ablation study of our method in Table 6.2. We first compare our baseline to the following variations:

- **Scalar Weight**: Instead of learning a 2×2 weight (inverse covariance) matrix (see (6.9)), we learn a scalar weight.
- **No Mah. Threshold**: We do not apply the outlier rejection threshold on the squared Mahalanobis distance 6.10.
- **No Masking**: We do not mask keypoints in void regions of the radar images, as described in Section 6.3.2.
- **No Augmentation**: We do not augment our training data with random rotations.

We also show the effect of varying the window size of our sliding window optimization and the resolution of the projected radar images. Our baseline uses a window size of 4 and a resolution of 0.2628 m/pixel.

The results in Table 6.2 suggest that rotation augmentation, masking keypoints, and thresholding on Mahalanobis distance were the most significant hyperparameters. We theorize that because a large fraction of the Oxford dataset consists of driving straight or waiting at a red light, training without rotation augmentation makes it less likely that the network will learn to match features across large frame-to-frame rotations properly. Putting a hard threshold on the Mahalanobis distance before backpropagation allows us to only backpropagate the errors from likely inliers. The masking

Table 6.2: Ablation study. Results for sequence 2019-01-10-14-02-34-radar-oxford-10k.

Configuration	**Translational Error (%)**	**Rotational Error (deg/1000m)**
Baseline	2.262	7.601
Scalar Weight	2.484	7.997
No Mah. Threshold	3.083	9.300
No Masking	3.203	10.07
No Augmentation	5.054	16.60
Window Size 2	2.488	7.901
Window Size 3	2.393	7.657
Cart Res: 0.2160	2.396	7.602
Cart Res: 0.3024	2.407	7.927
Baseline + ($\log\|\mathbf{R}\| < 4.0$)	**1.953**	**6.532**

of keypoints from empty regions of the input radar scan was necessary to achieve good odometry results. We believe adding this additional structure to the learning problem is needed for the self-supervised model to succeed.

6.4.3 Additional Experiments on the Boreas Dataset

In this section, we provide additional experimental results for our radar odometry using 100 km of driving from the Boreas dataset. See Chapter 4 for more details on the Boreas dataset. We do not require groundtruth position data to train our network, but obtain it for the test sequences to calculate KITTI odometry drift metrics. Groundtruth positioning is obtained from the post-processed GNSS results and is accurate to within 2-4 cm.

The 100 km of total driving data is divided into 11 sequences, each approximately 9 km in length. We use 7 sequences for training, 1 for validation, and 3 for testing. Two of the 3 test sequences were taken during a snow storm, and the other was taken on a sunny day.

Figure 1.1 illustrates the large differences in weather conditions. Between the sunny and snowy days, there were significant visual appearance changes. Most of the lane lines are not visible during the snow storm. The contrast between the lidar data taken on the two days (after the ground plane is removed) is even more stark. During the snow storm, the lidar scan is littered with many detections associated with snow flakes. Although we do not provide experimental results for this, it seems probable that the accuracy of a lidar odometry system would suffer under these conditions. However, Charron et al. [40] have shown that these effects can be compensated. Unsurprisingly, it is difficult to discern the difference between the radar scans other than the movement of large vehicles.

The odometry drift results reported in Table 6.3 exemplify the robustness of radar to inclement weather. The drift rates on the snowy days were lower than on the sunny day. We postulate that the error increases on a sunny day due to increased vehicle speed. It is interesting to note that our rotation drift metrics here are lower than on the Oxford dataset. While it is difficult to point to a single factor as the reason for this difference in performance, we note that our trajectory involves fewer turns, and we use a different source of groundtruth for testing. In Figure 6.5, we provide a plot of our odometry results during a snowy sequence. Here, we have plotted the results of training on the Oxford dataset and testing on this sequence, as well as training on the Boreas dataset and

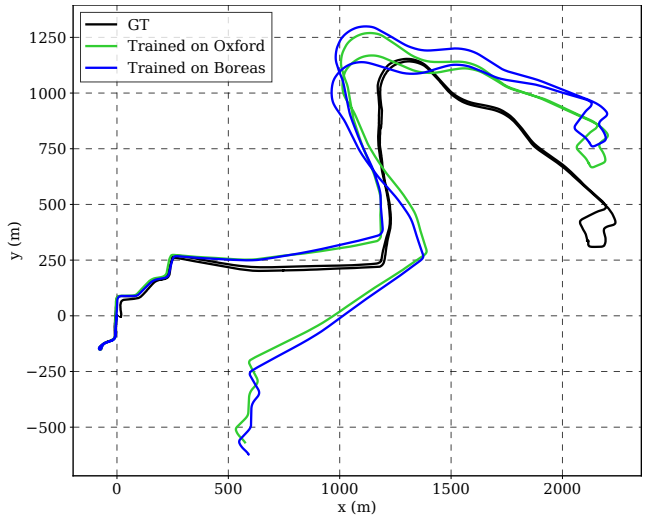


Figure 6.5: Odometry results on the snowy Boreas sequence 2021-01-26-11-22.

Table 6.3: Boreas test results.

Sequence	Weather	Trained On	Translational Error (%)	Rotational Error (deg/1000m)
01-26-10-59	Snow	Boreas	2.003	5.599
01-26-11-22	Snow	Boreas	1.980	5.288
02-09-12-55	Sun	Boreas	2.073	5.886
01-26-10-59	Snow	Oxford	2.355	8.137
01-26-11-22	Snow	Oxford	2.112	6.442
02-09-12-55	Sun	Oxford	2.546	8.911

testing on this sequence. There is a noticeable increase in the drift rate when transferring from the Oxford dataset to this Boreas dataset. We hypothesize that this difference is mainly due to the sensors' different range resolutions.

6.5 Conclusions

In this chapter, we applied the ESGVI parameter learning framework to learn radar features for odometry using only the onboard radar data. Our odometry performance on the Oxford Radar RobotCar Dataset approaches the current state of the art, which is a hand-crafted method [3]. We provided additional experimental results on 100 km of driving in an urban setting. Within this dataset, we demonstrated the effectiveness of radar odometry during heavy snowfall.

In recent years, robotics research has become increasingly reliant on high-quality datasets for training deep learning algorithms. The size and scope of these datasets is a limiting factor in the performance of many robotic systems. However, collecting and annotating these datasets is an expensive and labour-intensive process. For this reason, it is important that research in robotics trends towards solutions that are data efficient. By foregoing the need for groundtruth, our self-supervised architecture enables large quantities of training data to be collected with relative ease.

Our framework is a hybrid of modern deep learning and classic probabilistic state estimation. Deep learning can be used to process rich sensor data, while probabilistic estimation can be used to handle out-of-distribution samples through outlier rejection schemes. Furthermore, our modular framework enables many future extensions and avenues of research. Now that we have a highly performant radar odometry approach, the next logical step is to build a radar-based mapping and localization framework to enable robots to navigate autonomously. It is also important to consider whether our proposed radar-based localization is useful, given that lidar-based localization is readily available and has been shown to work well under nominal weather conditions.

Chapter 7

Are We Ready for Radar to Replace Lidar?

Many autonomous driving companies leverage detailed semantic maps to drive safely. These maps may include the locations of lanes, pedestrian crossings, traffic lights, and more. In this case, the vehicle no longer has to detect each of these features from scratch in real time. Instead, given the vehicle's current position, the semantic map can be used as a prior to simplify the perception task. However, it then becomes critical to know the robot's pose within the map with sufficient accuracy and reliability.

Dense lidar maps can be built using offline batch optimization while incorporating IMU measurements for improved local alignment and GPS for improved global alignment [90]. Highly accurate localization can be subsequently performed by aligning a live lidar scan with a pre-built map with reasonable robustness to weather conditions [152, 21]. Vision-based mapping and localization is an alternative that can be advantageous in the absence of environmental geometry. However, robustness to large appearance change (e.g., lighting) is a difficult and ongoing research problem [64]. Radar-based systems present another compelling alternative.

Models of atmospheric attenuation show that radar can operate under certain adverse weather conditions where lidar cannot. These conditions may include heavy rain (>25mm/hr), dense fog (>0.1g/m^3), or a dust cloud (>10g/m^3) [25, 24]. Existing literature does not describe the operational envelope of current lidar or radar sensors for localization. Prior works have assumed that lidar localization is susceptible to moderate rain or snow, necessitating the use of radar. In this chapter, we attempt to shed some light on this topic by comparing the performance of three topometric localization systems: radar-only, lidar-only, and a cross-modal radar-to-lidar system. We compare these systems across varying seasonal and weather conditions using the Boreas dataset, as described in Chapter 4. Such a comparison of topometric localization methods has not been shown in the literature before and forms our primary contribution.

Prior work has focused on localizing radar scans to satellite imagery [146, 144, 145] or to pre-built lidar maps [158, 159]. Localizing live radar scans to existing lidar maps built in ideal conditions is a desirable option, as we still benefit from the robustness of radar without incurring the expense of building brand-new maps. However, the global localization errors reported in these works are in

[2] `https://youtu.be/Cay6rSzeo1E/`

Figure 7.1: The 8km Glen Shields route[2] in Toronto. The yellow stars correspond to UTIAS, Dufferin, and Glen Shields (left to right) as in Figure 7.4.

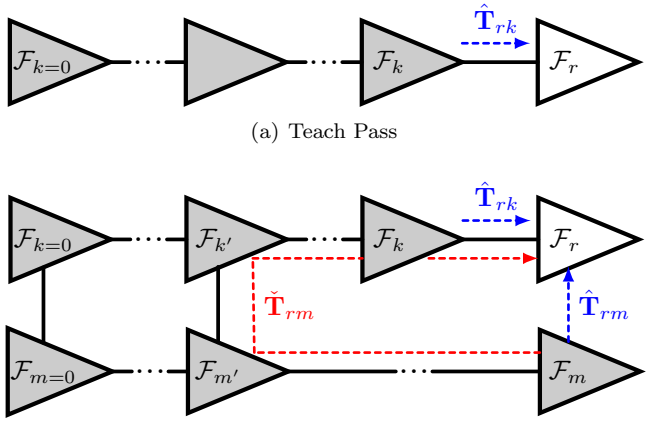
the range of 1m or greater. We demonstrate that we can successfully localize live radar scans to a pre-built lidar map with a relative localization error of around 0.1m.

In this chapter, we implement topometric localization that follows the Teach and Repeat paradigm [56, 117] without using GPS or IMU measurements. Hong et al. [71] recently compared the performance of their radar SLAM to SuMa, surfel-based lidar SLAM [21]. On the Oxford RobotCar dataset [19], they show that SuMa outperforms their radar SLAM. However, in their experiments, SuMa often fails partway through a route. Our interpretation is that SuMa losing track is likely due to an implementation detail inherent to SuMa itself rather than a shortcoming of all lidar-based SLAM systems. Importantly, our results seem to conflict with theirs by showing that lidar localization can operate successfully in even moderate to heavy snowfall. However, topometric localization may be more robust to adverse weather since it uses both odometry and localization to a pre-built map. In addition, we have a 128-beam lidar, which is more dense than the lidar used in their experiments.

7.1 Methodology

7.1.1 Lidar/Radar Teach and Repeat Overview

Teach and Repeat is an autonomous route following framework that manually teaches a robot a network of traversable paths [56, 118, 83]. A key enabling idea is the construction of a *topometric map* [14] of the taught paths, represented as a pose graph in Figure 7.2. In the teach pass, a sequence of sensor data (i.e., lidar or radar) from a driven route is processed into local submaps stored along the path (vertices) and are connected by relative pose estimates (edges). In the repeat pass, a new sequence of sensor data following the same route is processed into a new branch of the pose graph while simultaneously being localized against the vertices of the previous sequence to account for odometric drift. By localizing against local submaps along the taught paths, the robot can accurately localize and route-follow without needing an accurate global reconstruction. In this chapter, we focus on the estimation pipeline of Teach and Repeat (Figure 7.3). We divide the pipeline into: *Preprocessing*, *Odometry and Mapping*, and *Localization*



(a) Teach Pass

(b) Repeat Pass

Figure 7.2: The structure of the pose graph during (a) the teach pass and (b) the repeat pass. $\mathcal{F}_r$ is the moving robot frame, and others are vertex frames. We use subscript k for vertex frames from the current pass (teach or repeat) and m for vertex frames from the reference pass (always teach). During both teach and repeat passes, we estimate the transformation from the latest vertex frame $\mathcal{F}_k$ to the robot frame $\mathcal{F}_r$, $\hat{\mathbf{T}}_{rk}$, using odometry. For repeat passes only, we define $\mathcal{F}_{k'}$ to be the latest vertex frame that has been successfully localized to the reference pass, $\mathcal{F}_{m'}$ the corresponding map vertex of $\mathcal{F}_{k'}$, and $\mathcal{F}_m$ the spatially closest map vertex to $\mathcal{F}_r$. A prior estimate of the transform from $\mathcal{F}_m$ to $\mathcal{F}_r$, $\check{\mathbf{T}}_{rm}$, is generated by compounding transformations through $\mathcal{F}_{m'}$, $\mathcal{F}_{k'}$, and $\mathcal{F}_k$, which is then used to compute the posterior $\hat{\mathbf{T}}_{rm}$.

Preprocessing

This module performs feature extraction and filtering on raw sensor data, which in our work is from either lidar or radar sensors. More sensor-specific information is provided in 7.1.2.

Odometry and Mapping

During both teach and repeat passes, this module estimates the transformation between the submap of the latest (local) vertex frame, $\mathcal{F}_k$, and the latest live sensor scan at the current moving robot frame, $\mathcal{F}_r$ (i.e., $\hat{\mathbf{T}}_{rk}$ in Figure 7.2(a)). If the translation or rotation of $\hat{\mathbf{T}}_{rk}$ exceeds a predefined threshold (10m / 30 degrees), we add a new vertex $\mathcal{F}_{k+1}$ connected with a new edge $\mathbf{T}_{k+1,k} = \hat{\mathbf{T}}_{rk}$. Each edge consists of the mean relative pose and its covariance (uncertainty) estimate. The new submap stored at $\mathcal{F}_{k+1}$ is an accumulation of the last $n = 3$ processed sensor scans. All submaps are motion-compensated and stored in their respective (local) vertex frame. The live scan is also motion compensated and sent as input to the *Localization* module. We present the details of our motion-compensated odometry algorithm in Section 7.1.3.

Localization

During the repeat pass, this module localizes the motion-compensated live scan of the robot frame, $\mathcal{F}_r$, against the submap of the spatially closest vertex frame, $\mathcal{F}_m$, of the previous sequence (i.e., $\hat{\mathbf{T}}_{rm}$ as shown in Figure 7.2(b)). Vertex frame $\mathcal{F}_m$ is chosen by leveraging our latest odometry estimate and traversing the pose graph edges. Given the pose graph in Figure 7.2(b), the initial estimate to

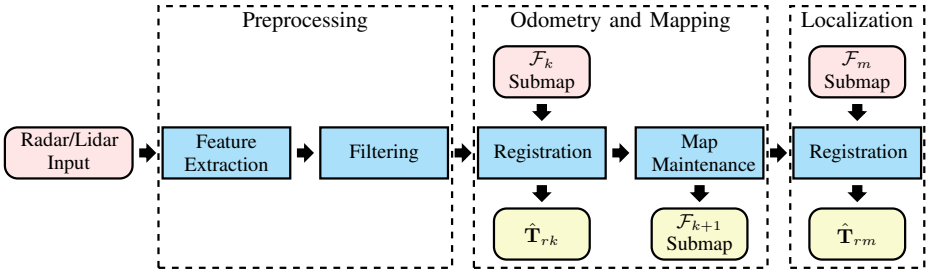


Figure 7.3: The data processing pipeline of our Teach and Repeat implementation, divided into three modules: *Preprocessing*, *Odometry and Mapping*, and *Localization*. See 7.1.1 for a detailed description of each module.

$\mathcal{F}_m$ is

$$\check{\mathbf{T}}_{rm} = \mathbf{T}_{rk}\mathbf{T}_{kk'}\mathbf{T}_{k'm'}\mathbf{T}_{m'm}. \tag{7.1}$$

We localize using ICP with $\check{\mathbf{T}}_{rm}$ as a prior, resulting in $\hat{\mathbf{T}}_{rm}$. If ICP alignment is successful, we add a new edge between the vertex of $\mathcal{F}_m$ and the latest vertex of the current sequence, $\mathcal{F}_k$, by compounding the mean localization result with the latest odometry result,

$$\hat{\mathbf{T}}_{km} = \hat{\mathbf{T}}_{rk}^{-1}\hat{\mathbf{T}}_{rm}, \tag{7.2}$$

as well as their covariances. We present the details of the ICP optimization in Section 7.1.4.

7.1.2 Raw Data Preprocessing

Lidar

We first perform voxel downsampling for each incoming lidar scan with voxel size $dl = 0.3$m. Only one point closest to the voxel centre is kept. Next, we extract plane features from the downsampled pointcloud by applying Principle Component Analysis (PCA) to each point and its neighbors from the raw scan. We define a feature score from PCA to be

$$s = 1 - \lambda_{\min}/\lambda_{\max}, \tag{7.3}$$

where $\lambda_{\min}$ and $\lambda_{\max}$ are the minimum and maximum eigenvalues, respectively. The downsampled pointcloud is filtered by this score, keeping no more than 20,000 points with scores above 0.95. We associate each point with its eigenvector of $\lambda_{\min}$ from PCA as the underlying normal.

Radar

We first extract point targets from each azimuth for each radar scan using the Bounded False Alarm Rate (BFAR) detector as described in [7]. BFAR adds a constant offset b to the usual Cell-Averaging CFAR threshold: $T = a \cdot Z + b$. We use the same (a, b) parameters as [7]. For each azimuth, we also perform *peak detection* by calculating the centroid of contiguous groups of detections as is done in [36]. We obtained a modest performance improvement by retaining the maximum of the left and right sub-windows relative to the cell under test as in (greatest-of) GO-CFAR [125]. These polar targets are then transformed into Cartesian coordinates and are passed to the Odometry and

Mapping module without further filtering.

7.1.3 VTR3 Continuous-Time Odometry

Our odometry algorithm combines the iterative data association of ICP with a continuous-time trajectory represented as exactly sparse Gaussian Process regression [10]. Our trajectory is $\mathbf{x}(t) = \{\mathbf{T}(t), \boldsymbol{\varpi}(t)\}$, where $\mathbf{T}(t) \in SE(3)$ is our robot pose and $\boldsymbol{\varpi}(t) \in \mathbb{R}^6$ is the body-centric velocity. Following Anderson and Barfoot [10], the GP motion prior is defined in (3.17).

The prior (3.17) is applied in a piecewise fashion between an underlying discrete trajectory of pose-velocity state pairs, $\mathbf{x}_i = \{\mathbf{T}_i, \boldsymbol{\varpi}_i\}$, that each correspond to the representative timestamp of the ith sensor scan. Each pose, $\mathbf{T}_i$, is the relative transform from the latest vertex, $\mathcal{F}_k$, to the robot frame, $\mathcal{F}_r$, that corresponds to the ith sensor scan (i.e., $\mathbf{T}_{rk}$). Likewise, $\boldsymbol{\varpi}_i$ is the corresponding body-centric velocity. We seek to align the latest sensor scan i to the latest vertex submap in frame $\mathcal{F}_k$ (see Figure 7.2).

We define a nonlinear optimization for the latest state $\mathbf{x}_i$, locking all previous states. The cost function is

$$J_{\text{odom}} = \phi_{\text{motion}} + \underbrace{\sum_{j=1}^{M} \left(\frac{1}{2} \mathbf{e}_{\text{odom},j}^T \mathbf{R}_j^{-1} \mathbf{e}_{\text{odom},j} \right)}_{\text{measurements}}. \tag{7.4}$$

In the interest of space, we refer readers to Anderson and Barfoot [10] for the squared-error cost expression of the motion prior, ϕ_{motion}. Each measurement error term is

$$\mathbf{e}_{\text{odom},j} = \mathbf{D} \left(\mathbf{p}_k^j - \mathbf{T}(t_j)^{-1} \mathbf{T}_{rs} \mathbf{q}_j \right), \tag{7.5}$$

where $\mathbf{q}_j$ is a homogeneous point with corresponding timestamp t_j from the ith sensor scan, $\mathbf{T}_{rs}$ is the extrinsic calibration from the sensor frame $\mathcal{F}_s$ to the robot frame $\mathcal{F}_r$, $\mathbf{T}(t_j)$ is a pose from our trajectory queried at t_j[3], $\mathbf{p}_k^j$ is a homogeneous point from the kth submap associated to $\mathbf{q}_j$ and expressed in $\mathcal{F}_k$, and $\mathbf{D}$ is a constant projection that removes the 4th homogeneous element. We define $\mathbf{R}_j^{-1}$ as either a constant diagonal matrix for radar data (point-to-point) or by using the outer product of the corresponding surface normal estimate for lidar data (point-to-plane).

We optimize for $\mathbf{x}_i = \{\mathbf{T}_i, \boldsymbol{\varpi}_i\}$ iteratively using Gauss-Newton, but with nearest-neighbours data association after every Gauss-Newton iteration. VTR3 odometry is therefore performed with the following steps:

1. Temporarily transform all points q_j to frame $\mathcal{F}_k$ using the latest trajectory estimate (motion undistortion).

2. Associate each point to its nearest neighbour in the map to identify its corresponding map point p_k^j in $\mathcal{F}_k$.

3. Formulate the cost function J in (7.4) and perform a single Gauss-Newton iteration to update $\mathbf{T}_i$ and $\boldsymbol{\varpi}_i$.

4. Repeat steps 1 to 3 until convergence.

[3]Through interpolation, $\mathbf{T}(t_j)$ depends on the state variables $\mathbf{x}_i = \{\mathbf{T}_i, \boldsymbol{\varpi}_i\}$ and $\mathbf{x}_{i-1} = \{\mathbf{T}_{i-1}, \boldsymbol{\varpi}_{i-1}\}$ since $t_j > t_{i-1}$ [10].

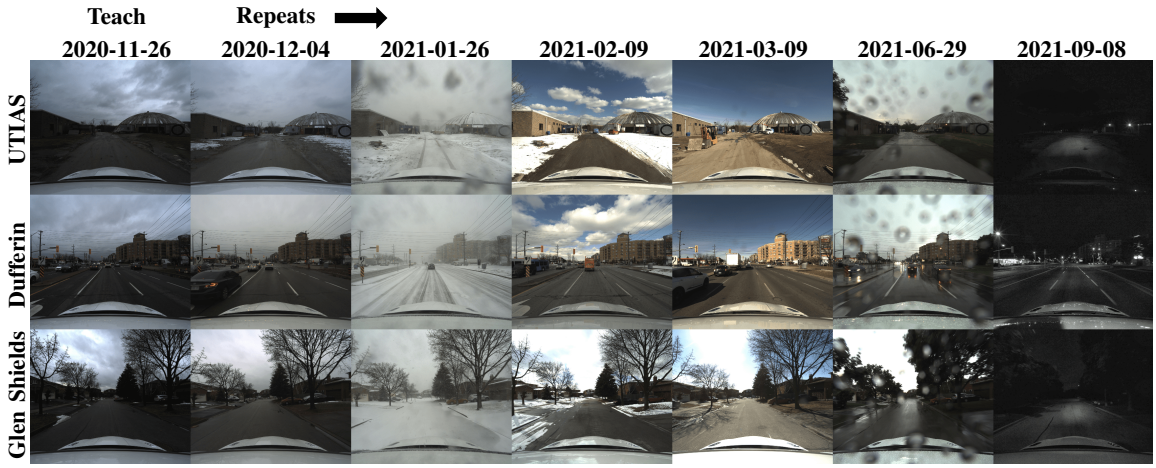


Figure 7.4: Our test sequences were collected by driving a repeated route over the source of one year. Our experiments use 2020-11-26 as our reference sequence for building maps. The remaining six sequences, which include sequences with rain and snow, are used to benchmark localization performance, which amounts to 48km of test driving. These camera images are provided for context.

The output of VTR3 at the timestamp of the latest sensor scan is then the odometry output $\hat{\mathbf{T}}_{r,k}$.

7.1.4 Localization ICP

We use ICP to localize the motion-compensated live scan of the robot frame, $\mathcal{F}_r$, against the submap of the spatially closest vertex frame, $\mathcal{F}_m$, of the previous sequence. The resulting relative transformation is $\hat{\mathbf{T}}_{rm}$, as shown in Figure 7.2(b). The nonlinear cost function is

$$J_{\text{loc}} = \phi_{\text{pose}} + \sum_{j=1}^{M} \left(\frac{1}{2} \mathbf{e}_{\text{loc},j}^T \mathbf{R}_j^{-1} \mathbf{e}_{\text{loc},j} \right), \tag{7.6}$$

where we use $\check{\mathbf{T}}_{rm}$ from (7.1) as an initial guess and a prior:

$$\phi_{\text{pose}} = \frac{1}{2} \ln(\check{\mathbf{T}}_{rm} \mathbf{T}_{rm}^{-1})^{\vee T} \mathbf{Q}_{rm}^{-1} \ln(\check{\mathbf{T}}_{rm} \mathbf{T}_{rm}^{-1})^{\vee}, \tag{7.7}$$

where $\ln(\cdot)$ is the logarithm map and the operator $\vee$ is the inverse of $\wedge$ [16]. The covariance $\mathbf{Q}_{rm}$ can be computed by compounding the edge covariances corresponding to the relative transformations in (7.1). Since all pointclouds are already motion-compensated, the measurement error term is simply

$$\mathbf{e}_{\text{loc},j} = \mathbf{D} \left(\mathbf{p}_m^j - \mathbf{T}_{rm}^{-1} \mathbf{T}_{rs} \mathbf{q}_j \right), \tag{7.8}$$

where $\mathbf{q}_j$ is a homogeneous point from the motion-compensated live scan, and $\mathbf{p}_m^j$ is the nearest neighbour submap point to $\mathbf{q}_j$. See Section 7.1.3 for how we define $\mathbf{T}_{rs}$, $\mathbf{D}$, and $\mathbf{R}_j^{-1}$.

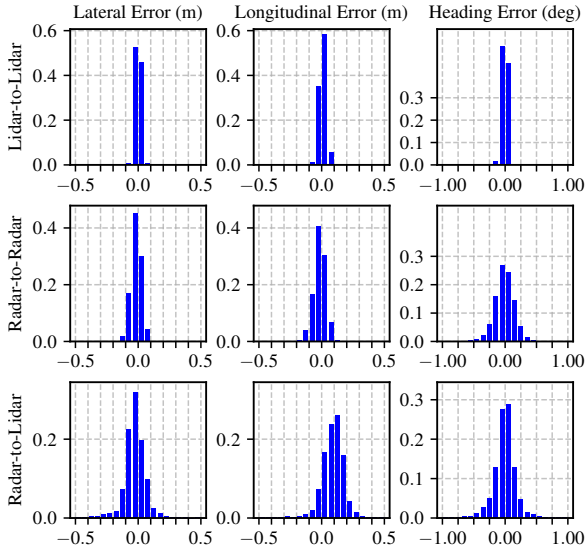


Figure 7.5: These histograms show the spread of the localization error for lidar-to-lidar, radar-to-radar, and radar-to-lidar, during a snowstorm (2021-01-26).

7.1.5 Doppler-Compensated ICP

In our prior work, [28], we showed that current Navtech radar sensors are susceptible to Doppler distortion and that this effect becomes significant during mapping and localization. A relative velocity between the sensor and the surrounding environment causes the received frequency to be altered according to the Doppler effect. If the velocity in the sensor frame is known, this effect can be compensated for using a simple additive correction factor $\Delta\mathbf{q}_j$. Compensating for this effect is more important in mapping and localization or in teach and repeat, but less important for odometry as we showed in Chapter 5. In this chapter, we include this correction factor, which depends on a continuous-time interpolation of the estimated body-centric velocity $\boldsymbol{\varpi}(t)$ at the measurement time of each target t_j, in the measurement error term for VTR3 radar odometry:

$$\mathbf{e}_{\text{odom},j} = \mathbf{D}\left(\mathbf{p}_k^j - \mathbf{T}(t_j)^{-1}\mathbf{T}_{rs}(\mathbf{q}_j + \Delta\mathbf{q}_j)\right) \tag{7.9}$$

$$\text{where } \Delta\mathbf{q}_j = \mathbf{D}^T\beta\mathbf{a}_j\mathbf{a}_j^T\mathbf{D}\mathbf{q}_j^{\odot}\text{Ad}(\mathbf{T}_{sr})\boldsymbol{\varpi}(t_j), \tag{7.10}$$

β is Doppler distortion constant inherent to the sensor [28], and $\mathbf{a}_j$ is a 3×1 unit vector in the direction of $\mathbf{q}_j$.

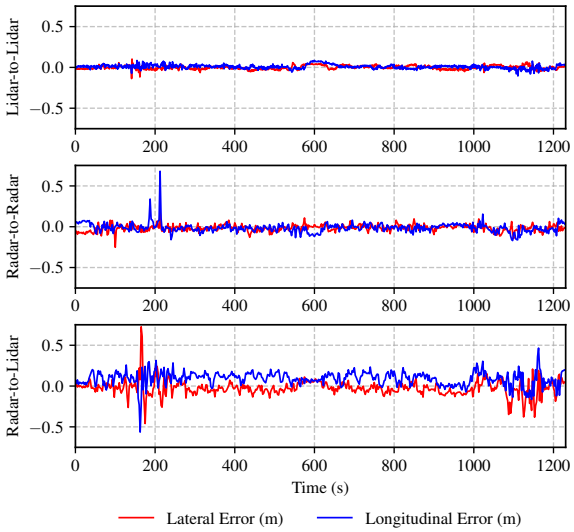
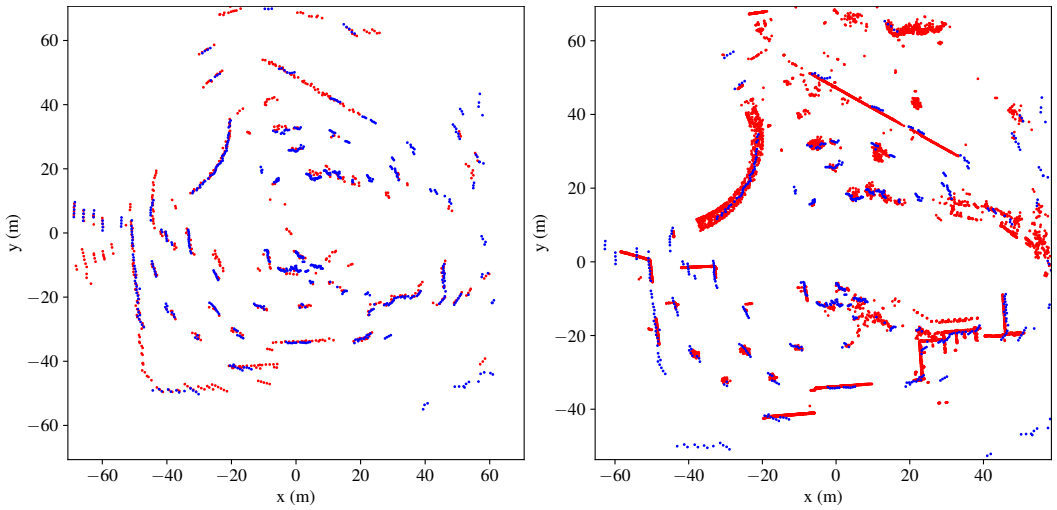


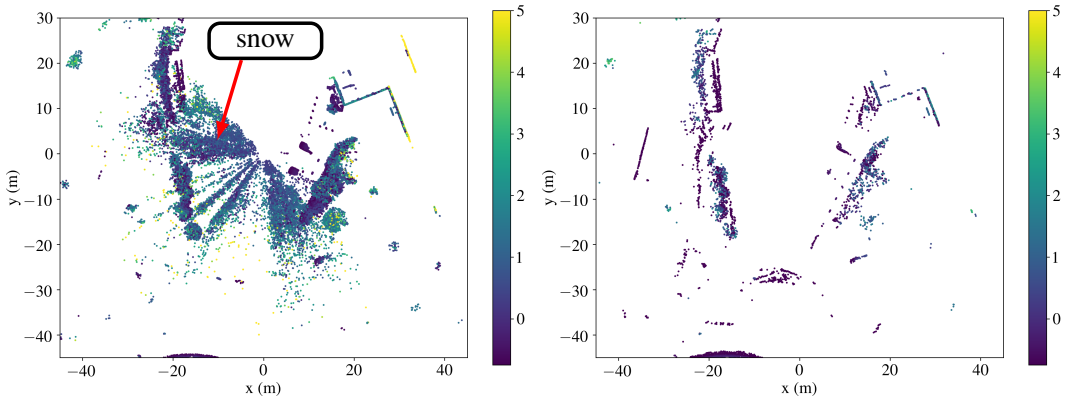
Figure 7.6: Here we plot metric localization errors during a snowstorm (2021-01-26). Note that lidar localization estimates remain accurate even with 1/4 of its field of view being blocked by a layer of ice as shown in Figure 7.8.



(a) Radar-to-Radar

(b) Radar-to-Lidar

Figure 7.7: This figure shows the live radar pointcloud (blue) registered to a submap (red) built during the teach pass. In (a) we are performing radar-to-radar localization and so the submap is made up of radar points. In (b) we are localizing a radar pointcloud (blue) to a previously built lidar submap.



(a) Lidar with snow detections

(b) Lidar with outliers removed

Figure 7.8: This figure illustrates the noisy lidar data used to localize during the 2021-01-26 sequence. Points are coloured by their z-height. In (a), the ground plane has been removed, and the pointcloud has been randomly downsampled by 50% to highlight the snowflake detections. Note that a layer of ice blocks a large forward section of the lidar's field of view. However, as the results in Table 7.2 and Figure 7.1.5 show, lidar localization remains quite robust under these adverse conditions. (b) shows the lidar pointcloud after filtering points by their normal score as in Equation 7.3 and only retaining the inliers of truncated least squares. Note that the snowflake detections seem to disappear, illustrating the robustness of our lidar pipeline.

7.2 Experimental Results

This section compares the performance of radar-only, lidar-only, and radar-to-lidar topometric localization using the Boreas datset. See Chapter 4 for more details on the Boreas dataset. In this experiment, we used seven sequences of the Glen Shields route (shown in Figure 7.1) chosen for their distinct weather conditions.. These sequences are depicted in Figure 7.4. During the teach pass, a map is constructed using the reference sequence 2020-11-26. The radar-only and lidar-only pipelines use their respective sensor types to construct the map. No GPS or IMU information is required during the map-building process. Note that our test sequences include significant seasonal variation, with ten months separating the initial teach pass and the final repeat pass. Sequences 2021-06-29 and 2021-09-08 include trees with full foliage, while the remaining sequences lack this. 2021-01-26 was collected during a snowstorm, 2021-06-29 was collected in the rain, and 2021-09-08 was collected at night.

During each of the repeat traversals, our topometric localization outputs a relative localization estimate between the live sensor frame s_2 and a sensor frame in the map s_1: $\hat{\mathbf{T}}_{s_1,s_2}$. We then compute RMSE values for the relative translation and rotation error as in [31]. We separate translational error into lateral and longitudinal components. Since the Navtech radar is a 2D sensor, we restrict our comparison to SE(2) by omitting z errors and reporting heading error as the rotation error.

Figure 7.1.4 depicts the spread of localization error during sequence 2021-01-26. Note that, although lidar-to-lidar localization is the most accurate, radar-to-radar localization remains reasonably competitive. The longitudinal and heading errors incur a bias when localizing radar scans to lidar maps. The longitudinal bias could be due to some residual Doppler distortion effects, and the

heading bias could result from an error in the radar-to-lidar extrinsic calibration. A video showcasing radar mapping and localization can be found at this link[4].

Figure 7.1.5 shows the localization errors as a function of time during the snowstorm sequence 2021-01-26. Surprisingly, lidar localization appears to be unperturbed by the adverse weather conditions. During the snowstorm sequence, the lidar pointcloud becomes littered with detections associated with snowflakes, and a large section of the horizontal field of view becomes blocked by a layer of ice as shown in Figure 7.8 (a). However, in Figure 7.8 (b), we show that in actuality, these snowflake detections have little impact on ICP registration after filtering by normal score and only retaining the inliers of truncated least squares. Charron et al. [40] previously demonstrated that snowflake detections can be removed from lidar pointclouds, although we do not use their method here. The robustness of our lidar pipeline to both weather and seasonal variations is reflected in the RMSE results displayed in Table 7.2.

Our results show that contrary to the assertions made by prior works, lidar localization can be robust to moderate levels of precipitation and seasonal variation. More work is required by the community to identify operational conditions where radar localization has a clear advantage. These conditions may include heavy precipitation, dense fog, or dust clouds. Nevertheless, we demonstrated that our radar localization is reasonably competitive with lidar. Furthermore, radar localization may still be an important redundant system in autonomous vehicles. Figure 7.7 illustrates the live scan and submap during radar-to-radar and radar-to-lidar localization.

It is important to recognize that the results reported in this chapter are taken at a snapshot in time. Radar localization is not as mature of a field as lidar localization and radar sensors still have room to improve. Incorporating IMU or wheel encoder measurements would improve the performance of all three compared systems. The detector we used, BFAR [7], did not immediately work when applied to a new radar with different noise characteristics. A learning-based approach to feature extraction and matching may improve performance. Switching to a landmark-based pipeline or one based on image correlation may also be interesting avenues for comparison.

Radar-to-lidar localization is attractive because it allows us to use existing lidar maps, which many autonomous driving companies already have, while taking advantage of the robustness of radar sensing. Radar-based maps are less useful than lidar maps since they lack sufficient detail to create semantic maps.

Table 7.2 shows the computational and storage requirements of the different pipelines discussed in this chapter. We used a Lenovo P53 laptop with Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz and 32GB of memory. A GPU was not used. Our radar-based maps use less storage (5.6MB/km) than our lidar-based maps (86.4MB/km).

7.3 Conclusions

In this chapter, we compared the performance of lidar-to-lidar, radar-to-radar, and radar-to-lidar topometric localization. Our results showed that radar-based pipelines are a viable alternative to lidar localization, but lidar continues to yield the best results. Surprisingly, our experiments showed that lidar-only mapping and localization is robust to adverse weather, such as a snowstorm with a partial sensor blockage due to ice. We identified several areas for future work and noted that more

[4] `https://youtu.be/okS7pF6xX7A`

Table 7.1: Metric Localization RMSE Results (Reference Sequence: 2020-11-26)

	Lidar-to-Lidar		
	lateral (m)	longitudinal (m)	heading (deg)
2020-12-04	0.060	0.059	0.035
2021-01-26	0.023	0.026	0.040
2021-02-09	0.030	0.031	0.041
2021-03-09	0.021	0.028	0.035
2021-06-29	0.025	0.056	0.050
2021-09-08	0.030	0.036	0.048
mean	**0.032**	**0.039**	**0.042**
	Radar-to-Radar		
	lateral (m)	longitudinal (m)	heading (deg)
2020-12-04	0.072	0.082	0.211
2021-01-26	0.048	0.055	0.227
2021-02-09	0.053	0.051	0.235
2021-03-09	0.051	0.053	0.233
2021-06-29	0.069	0.095	0.246
2021-09-08	0.067	0.110	0.269
mean	**0.060**	**0.074**	**0.237**
	Radar-to-Lidar		
	lateral (m)	longitudinal (m)	heading (deg)
2020-12-04	0.074	0.135	0.135
2021-01-26	0.095	0.128	0.183
2021-02-09	0.061	0.125	0.135
2021-03-09	0.057	0.123	0.135
2021-06-29	0.069	0.122	0.139
2021-09-08	0.063	0.108	0.161
mean	**0.070**	**0.124**	**0.148**

experiments are needed to identify conditions where the performance of radar-based pipelines exceeds that of lidar. Given these results, our goal is to reduce the performance gap between radar and lidar-based odometry by including an inertial measurement measurement unit (IMU). We investigate this topic in Chapter 9. In the following chapter, we compare treating IMU measurements as an input to a motion model vs. treating it as a measurement of the state. This investigation is important to this thesis as it informed our design decisions toward creating a radar-inertial odometry pipeline.

Table 7.2: Computational and Storage Requirements

	Odom. (FPS)	Loc. (FPS)	Storage (MB/km)
Lidar-to-Lidar	3.6	3.0	86.4
Radar-to-Radar	**5.3**	**5.1**	**5.6**
Radar-to-Lidar	N/A	**5.1**	86.4

Chapter 8

IMU as an Input versus a Measurement

Inertial measurement units (IMUs) are important sensors in state estimation. A popular approach in robotics is to treat IMU measurements as inputs to a motion model and then to numerically integrate the motion model to form relative motion factors between pairs of estimation times in a process known as preintegration [95, 55, 26]. In this paper, we investigate the treatment of IMUs as a measurement of the state using continuous-time state estimation with a Gaussian process motion prior. We compare treating an IMU as an input vs. a measurement on a simple 1D simulation problem. We then test our approach to lidar-inertial odometry using a simulated environment and compare it to a baseline representing the IMU-as-input approach. Finally, we provide experimental results for our lidar-inertial odometry on the Newer College Dataset.

Preintegration was initially devised to avoid estimating the state at each measurement time in (sliding-window) batch trajectory estimation. We will show that by employing our continuous-time estimation techniques, we can achieve the same big-O complexity as classic preintegration, which is linear in the number of measurements.

8.1 Related Work

Lupton and Sukkarieh [95] were the first to show that a temporal window of inertial measurements could be summarized using a single relative motion factor in a method known as preintegration. Forster et al. [55] then showed how to perform preintegration on the manifold $SO(3)$. Subsequently, Brossard et al. [26] demonstrated how to perform preintegration on the manifold of extended poses $SE_2(3)$, showing that this approach captures the uncertainty resulting from IMU measurements more consistently than $SO(3) \times \mathbb{R}^3$. These approaches treat the IMU as an input to a motion model. This approach has a few shortcomings. First, it conflates the IMU measurement noise with the underlying process noise. Second, it is unclear how the state and covariance should be propagated in the absence of IMU measurements. IMU measurement dropout is rare. However, it is worth considering the possibility in safety-critical applications. The third issue is that classic preintegration does not lend itself well to dealing with multiple high-rate sensors such as a lidar and an IMU or multiple asynchronous IMUs.

Previous work in continuous-time lidar-only odometry and lidar-inertial odometry include [53, 122, 112, 106, 96, 86] all of which employed B-splines. In B-spline approaches, exact derivatives of the continuous-time trajectory can be computed, creating unary factors for each accelerometer and gyroscope measurement and removing the need for preintegration. However, the spacing of control points greatly impacts the smoothness of B-spline trajectories. Determining this spacing is an important engineering challenge in B-spline approaches; this can be avoided by working with Gaussian processes instead. For a detailed comparison between B-splines and Gaussian processes in continuous-time state estimation, we refer the reader to [75].

Recently, Zheng and Zhu [163] demonstrated continuous-time lidar-inertial odometry using Gaussian processes where rotation is decoupled from translation. They use a white-noise-on-jerk prior for position in a global frame, a white-noise-on-acceleration prior for rotation using a sequence of local Gaussian processes, and a white-noise-on-velocity prior for the IMU biases. Using this approach, they demonstrate competitive performance on the HILTI SLAM benchmark [67]. One consequence of estimating position in a global frame is that the power spectral density matrix $\mathbf{Q}$ of the prior must typically be isotropic. In contrast, a body-centric approach allows lateral-longitudinal-vertical dimensions to be weighted differently. In addition, as was shown by Brossard et al. [26], decoupling rotation from translation does not capture the uncertainty resulting from IMU measurements as accurately as keeping them coupled. However, one clear advantage of their approach is that all parts of the state are directly observable by the measurements, whereas in our approach, angular acceleration is not directly observable.

In the previous chapter, we demonstrated continuous-time lidar-only odometry [30] using a white-noise-on-acceleration motion prior. Lidar-only odometry using a white-noise-on-jerk prior [143] and a Singer prior [154] have also been previously demonstrated. In this chapter, we choose to work with the Singer prior, which includes body-centric acceleration in the state. By including acceleration in the state, we can treat gyroscope and accelerometer measurements as direct measurements of the state rather than as inputs to a motion model.

Another approach based on Gaussian processes is that of Le Gentil and Vidal-Calleja [88, 87]. They employ six independent Gaussian processes, three for global frame acceleration and three for angular velocity. They optimize for the state at several inducing points given a set of IMU measurements over a preintegration window. They then analytically integrate these Gaussian processes to obtain preintegrated measurements that can be queried at any time of interest.

8.2 1D Simulation Comparison

In this section, we investigate an approach where IMU measurements are treated as direct measurements of the state using a continuous-time motion prior. We compare the performance of these two approaches on a simulated toy problem where we estimate the position and velocity of a 1D robot given noisy measurements of position and acceleration. Position measurements are acquired at a lower rate (10Hz), while acceleration measurements are acquired at a higher rate (100Hz). We aim to reduce the number of states estimated through preintegration for both approaches.

8.2.1 IMU as Input

As a baseline, we consider treating IMU measurements as inputs to a discrete motion model,

$$\underbrace{\begin{bmatrix}\mathbf{p}_k\\ \dot{\mathbf{p}}_k\end{bmatrix}}_{\mathbf{x}_k} = \underbrace{\begin{bmatrix}\mathbf{1} & \Delta t_k\mathbf{1}\\ \mathbf{0} & \mathbf{1}\end{bmatrix}}_{\mathbf{\Phi}(t_k,t_{k-1})}\underbrace{\begin{bmatrix}\mathbf{p}_{k-1}\\ \dot{\mathbf{p}}_{k-1}\end{bmatrix}}_{\mathbf{x}_{k-1}} + \underbrace{\begin{bmatrix}\frac{1}{2}\Delta t_k^2\\ \Delta t_k\end{bmatrix}}_{\mathbf{B}_k}\mathbf{u}_k, \tag{8.1}$$

where $\Delta t_k = t_k - t_{k-1}$, $\mathbf{\Phi}(t_k, t_{k-1})$ is the transition function, and $\mathbf{u}_k$ are acceleration measurements. A preintegration window is then defined between two endpoints, (t_{k-1}, t_k), which includes times $\tau_0, \ldots, \tau_J$. Following the approach of [95, 55], the preintegrated measurements $\Delta\mathbf{x}_{k,k-1}$ are computed as

$$\Delta\mathbf{x}_{k,k-1} = \sum_{n=1}^{J}\mathbf{\Phi}(\tau_J, \tau_n)\mathbf{B}_n\mathbf{u}_n. \tag{8.2}$$

These preintegrated measurements can be used to replace the acceleration measurements with a single relative motion factor between two endpoints:

$$J_{v,k} = \frac{1}{2}\mathbf{e}_k^T\mathbf{\Sigma}_k^{-1}\mathbf{e}_k, \tag{8.3a}$$

$$\mathbf{e}_k = \mathbf{x}_k - \mathbf{\Phi}(t_k, t_{k-1})\mathbf{x}_{k-1} - \Delta\mathbf{x}_{k,k-1}, \tag{8.3b}$$

where

$$\mathbf{\Sigma}_k = \sum_{n=1}^{J}\mathbf{\Phi}(\tau_J, \tau_n)\mathbf{B}_n\mathbf{Q}_n\mathbf{B}_n^T\mathbf{\Phi}(\tau_J, \tau_n)^T, \tag{8.4}$$

and $\mathbf{Q}_n$ is the covariance of the acceleration input $\mathbf{u}_n \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_n)$. Note that in this approach, uncertainty is propagated using the covariance on the acceleration input, $\mathbf{Q}_n$, which conflates IMU measurement noise and the underlying process noise. If the acceleration measurements were to drop out suddenly, it is unclear how the state and covariance should be propagated using this approach. It can be shown that preintegration is mathematically equivalent to marginalizing out the states between $\mathbf{x}_{k-1}$ and $\mathbf{x}_k$. The overall objective function that we seek to minimize is then

$$J(\mathbf{x}) = \sum_{k=0}^{K}\left(J_{v,k}(\mathbf{x}) + J_{y,k}(\mathbf{x})\right), \tag{8.5}$$

where

$$J_{y,k} = \frac{1}{2}\left(\mathbf{y}_k - \mathbf{C}_k\mathbf{x}_k\right)^T\mathbf{R}_k^{-1}\left(\mathbf{y}_k - \mathbf{C}_k\mathbf{x}_k\right) \tag{8.6}$$

are the measurement factors, and $\mathbf{R}_k$ is the associated measurement covariance.

8.2.2 A Generalization to Preintegration

In section 8.2.1, we showed how to perform preintegration when considering acceleration measurements as inputs to a motion model following closely from [95, 55]. In this section, we generalize the concept of preintegration to support heterogeneous factors (a combination of binary and unary factors). As a motivating example, consider having measurements of position, such as from a GPS, and measurements of acceleration coming from an accelerometer. These are unary measurement

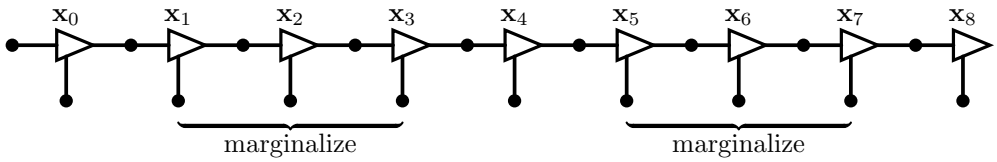


Figure 8.1: In this factor graph, we consider a case where we would like to marginalize several states out of the full Bayesian posterior. The triangles represent states, and the black dots represent factors. This factor graph could potentially correspond to doing continuous-time state estimation with binary motion prior factors, unary measurement factors, and a unary prior factor on the initial state $\mathbf{x}_0$.

factors, called such, because they only involve a single state. The binary factors here are motion prior factors derived from our Gaussian process motion prior, which is called binary because they are between two states. In classic preintegration, the only factors that are preintegrated are binary. Here, we show that we can simply use the formulas for querying a Gaussian process posterior at the endpoints of the preintegration window to form a preintegration factor that summarizes all the measurements contained therein. First, we consider the joint density of the state at a set of query times ($\tau_0 < \tau_1 < \cdots < \tau_J$) and the measurements,

$$p\left(\begin{bmatrix}\mathbf{x}_\tau \\ \mathbf{y}\end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix}\check{\mathbf{x}}_\tau \\ \mathbf{C}\check{\mathbf{x}}_\tau\end{bmatrix}, \begin{bmatrix}\check{\mathbf{P}}_{\tau,\tau} & \check{\mathbf{P}}_\tau \mathbf{C}^T \\ \mathbf{C}\check{\mathbf{P}}_\tau^T & \mathbf{R} + \mathbf{C}\check{\mathbf{P}}\mathbf{C}^T\end{bmatrix}\right). \tag{8.7}$$

We then perform the usual factoring using a Schur complement to obtain the posterior,

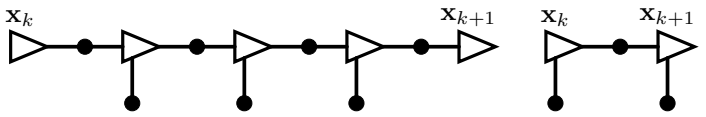
$$\begin{aligned} p(\mathbf{x}_\tau|\mathbf{y}) = \mathcal{N}\Big(&\check{\mathbf{x}}_\tau + \check{\mathbf{P}}_\tau \mathbf{C}^T(\mathbf{C}\check{\mathbf{P}}\mathbf{C} + \mathbf{R})^{-1}(\mathbf{y} - \mathbf{C}\check{\mathbf{x}}), \\ &\check{\mathbf{P}}_{\tau,\tau} - \check{\mathbf{P}}_\tau \mathbf{C}^T(\mathbf{C}\check{\mathbf{P}}\mathbf{C}^T + \mathbf{R})^{-1}\mathbf{C}\check{\mathbf{P}}_\tau^T\Big), \end{aligned} \tag{8.8}$$

where we obtain expressions for $\hat{\mathbf{x}}_\tau$ and $\hat{\mathbf{P}}_{\tau,\tau}$. We rearrange this further by inserting $\check{\mathbf{P}}^{-1}\check{\mathbf{P}}$ after the first instance of $\check{\mathbf{P}}_\tau$ and by applying the Sherman-Morrison-Woodbury identities to obtain

$$\hat{\mathbf{x}}_\tau = \check{\mathbf{x}}_\tau + \check{\mathbf{P}}_\tau\check{\mathbf{P}}^{-1}(\check{\mathbf{P}}^{-1} + \mathbf{C}^T\mathbf{R}^{-1}\mathbf{C})^{-1}\mathbf{C}^T\mathbf{R}^{-1}(\mathbf{y} - \mathbf{C}\check{\mathbf{x}}), \tag{8.9a}$$

$$\hat{\mathbf{P}}_{\tau,\tau} = \check{\mathbf{P}}_{\tau,\tau} - \check{\mathbf{P}}_\tau\check{\mathbf{P}}^{-1}(\check{\mathbf{P}}^{-1} + \mathbf{C}^T\mathbf{R}^{-1}\mathbf{C})^{-1}\mathbf{C}^T\mathbf{R}^{-1}\mathbf{C}\check{\mathbf{P}}_\tau^T, \tag{8.9b}$$

where we note that $(\check{\mathbf{P}}^{-1} + \mathbf{C}^T\mathbf{R}^{-1}\mathbf{C})$ is block-tridiagonal, and so the product $(\check{\mathbf{P}}^{-1} + \mathbf{C}^T\mathbf{R}^{-1}\mathbf{C})^{-1}\mathbf{C}^T\mathbf{R}^{-1}(\mathbf{y} - \mathbf{C}\check{\mathbf{x}})$ can be evaluated in $O(K)$ time using a sparse Cholesky solver. When we encounter products resembling $\mathbf{A}^{-1}\mathbf{b}$ where $\mathbf{A}$ is block-tridiagonal, we can instead solve $\mathbf{A}\mathbf{z} = \mathbf{b}$ for $\mathbf{z}$ using an efficient solver that takes advantage of the sparsity of $\mathbf{A}$. Finally, the product $\check{\mathbf{P}}_\tau\check{\mathbf{P}}^{-1}$ is quite sparse, having only two nonzero block columns per block row as shown earlier in (3.15). It follows that $\hat{\mathbf{x}}_\tau$ and $\hat{\mathbf{P}}_{\tau,\tau}$ can be computed in time that scales linearly with the number of measurements. Now, we consider the case where the query times consist of the beginning and end of a preintegration window, (t_k, t_{k+1}). The queried mean and covariance can be treated as pseudomeasurements summarizing the measurements contained in the preintegration window. We adjust our notation to make it clear that these are now being treated as measurements by using $\tilde{\mathbf{x}}$



(a) Original Preintegration Window (b) Preintegrated

Figure 8.2: This figure depicts the results of our preintegration, which can incorporate heterogeneous factors. The resulting joint Gaussian factor in (8.10) can be thought of as two unary factors, one each for $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$, and an additional binary factor between $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$.

instead of $\hat{\mathbf{x}}$. What we obtain is a joint Gaussian factor for the states at times t_k and t_{k+1},

$$J = \frac{1}{2} \begin{bmatrix} \mathbf{x}_k - \tilde{\mathbf{x}}_k \\ \mathbf{x}_{k+1} - \tilde{\mathbf{x}}_{k+1} \end{bmatrix}^T \begin{bmatrix} \tilde{\mathbf{P}}_{k,k} & \tilde{\mathbf{P}}_{k,k+1} \\ \tilde{\mathbf{P}}_{k,k+1} & \tilde{\mathbf{P}}_{k+1,k+1} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{x}_k - \tilde{\mathbf{x}}_k \\ \mathbf{x}_{k+1} - \tilde{\mathbf{x}}_{k+1} \end{bmatrix}. \tag{8.10}$$

A diagram depicting how this affects the resulting factor graph is shown in Figure 8.2. Using this approach, we can 'preintegrate' heterogeneous factors between pairs of states. One clear advantage of this approach is that it offers a tidy method for bookkeeping measurement costs and uncertainties. In the appendix, we provide an alternative formulation using a Schur complement with the same linear time complexity. Indeed, marginalization with a Schur complement is equivalent to the presented marginalization approach using a Gaussian process. However, the Gaussian process still serves a useful purpose in creating motion prior factors. Furthermore, the Gaussian process provides methods for interpolating the posterior.

It is unclear how to extend this marginalization approach to $SE(3)$ due to our choice to approximate $SE(3)$ trajectories using sequences of local Gaussian processes [10]. This marginalization approach could be applied using a global GP formulation like the one presented by Le Gentil and Vidal-Calleja [87]. However, their approach must contend with rotational singularities. We leave the extension to $SE(3)$ as an area of future work. In our implementation of lidar-inertial odometry, we instead use the posterior Gaussian process interpolation formula as in (3.31) to form continuous-time measurement factors. This is actually an approximation, as it is not exactly the same as marginalization. However, we have found that the interpolation approach works well in practice. Furthermore, the interpolation approach lends itself to parallelization, enabling a highly efficient implementation.

8.2.3 IMU as Measurement

In our proposed approach, we treat IMU measurements as direct measurements of the state within a continuous-time estimation framework. For the 1D toy problem, we chose to use a Singer prior, which is defined by the following linear time-invariant (LTI) stochastic differential equation (SDE),

$$\begin{aligned} \dddot{\mathbf{p}}(t) &= -\boldsymbol{\alpha}\ddot{\mathbf{p}}(t) + \mathbf{w}(t), \\ \mathbf{w}(t) &\sim \mathcal{GP}(\mathbf{0}, \mathbf{Q}_c \delta(t - t')), \end{aligned} \tag{8.11}$$

where $\mathbf{w}(t)$ is a white noise Gaussian process and $\mathbf{Q}_c = 2\boldsymbol{\alpha}\boldsymbol{\sigma}^2$ is the power spectral density matrix [153]. By varying $\boldsymbol{\alpha}$ and $\boldsymbol{\sigma}^2$, the Singer prior can model motion priors ranging from white-noise-on-acceleration ($\boldsymbol{\alpha} \to \boldsymbol{\infty}, \tilde{\boldsymbol{\sigma}}^2 = \boldsymbol{\alpha}^2\boldsymbol{\sigma}^2$) to white-noise-on-jerk ($\boldsymbol{\alpha} \to \mathbf{0}$).The discrete-time motion model

is given by

$$\mathbf{x}_k = \mathbf{\Phi}(t_k, t_{k-1})\mathbf{x}_{k-1} + \mathbf{w}_k, \quad \mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k), \tag{8.12}$$

where $\mathbf{w}_k$ is the process noise, $\mathbf{x}_k = [\mathbf{p}_k^T \ \dot{\mathbf{p}}_k^T \ \ddot{\mathbf{p}}_k^T]^T$, $\mathbf{\Phi}(t_k, t_{k-1})$ is the state transition function, and $\mathbf{Q}_k$ is the discrete-time covariance. Expressions for $\mathbf{\Phi}(t_k, t_{k-1})$ and $\mathbf{Q}_k$ for the Singer prior are provided by Wong et al. [153] and are repeated in the appendix. The binary motion prior factors are given by

$$J_{v,k} = \frac{1}{2}\mathbf{e}_k^T \mathbf{Q}_k^{-1} \mathbf{e}_k, \tag{8.13a}$$

$$\mathbf{e}_k = \mathbf{x}_k - \mathbf{\Phi}(t_k, t_{k-1})\mathbf{x}_{k-1}. \tag{8.13b}$$

The unary measurement factors have the same form as in (8.6) except that now accelerations are treated as direct measurements of the state. In addition, the overall objective is the same as in (8.5). We arrive at the linear system of equations from (3.12). By default, this approach would require estimating the state at each measurement time. To reduce the size of the state space, as in Section 8.2.1, we build preintegrated factors using the approach presented in Section 8.2.2 to bundle together both the unary measurement factors as well as the binary motion prior factors into a single factor. Similarly to the IMU-as-input approach, we preintegrate between pairs of states associated with the low-rate measurement times.

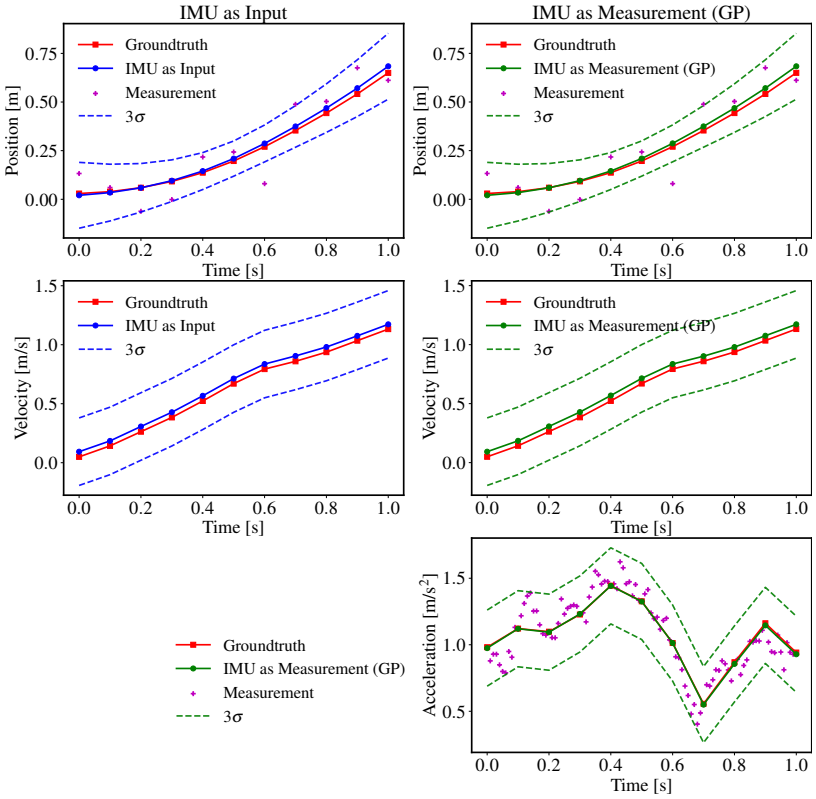


Figure 8.3: The estimated trajectories of IMU-as-input and IMU-as-measurement are plotted alongside the ground-truth trajectory, which is sampled from white-noise-on-jerk motion prior with $Q_c = 1.0$. Both methods were pretrained on a hold-out validation set of simulated trajectories.

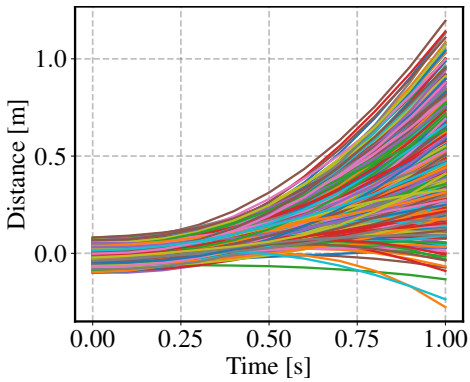


Figure 8.4: This figure depicts 1000 simulated trajectories sampled from a white-noise-on-jerk (WNOJ) prior where $\check{\mathbf{x}}_0 = [0.0\ 0.0\ 1.0]^T$, $\check{\mathbf{P}}_0 = \text{diag}(0.001, 0.001, 0.001)$, $Q_c = 1.0$.

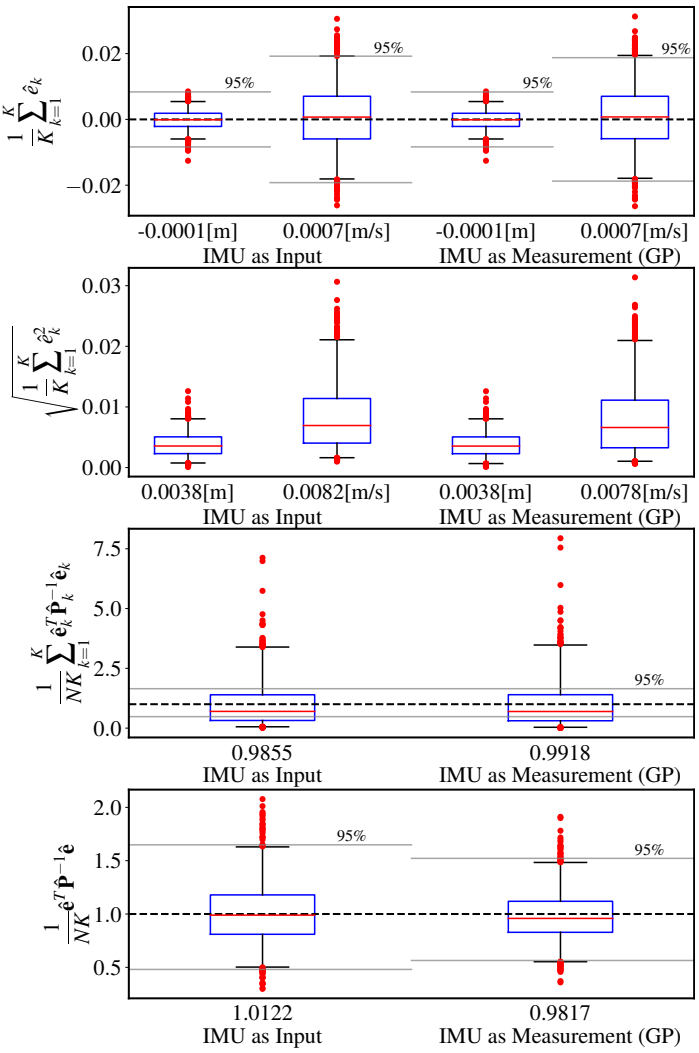


Figure 8.5: This figure compares the performance of the baseline IMU-as-input approach vs. our proposed IMU-as-measurement approach that leverages a Gaussian process (GP) motion prior. The ground-truth trajectories are sampled from a white-noise-on-jerk (WNOJ) prior, as shown in Figure 8.4. The IMU input covariance Q_k for the IMU-as-input method was trained on a validation set with a value of 0.00338. The parameters of our proposed method were also trained on the same validation set, with values of $\sigma^2 = 1.0069$, $\alpha = 0.0$. R_{pos} for both methods was chosen to match the simulated noise added to the position measurements. Similarly, R_{acc} for the IMU-as-measurement approach was set to match the simulated noise added to the acceleration measurements. In the bottom row, the chi-squared bounds have a different size because the dimension of the state in IMU-as-measurement is greater (it includes acceleration), and so the dimension of the chi-squared distribution increases, resulting in a tighter chi-squared bound.

8.2.4 Simulation Results

In this section, we compare two different approaches to combining high-rate acceleration measurements with low-rate position measurements. We refer to these two different approaches as IMU-as-input and IMU-as-measurement. The comparison is conducted on a toy 1D simulation problem.

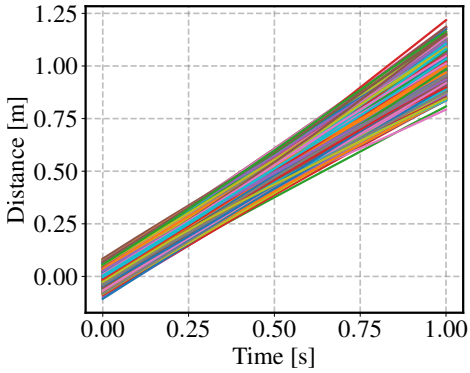


Figure 8.6: This figure depicts 1000 simulated trajectories sampled from a Singer prior where $\check{\mathbf{x}}_0 = [0.0\ 1.0\ 0.0]^T$, $\check{\mathbf{P}}_0 = \text{diag}(0.001, 0.001, 0.001)$, $\alpha = 10.0$, $\sigma^2 = 1.0$. A large value of α is intended to approximate a white-noise-on-acceleration (WNOA) prior.

We sample trajectories from a GP motion prior as defined in (3.11) by using standard methods for sampling from a multidimensional Gaussian [16].

In our first simulation, we sample trajectories from a white-noise-on-jerk (WNOJ) motion prior with $Q_c = 1.0$. Figure 8.3 provides a qualitative comparison of the IMU-as-input and IMU-as-measurement approaches for a single sampled simulation trajectory. Note that the performance of the two estimators, including their 3-σ confidence bounds, appears to be nearly identical. Figure 8.4 depicts 1000 trajectories sampled from this motion prior. To simulate noisy sensors, we also corrupt both position and acceleration measurements using zero-mean Gaussian noise where $y_{\text{pos},k} = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \mathbf{x}_k + n_{\text{pos},k}$, $n_{\text{pos},k} \sim \mathcal{N}(0, \sigma^2_{\text{pos}})$, $y_{\text{acc},k} = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \mathbf{x}_k + n_{\text{acc},k}$, $n_{\text{acc},k} \sim \mathcal{N}(0, \sigma^2_{\text{acc}})$. In the simulation, we set $\sigma_{\text{pos}} = 0.01m$, $\sigma_{\text{acc}} = 0.01m/s^2$.

Both IMU-as-input and IMU-as-measurement have parameters that must be tuned on the dataset. For this purpose, we use a separate training set of 100 sampled trajectories. For the IMU-as-input approach, we learn the covariance of the acceleration input Q_k using maximum likelihood over the training set, where we benefit from using noiseless ground truth states in simulation. The objective that we seek to minimize is

$$J = \frac{1}{2} \ln \left| \check{\mathbf{P}} \right| + \frac{1}{2T} \sum_{t=1}^{T} (\mathbf{x}_t - \check{\mathbf{x}})^T \check{\mathbf{P}}^{-1} (\mathbf{x}_t - \check{\mathbf{x}}), \tag{8.14}$$

where $\mathbf{x}_t$ are validation set trajectories, $\check{\mathbf{x}} = \mathbf{A}\mathbf{v}$ is the full trajectory prior for the IMU-as-input method, $\check{\mathbf{P}} = \mathbf{A}\mathbf{Q}\mathbf{A}^T$ is the prior covariance over the whole trajectory, $\mathbf{A}$ is the lifted transition matrix as in (3.7), $\mathbf{v} = \begin{bmatrix} \check{\mathbf{x}}_0^T & \Delta\mathbf{x}_{1,0}^T & \cdots & \Delta\mathbf{x}_{K,K-1}^T \end{bmatrix}^T$, and $\mathbf{Q} = \text{diag}(\check{\mathbf{P}}_0, \boldsymbol{\Sigma}_1, \cdots, \boldsymbol{\Sigma}_K)$.

Using a numerical optimizer, we found that $Q_k \approx 0.00338$ given a WNOJ motion prior with $Qc = 1.0$, and acceleration measurement noise of $\sigma^2_{\text{acc}} = 0.0001m^2/s^4$. Note that the input covariance Q_k is much greater than the simulated noise on the acceleration measurements σ^2_{acc}. This is because, for the IMU-as-input approach, the covariance on the acceleration input Q_k conflates two sources of noise: the IMU measurement noise and the underlying process noise. It also means that Q_k must be trained and adapted to new datasets to maintain a consistent estimator.

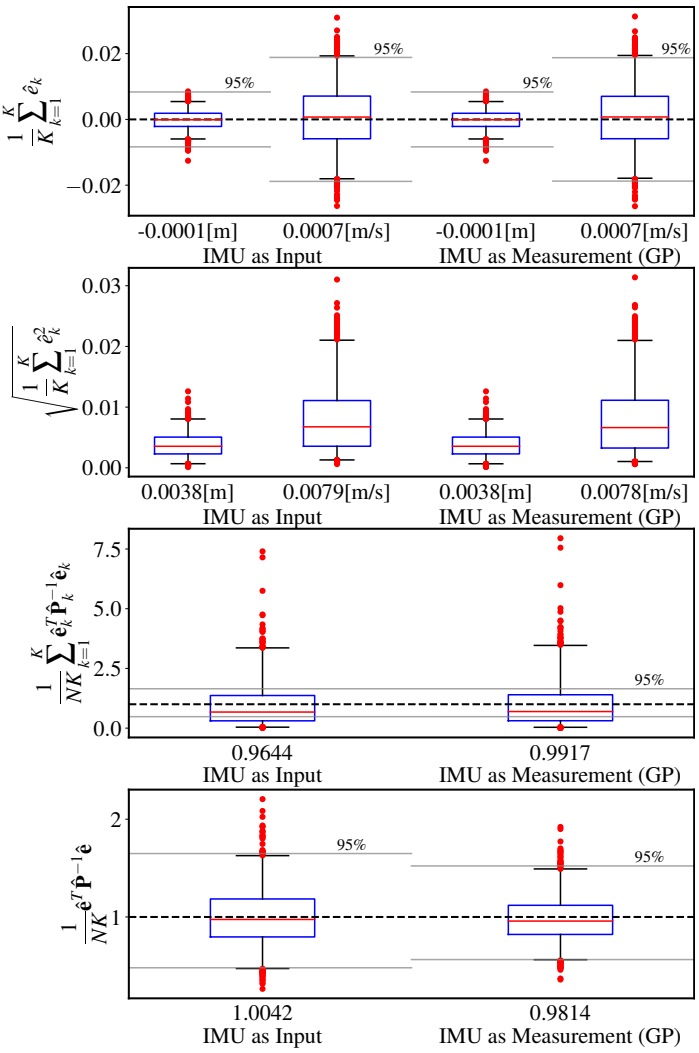


Figure 8.7: This figure summarizes the results of our second simulation experiment where the ground-truth trajectories were sampled from a Singer prior with $\alpha = 10.0$, $\sigma^2 = 1.0$. The trained acceleration input covariance for IMU-as-input was $Q_k \approx 0.00283$. The trained Singer prior parameters were $\alpha = 10.2442$, $\sigma^2 = 1.0074$. Note that even though the underlying prior changed by a lot from the first simulation to the second, from a white-noise-on-jerk prior to an approximation of a white-noise-on-jerk prior, both estimators were able to remain unbiased and consistent.

For the IMU-as-measurement approach, we need to train the parameters of our Gaussian process (GP) motion prior. To highlight the versatility of our proposed approach, we employ a Singer motion prior [153], which can model priors ranging from white-noise-on-acceleration (WNOA) to white-noise-on-jerk (WNOJ). The Singer prior is parametrized by an inverse length scale matrix $\boldsymbol{\alpha}$ and a variance $\boldsymbol{\sigma}^2$, both diagonal. For values of $\boldsymbol{\alpha}$ close to zero, numerical optimizers encounter difficulties due to numerical instabilities of $\mathbf{Q}_k$ and its Jacobians. Instead, we derive the analytical gradients to learn $\{\boldsymbol{\alpha}, \boldsymbol{\sigma}^2\}$ using gradient descent following the approach presented by Wong et al.

[153]. The objective that we seek to minimize is

$$J = \frac{1}{2}\sum_{t=1}^{T}\sum_{k=1}^{K}\left(\mathbf{e}_{k,t}^{T}\mathbf{Q}_{k,t}^{-1}\mathbf{e}_{k,t} + \ln|\mathbf{Q}_{k,t}|\right), \tag{8.15}$$

where both $\mathbf{e}_{k,t}$ and $\mathbf{Q}_{k,t}$ are functions of the Singer prior parameters $\{\boldsymbol{\alpha}, \boldsymbol{\sigma}^2\}$. Further details on the analytical gradients are provided in the appendix. Note that the approach of Wong et al. [153] supports learning the parameters of the Singer prior even with noisy ground truth; however, we must first estimate the measurement covariances and then keep them fixed during the optimization. To learn both the GP parameters and the measurement covariances simultaneously, Wong et al. leverage exactly sparse Gaussian variational inference [154].

Figure 8.5 shows the results of our first simulation experiment with the WNOJ prior. Each row in Figure 8.5 is a box plot of a metric computed independently for each of the 1000 simulated trajectories. The blue boxes represent the interquartile range, the red lines are the medians, the whiskers correspond to the 2.5 and 97.5 percentiles, and the red dots denote outliers (data points beyond the whiskers). The black dashed lines in the first, third, and fourth rows correspond to the target value of 0 for the mean error and 1 for the normalized estimation error squared (NEES). Underneath each box plot, we also compute the mean value of the metrics across all data points.

In the first row of Figure 8.5, we can see that the mean error in both position and velocity is close to zero for both estimators. The grey lines in the first row denote a 95% two-sided confidence interval, a statistical test to check that the estimators are unbiased. We expect to see the whiskers of the box plots lie within the 95% confidence interval to confirm that the estimator is unbiased, which is the case.

The third row displays a commonly used method for computing the normalized estimation error squared (NEES). This method uses the marginals of the posterior covariance and relies on the ergodic hypothesis to treat the error from each timestep as being independent. In this case, we compute the marginal covariance at each timestep for position and velocity only so that the results of the two estimators can be compared directly. The grey lines denote a 95% chi-squared bound, a statistical test for checking that the estimators are consistent. Interestingly, we observe that neither estimator passes the statistical test in this case even though the mean and median NEES are close to 1. It appears that, in this case, the ergodic hypothesis is not valid.

In the fourth row, we present an alternative formulation of the NEES that uses the full posterior covariance over the entire trajectory. In this case, we are satisfied that both estimators pass the statistical test to confirm that they are consistent. The main difference between this version of the NEES and the previous one is that we have retained the cross-covariance terms between timesteps.

In summary, we observe that the two approaches achieve nearly identical performance. Both estimators are unbiased and consistent so long as their parameters are trained on a training set.

Figures 8.6, 8.7 depict the results of our second experiment where the ground-truth trajectories are sampled from a Singer prior with $\alpha = 10.0$, $\sigma^2 = 1.0$. This large value of α is intended to approximate a white-noise-on-acceleration prior. Our results show that both estimators are capable of adapting to a dataset with a different underlying motion prior while remaining unbiased and consistent.

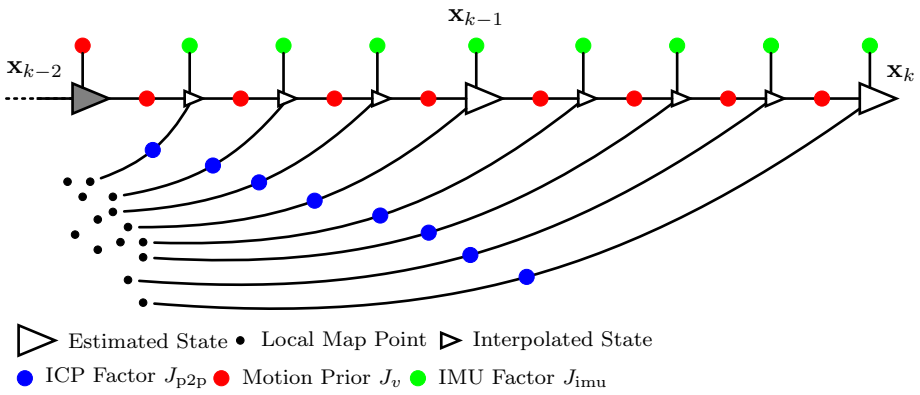


Figure 8.8: This figure depicts a factor graph of our sliding window lidar-inertial odometry using a continuous-time motion prior. The larger triangles represent the estimation times that form our sliding window. The state $\mathbf{x}(t) = \{\mathbf{T}(t), \boldsymbol{\varpi}(t), \dot{\boldsymbol{\varpi}}(t), \mathbf{b}(t)\}$ includes the pose $\mathbf{T}(t)$, the body-centric velocity $\boldsymbol{\varpi}(t)$, the body-centric acceleration $\dot{\boldsymbol{\varpi}}(t)$, and the IMU biases $\mathbf{b}(t)$. The grey-shaded state $\mathbf{x}_{k-2}$ is next to be marginalized and is held fixed during the optimization of the current window. The smaller triangles are interpolated states that we do not directly estimate during optimization. Instead, we construct continuous-time measurement factors using the posterior Gaussian process interpolation formula. We include a unary prior on $\mathbf{x}_{k-2}$ to denote the prior information from the sliding window filter.

8.2.5 Discussion

As mentioned previously, the big-O time complexity of classic preintegration and our approach are the same. In practice, our approach is slightly slower but not by much. Using a modern CPU, either approach can be considered real-time capable. Note that the number of preintegration windows could be adjusted and each preintegration window could be computed in parallel to make the approach more efficient. In this way, we could parallelize the solving of some estimation problems. We are motivated by sensor configurations that cannot be easily handled by classic preintegration such as multiple asynchronous high-rate sensors. This could include a lidar and an IMU or multiple asynchronous IMUs. Note that our approach also enables the preintegration of more than just IMU measurements and can include additional measurements such as position and velocity measurements. We only demonstrate our preintegration on a vector space in section 8.2.2. The extension of this new preintegration method to SE(3) remains an area of future work.

8.3 Lidar-Inertial Odometry

Our lidar-inertial odometry is implemented as sliding-window batch trajectory estimation. The factor graph corresponding to our approach is depicted in Figure 8.8. The state $\mathbf{x}(t) = \{\mathbf{T}(t), \boldsymbol{\varpi}(t), \dot{\boldsymbol{\varpi}}(t), \mathbf{b}(t)\}$ consists of the $SE(3)$ pose $\mathbf{T}_{vi}(t)$, the body-centric velocity $\boldsymbol{\varpi}_v^{vi}(t)$, the body-centric acceleration $\dot{\boldsymbol{\varpi}}_v^{vi}(t)$, and the IMU biases $\mathbf{b}(t)$. We approximate the $SE(3)$ trajectory using a sequence of local Gaussian processes as in [10]. Between pairs of estimation times, the local variable $\boldsymbol{\xi}_k(t)$ is defined as

$$\boldsymbol{\xi}_k(t) = \ln(\mathbf{T}(t)\mathbf{T}(t_k)^{-1})^\vee. \tag{8.16}$$

We use (8.16) and the following to convert between global and local variables:

$$\dot{\boldsymbol{\xi}}_k(t) = \boldsymbol{\mathcal{J}}(\boldsymbol{\xi}_k(t))^{-1}\boldsymbol{\varpi}(t), \tag{8.17}$$

$$\ddot{\boldsymbol{\xi}}_k(t) \approx -\frac{1}{2}(\boldsymbol{\mathcal{J}}(\boldsymbol{\xi}_k(t))^{-1}\boldsymbol{\varpi}(t))^{\curlywedge}\boldsymbol{\varpi}(t) + \boldsymbol{\mathcal{J}}(\boldsymbol{\xi}_k(t))^{-1}\dot{\boldsymbol{\varpi}}(t), \tag{8.18}$$

where the approximation for $\ddot{\boldsymbol{\xi}}_k(t)$ was originally derived by Tang et al. [143]. We use a Singer prior, introduced in [153], which is defined by the following Gaussian process,

$$\ddot{\boldsymbol{\xi}}_k(t) \sim \mathcal{GP}(\mathbf{0}, \boldsymbol{\sigma}^2 \exp(-\boldsymbol{\ell}^{-1}|t - t'|)), \tag{8.19}$$

and which can equivalently be represented using the following linear time-invariant stochastic differential equation,

$$\dot{\boldsymbol{\gamma}}_k(t) = \mathbf{A}\boldsymbol{\gamma}_k(t) + \mathbf{L}\mathbf{w}(t), \tag{8.20}$$

where $\mathbf{w}(t) \sim \mathcal{GP}(\mathbf{0}, \mathbf{Q}_c\delta(t - t')$,

$$\boldsymbol{\gamma}_k(t) = \begin{bmatrix} \boldsymbol{\xi}_k(t) \\ \dot{\boldsymbol{\xi}}_k(t) \\ \ddot{\boldsymbol{\xi}}_k(t) \end{bmatrix}, \ \mathbf{A} = \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\boldsymbol{\alpha} \end{bmatrix}, \ \mathbf{L} = \begin{bmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{1} \end{bmatrix},$$

$\boldsymbol{\sigma}^2$ is a variance, $\boldsymbol{\ell}$ is a length scale, $\boldsymbol{\alpha} = \boldsymbol{\ell}^{-1}$, and $\mathbf{w}(t)$ is a white-noise Gaussian process where $\mathbf{Q}_c = 2\boldsymbol{\alpha}\boldsymbol{\sigma}^2$ is the associated power spectral density matrix. As shown in (3.21), (8.20) can be stochastically integrated to arrive at a local Gaussian process

$$\boldsymbol{\gamma}_k(t) \sim \mathcal{GP}(\boldsymbol{\Phi}(t, t_k)\check{\boldsymbol{\gamma}}_k(t_k)), \boldsymbol{\Phi}(t, t_k)\check{\mathbf{P}}(t_k)\boldsymbol{\Phi}(t, t_k)^T + \mathbf{Q}_k), \tag{8.21}$$

where the formulation for the transition function $\boldsymbol{\Phi}(t, t_k)$ and the covariance $\mathbf{Q}_k$ can be found Appendix E. To convert our continuous-time formulation into a factor graph, we build a sequence of motion prior factors between pairs of estimation times using (3.25). Our IMU measurement model is

$$\begin{bmatrix} \tilde{\mathbf{a}} \\ \tilde{\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} \mathbf{a}_v^{vi} - \mathbf{C}_{vi}\mathbf{g}_i \\ \boldsymbol{\omega}_v^{vi} \end{bmatrix} + \begin{bmatrix} \mathbf{b}_a \\ \mathbf{b}_\omega \end{bmatrix} + \begin{bmatrix} \mathbf{w}_a \\ \mathbf{w}_\omega \end{bmatrix}, \tag{8.22}$$

where $\mathbf{b}_a$ and $\mathbf{b}_\omega$ are the accelerometer and gyroscope biases, $\mathbf{w}_a \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_a)$ and $\mathbf{w}_\omega \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_\omega)$ are zero-mean Gaussian noise. Due to angular velocity and acceleration being a part of the state, the associated IMU error function is straightforward:

$$J_{\text{imu},\ell} = \frac{1}{2}\begin{bmatrix} \mathbf{e}_{a,\ell} \\ \mathbf{e}_{\omega,\ell} \end{bmatrix}^T \begin{bmatrix} \mathbf{R}_a & \\ & \mathbf{R}_\omega \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{e}_{a,\ell} \\ \mathbf{e}_{\omega,\ell} \end{bmatrix}, \tag{8.23a}$$

$$\mathbf{e}_{a,\ell} = \tilde{\mathbf{a}}_\ell - \dot{\boldsymbol{\nu}}(\tau_\ell) + \mathbf{C}_{vi}(\tau_\ell)\mathbf{g}_i - \mathbf{b}_a(\tau_\ell), \tag{8.23b}$$

$$\mathbf{e}_{\omega,\ell} = \tilde{\boldsymbol{\omega}}_\ell - \boldsymbol{\omega}(\tau_\ell) - \mathbf{b}_\omega(\tau_\ell), \tag{8.23c}$$

where we rely on forming measurement factors using the posterior Gaussian process interpolation formula. For each of these continuous-time measurement factors, we compute Jacobians of the perturbation to the state at the interpolated times with respect to the bracketing estimation times. This is an approximation, as it is not exactly the same as marginalization. However, we have found it to work well in practice. We also include motion prior factors for the IMU biases,

$$J_{v,b,k} = \frac{1}{2}\mathbf{e}_{v,b,k}^T \mathbf{Q}_{b,k}^{-1} \mathbf{e}_{v,b,k}, \tag{8.24a}$$

$$\mathbf{e}_{v,b,k} = \mathbf{b}(t_{k+1}) - \mathbf{b}(t_k), \tag{8.24b}$$

where $\mathbf{Q}_{b,k} = \boldsymbol{Q}_b \Delta t_k$ is the covariance resulting from a white-noise-on-velocity motion prior, and $\boldsymbol{Q}_b$ is the associated power spectral density matrix. We use point-to-plane factors. The associated error function is

$$J_{\mathrm{p2p},j} = \mathbf{e}_{\mathrm{p2p},j}^T \mathbf{R}_{\mathrm{p2p},j}^{-1} \mathbf{e}_{\mathrm{p2p},j}, \tag{8.25a}$$

$$\mathbf{e}_{\mathrm{p2p},j} = \alpha_j \mathbf{n}_j^T \mathbf{D}(\mathbf{p}_j - \mathbf{T}_{vi}(\tau_j)^{-1} \mathbf{T}_{vs} \mathbf{q}_j), \tag{8.25b}$$

where $\mathbf{q}_j$ is the query point, $\mathbf{p}_j$ is the matched point in the local map, $\mathbf{n}_j$ is an estimate of the surface normal at $\mathbf{p}_j$ given neighboring points in the map, $\mathbf{D}$ is a constant matrix removing the homogeneous component, $\mathbf{T}_{vs}$ is a known extrinsic calibration between the lidar frame and the vehicle frame, and $\alpha_j = (\sigma_2 - \sigma_3)/\sigma_1$ [48] is a heuristic weight to favour planar neighborhoods. The objective function that we seek to minimize is

$$J = \sum_k J_{v,k} + \sum_j J_{\mathrm{p2p},j} + \sum_\ell J_{\mathrm{imu},\ell}. \tag{8.26}$$

We solve this nonlinear least squares problem for the optimal perturbation to the state using Gauss-Newton. Once the solver has converged, we update the pointcloud correspondences and iterate this two-step process to convergence. In practice, we typically limit the maximum number of inner-loop Gauss-Newton iterations to 10, and the number of outer-loop iterations to 10 to enable real-time operation.

In our approach, we estimate the orientation of the gravity vector relative to the initial map frame at startup. We perform sliding-window batch trajectory estimation where the sliding window length is 200ms. We output the pose in the middle of the newest lidar scan.

8.4 IMU-as-Input Lidar-Inertial Baseline

As a baseline where IMU measurements are treated as an input, we consider a lidar-inertial odometry approach where IMU measurements are used to de-skew the lidar pointcloud, and classic preintegration is used as a prior. The baseline is implemented as a sliding-window batch trajectory estimation, and the factor graph corresponding to the baseline approach is depicted in Figure 8.9. The state $\mathbf{x}(t_k) = \{\mathbf{C}_{iv}(t_k), \mathbf{r}_i^{vi}(t_k), \mathbf{v}_i^{vi}(t_k), \mathbf{b}(t_k)\}$ consists of the orientation $\mathbf{C}_{iv}(t_k)$, position $\mathbf{r}_i^{vi}(t_k)$, velocity $\mathbf{v}_i^{vi}(t_k)$, and IMU biases. All variables are expressed in a global frame. We use classic preintegration to form binary factors between pairs of estimated states in the sliding window [55]. At each iteration of the optimization, we integrate the IMU measurements to extrapolate for the state at each IMU

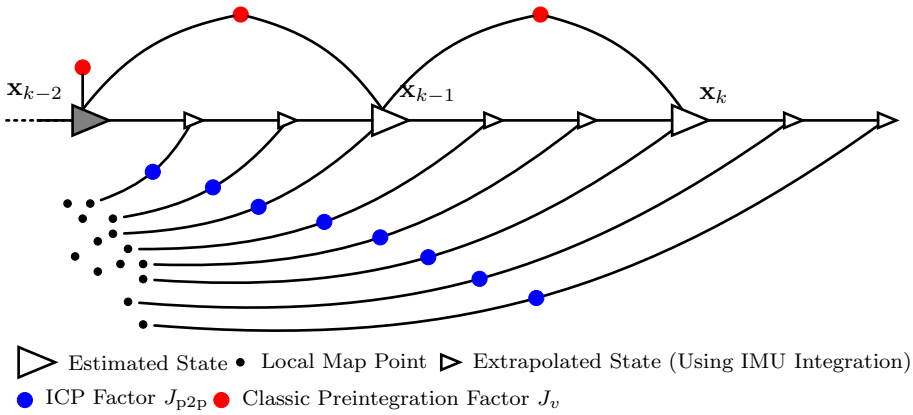


Figure 8.9: This figure depicts a factor graph of our baseline approach that uses IMU measurements to de-skew the pointcloud and to form relative motion priors using classic preintegration. The larger triangles represent the estimation times that form our sliding window. The state $\mathbf{x}(t_k) = \{\mathbf{C}_{iv}(t_k), \mathbf{r}_i^{vi}(t_k), \mathbf{v}_i^{vi}(t_k), \mathbf{b}(t_k)\}$ includes the orientation and position in a global frame, the velocity in a global frame, and the IMU biases. The grey-shaded state $\mathbf{x}_{k-2}$ is next to be marginalized. The smaller triangles are extrapolated states that we do not directly estimate during optimization. Instead, we extrapolate for these states using IMU integration starting at an estimated state. The factor graph includes a unary prior on $\mathbf{x}_{k-2}$ to denote the prior information from the sliding window filter

measurement time using

$$\mathbf{C}_j = \mathbf{C}_i \prod_{k=i}^{j-1} \exp\left(\Delta t_k (\tilde{\boldsymbol{\omega}}_k - \mathbf{b}_\omega(t_k))^\wedge\right), \tag{8.27a}$$

$$\mathbf{v}_j = \mathbf{v}_i + \mathbf{g}\Delta t_{ij} + \sum_{k=i}^{j-1} \mathbf{C}_k(\tilde{\mathbf{a}}_k - \mathbf{b}_a(t_k))\Delta t_k, \tag{8.27b}$$

$$\mathbf{r}_j = \mathbf{r}_i + \sum_{k=i}^{j-1} \left[\mathbf{v}_k \Delta t_k + \frac{1}{2}\mathbf{g}\Delta t_k^2 + \frac{1}{2}\mathbf{C}_k(\tilde{\mathbf{a}}_k - \mathbf{b}_a(t_k))\Delta t_k^2\right]. \tag{8.27c}$$

The position at a given lidar point time can be obtained by linearly interpolating between the positions at the IMU measurement times. The orientation at a given lidar point time can be obtained using the following formula,

$$\mathbf{C}(\tau_j) = \mathbf{C}(t_\ell)\left(\mathbf{C}(t_\ell)^T \mathbf{C}(t_{\ell+1})\right)^\alpha \tag{8.28}$$

where $\alpha = (\tau_j - t_\ell)/(t_{\ell+1} - t_\ell)$. Using these interpolated states, we can write the point-to-plane error function as

$$\mathbf{e}_{\text{p2p},j} = \mathbf{n}_j^T\left(\mathbf{p}_j - \mathbf{C}_{iv}(\tau_j)(\mathbf{C}_{vs}\mathbf{q}_j + \mathbf{r}_v^{sv}) - \mathbf{r}_i^{vi}(\tau_j)\right). \tag{8.29}$$

The Jacobians of this error function with respect to perturbations to the state variables are provided in the appendix.

8.5 Lidar-Inertial Simulation

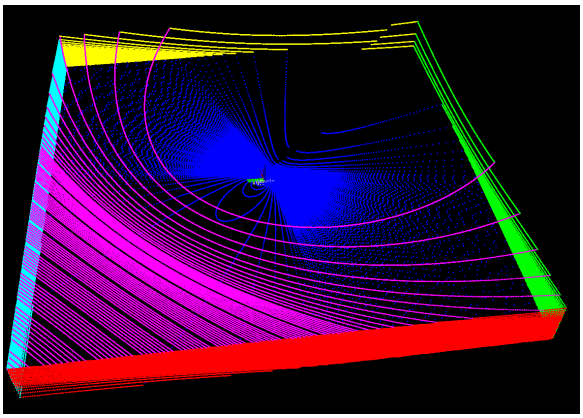


Figure 8.10: This figure depicts an example lidar pointcloud produced by our simulation, which contains motion distortion. The pointcloud is coloured based off which wall the lidar point is reflected

In this section, we compare the performance of our lidar-inertial odometry to the baseline imu-as-input approach in a simulated environment. The simulated environment is a rectangular room, and we simulate trajectories using sinusoidal body-centric velocities,

$$[\boldsymbol{\varpi}(t)]_j = A_j \sin(2\pi f_j t), \tag{8.30}$$

where A_j and f_j are configurable amplitude and frequency parameters. The resulting body-centric acceleration can be obtained via differentiation,

$$[\dot{\boldsymbol{\varpi}}(t)]_j = A_j 2\pi f_j \cos(2\pi f_j t). \tag{8.31}$$

We then step through the simulation to replicate the lidar firing sequence of a Velodyne Alpha-Prime 128-beam lidar, obtaining the pose of the sensor for each firing sequence. Starting with $\mathbf{T}_0 = \mathbf{T}_{vi}(t_0) = \mathbf{1}$,

$$\mathbf{T}_{k+1} \approx \exp\left(\left(\boldsymbol{\varpi}(t_k)\Delta t_k + \frac{1}{2}\Delta t_k^2 \dot{\boldsymbol{\varpi}}(t_k)\right)^\wedge\right)\mathbf{T}_k, \tag{8.32}$$

where Δt_k is very small (53.3μs). By generating the trajectories in this way, it is straightforward to extract the body-centric angular velocity and linear acceleration to simulate IMU measurements. We use the measurement model in (8.22) to simulate biases and gravity components. We simulate a constant nonzero bias on each gyroscope and accelerometer axis. We include zero-mean Gaussian noise on IMU measurements and Gaussian noise on the range measurement of each lidar point. We chose measurement noises that were close to what we experienced on our experimental platform. Figure 8.10 depicts an example pointcloud produced in our simulation environment where the points are coloured based on which wall they are reflected. Figure 8.11 compares the trajectory estimated by our lidar-inertial odometry with the ground truth.

For the simulation parameters, we use an IMU rate of 200Hz, a simulation length of 20s, and three motion regimes denoted *slow*, *medium*, and *fast*. For each of these motion regimes, we randomly sample the amplitudes and frequencies of the body-centric velocities used in the simulation of a sequence. Table 8.1 gives the ranges for these parameters. One set of amplitudes and frequencies

Table 8.1: Lidar-inertial simulation parameter ranges for the different motion regimes.

	Slow	Medium	Fast
Linear Velocity Amplitude [m/s]	$A \in [0.1, 0.5]$	$A \in [0.5, 1.0]$	$A \in [1.0, 2.0]$
Angular Velocity Amplitude [rad/s]	$A \in [0.1, 0.5]$	$A \in [0.5, 1.0]$	$A \in [1.0, 2.0]$
Linear Velocity Frequency [Hz]	$f \in [0.5, 1.0]$	$f \in [1.0, 2.0]$	$f \in [2.0, 4.0]$
Angular Velocity Frequency [Hz]	$f \in [1.0, 2.0]$	$f \in [2.0, 4.0]$	$f \in [4.0, 8.0]$

Table 8.2: Simulation results. Root Mean Squared Absolute Trajectory Error (m). For each speed category (slow, medium, fast), 20 randomized sequences were created. The results in this table are the overall root mean squared absolute position error across 20 sequences, for each approach

	Slow	Medium	Fast
Baseline (IMU as Input)	0.0026	2.1734	Failed
Singer-LIO	0.0026	**0.0025**	**0.0208**
Singer-LO + Gyro	0.0052	0.0085	0.0445
Singer-LO	**0.0012**	12.34	Failed

is sampled for each of the 20 sequences simulated for the three motion regimes. We set the standard deviation of the accelerometer measurement noise to 0.02 m/s^2, the standard deviation of the gyroscope measurement noise to 0.01 rad/s, and the standard deviation of the lidar range measurements to 0.02m. The accelerometers were given a constant bias of 0.05 m/s^2, and the gyroscopes were given a constant bias of 0.05 rad/s.

We compare the performance of our lidar-inertial odometry against the baseline in Table 8.2, where we also show the performance of our approach using lidar only, and the gyroscope and lidar only. We obtained the results by computing the absolute trajectory error between our estimated trajectories and the ground truth using the evo evaluation tool[1]. The results in the table are the overall root mean squared error obtained by concatenating the error from each sequence. The results show that in the low-speed regime, the imu-as-input baseline approach and our imu-as-measurement approach based on the Singer prior achieve nearly identical results. This is unsurprising as it appears to replicate the results from Section 8.2.4. Interestingly, our lidar-only approach performs the best on the slow regime. However, in the medium and fast regime, the advantage of our lidar-inertial approach becomes apparent. In the medium regime, the baseline imu-as-input approach begins to break down. This is possibly due to the fact that the motion is no longer approximately constant acceleration and constant angular rate. On the other hand, our lidar-inertial approach performs roughly the same in the medium regime. Our lidar-only approach also breaks down in the medium regime. In the fast regime, our lidar-inertial and lidar with gyro approaches achieve respectable results, while the lidar-only approach and the imu-as-input baseline fail.

8.6 Experimental Results

In this section, we provide experimental results on the Newer College Dataset [123]. This dataset features a 64-beam Ouster lidar and provides the internal IMU measurements of the Ouster lidar.

[1] `https://github.com/MichaelGrupp/evo`

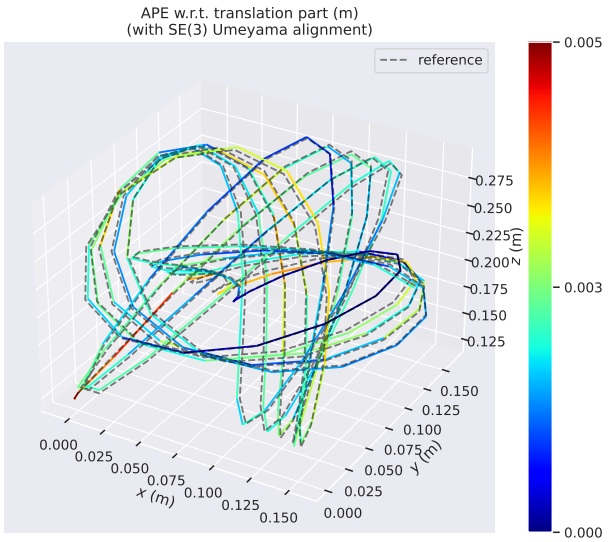


Figure 8.11: This figure depicts the results of our lidar-inertial simulation where the ground truth position (dashed line) is compared to the position estimated by Singer-LIO coloured by the absolute position error. This trajectory is an example of one of the slow sequences

Ground-truth poses were obtained by matching live lidar poses to a map of the environment created using a survey-grade lidar at several stationary poses. This dataset is somewhat unique because it features several sequences with highly dynamic motions. In Table 8.3, we compare the performance of using our continuous-time Singer prior using lidar only (Singer-LO), lidar and a gyroscope only (Singer-LO + Gyro), and a full lidar-inertial setup including an accelerometer (Singer-LIO). We also compare the performance of our approach to some comparable works in the literature such as CT-ICP [48], a lidar-only approach, FAST-LIO2 [156] and DLIO [44], which can be considered the prior state of the art for this dataset, and SLICT [108] and CLIO [96], which are two continuous-time approaches that use linear interpolation and B-splines, respectively.

Our approach, Singer-LIO, demonstrates the best performance on the 01-Short and 02-Long sequences and also demonstrates the best overall performance. Interestingly, the sequences in which we expected the IMU to make the most difference were 05-Quad w/ Dynamics and 06-Dynamic Spinning due to their dynamic motions. However, we observe that in these sequences, our lidar-only approach performs similarly or even better, replicating the results of our lidar-inertial simulation. It appears that, in this dataset, the addition of an IMU mainly helps in areas with geometric degeneracies rather than the areas with dynamic motions. Sequences 05-Quad w/ Dynamics and 06-Dynamic Spinning are very similar to our lidar-inertial simulation in the slow regime, as they are conducted in a rectangular quad at New College, Oxford. FAST-LIO2 and DLIO can be considered state-of-the-art IMU-as-input approaches, and our approach demonstrates a clear advantage over these methods.

Table 8.3: Newer College dataset results. Root Mean Squared Absolute Trajectory Error (m). Estimated trajectory aligned with ground truth using Umeyama algorithm. * uses loop closures, † results obtained from [44], ‡ uses camera images

Newer College Dataset	01-Short	02-Long	05-Quad w/ Dynamics	06-Dynamic Spinning	07-Parkland Mound	**Overall**
CT-ICP* [48]	0.36					
KISS-ICP† [151]	0.6675	1.5311	0.1040	Failed	0.2027	
FAST-LIO2† [156]	0.3775	0.3324	0.0879	0.0771	0.1483	0.3152
DLIO [44]	0.3606	0.3268	**0.0837**	**0.0612**	**0.1196**	0.3048
SLICT* [108]	0.3843	0.3496	0.1155	0.0844	0.1290	0.3263
CLIO*‡ [96]	0.408	0.381		0.091		
Singer-LO (Ours)	0.4543	Failed	0.1120	0.0804	Failed	
Singer-LO + Gyro (Ours)	0.3044	0.3267	0.1092	0.0818	0.1457	0.2887
Singer-LIO (Ours)	**0.3020**	**0.3186**	0.1091	0.0821	0.1411	**0.2832**

8.7 Conclusions

In this chapter, we compared treating an IMU as an input to a motion model vs. treating it as a measurement of the state. On a 1D simulation problem, we showed that these two approaches performed identically when the data is sampled from either a constant velocity or constant acceleration prior and both methods are trained on a hold-out set. We demonstrated our approach to continuous-time lidar-inertial odometry using the Singer prior, where body-centric acceleration is included in the state. Our simulated environment showed that our lidar-inertial odometry outperformed lidar-only odometry and an IMU-as-input baseline approach. On the Newer College Dataset, we demonstrated state-of-the-art results. There is still plenty of work to be done to treat IMU measurements as measurements of the state. Similar to non-uniform B-splines, it would be interesting to investigate a setup where the parameters of the Singer prior are adjusted on the fly to adjust between smooth vs. highly dynamic motion periods. When the IMU is treated as a measurement of the state, this allows us now to incorporate exogenous control inputs into our Gaussian process motion prior. This could be a promising area of research for estimating the state of drones where the torque commanded to the motors is often known. Our approach to combining multiple asynchronous high-rate sensors may prove beneficial in other sensor configurations, such as multiple asynchronous IMUs. Although we achieved state-of-the-art performance using our Singer-LIO, the algorithm is not quite real-time. In addition, the lidar-only version of our approach, Singer-LO, is somewhat brittle. These shortcomings informed the design of our lidar-inertial and radar-inertial odometry in the following chapter, where we instead employ a white-noise-on-acceleration motion prior and choose to preintegrate accelerometer measurements to form relative body-centric velocity factors.

Chapter 9

Continuous-Time Radar-Inertial Odometry

In Chapter 7, we showed that lidar localization can still function under moderate levels of precipitation [30]. Nevertheless, radar may perform better under more extreme weather conditions. Furthermore, radar-based localization may be valuable as a redundant backup system in safety-critical applications. Our goal in this chapter is to tackle the problem of aggressive motion using continuous-time state estimation and an inertial measurement unit (IMU). In addition, we apply our approach to radar-inertial odometry to address adverse weather conditions. Our secondary goal is to reduce the performance gap between radar and lidar odometry by incorporating an IMU.

Inertial measurement units play an important role in many robotic estimation systems and are often fused with low-rate exteroceptive measurements from sensors such as a camera. The addition of an IMU encourages the estimated trajectory to be locally smooth and helps the overall pipeline to be more robust to temporary failures of the exteroceptive measurements. Ordinarily, in discrete-time batch state estimation, one would need to estimate the state at each measurement time. However, due to the high rate of IMUs (100Hz - 1000Hz), the number of measurement times can become quite large, and consequently, the computational requirements can become too expensive for real-time operation.

To address this problem, Lupton and Sukkarieh [95] proposed to preintegrate IMU measurements between pairs of consecutive camera measurements to combine them into a single relative motion factor. This significantly improves runtime since we only need to estimate the state at each camera measurement. Forster et al. [55] then showed how to perform on-manifold preintegration within the space of 3D rotations, $SO(3)$. Recently, Brossard et al. [26] demonstrated how to perform on-manifold preintegration within the space of extended poses, $SE_2(3)$, which captures the uncertainty resulting from IMU measurements more consistently.

Classical preintegration was designed to address the problem of combining a low-rate sensor with a high-rate inertial sensor. However, in some cases, we must work with multiple high-rate sensors such as a lidar or radar and an IMU. While lidar sensors typically spin at around 10Hz, the laser measurements are acquired at a much higher rate, on the order of 10kHz. At this rotational rate (10Hz), the robot's motion causes the pointclouds to become distorted due to the scanning-while-moving nature of the sensor. Our previous work addressed this motion-distortion effect using

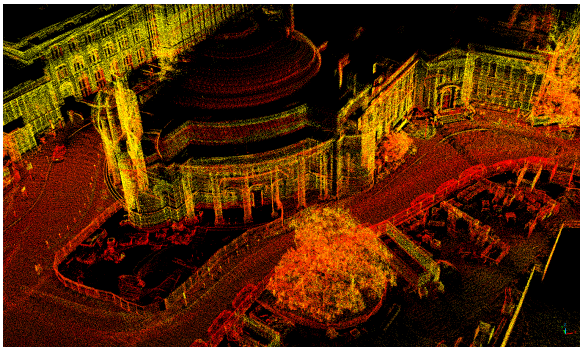


Figure 9.1: A lidar map generated of the University of Toronto obtained during a sequence of the Boreas dataset. This high-quality map is generated as a byproduct of our odometry pipeline. The pointcloud is coloured by intensity.

continuous-time point-to-point factors and a Gaussian process motion prior [30]. This allowed us to estimate the sensor's pose and body-centric velocity while undistorting the data. Given this previously demonstrated success, we are motivated to apply our continuous-time techniques to radar-inertial and lidar-inertial odometry. It is possible to employ a constant-velocity assumption [151] when the motion of a robot is relatively smooth, such as in the case of heavy ground robots. However, continuous-time approaches present considerable advantages when working with highly dynamic motions, such as in the case of drones or walking robots. Using the Newer College Dataset, we will demonstrate this in the experimental results section.

We aim to treat the lidar, radar, and IMU all as high-rate measurements using continuous-time estimation. We are thus faced with the choice of picking a suitable Gaussian process motion prior. In this chapter, we use a white-noise-on-acceleration motion prior [10]. In this setup, angular velocity and body-centric linear velocity are a part of our state. As such, we treat the gyroscope as a direct measurement of the state rather than preintegrating it. However, since acceleration is not a part of the state, we still need to preintegrate the accelerometer measurements to form relative velocity factors. We only need to integrate the accelerometer measurements once, as we rely on the Gaussian process estimation framework to do the remaining integration into position. Other motion prior factors are also possible, such as white-noise-on-jerk [143], or the Singer prior [154], both of which include body-centric acceleration in the state. Our experiments showed that including these higher-order derivatives in the state sometimes improved performance. However, the effect was not consistent across datasets. Furthermore, we observed that including acceleration in the state made the overall pipeline less reliable; thus, we opted not to include it.

Another work that employed Gaussian processes in continuous-time state estimation is that of Le Gentil and Vidal-Calleja [87]. They proposed to model the state using six independent Gaussian processes, three for angular velocity and three for linear acceleration. They estimate the state of the Gaussian processes at several inducing points given the measurements and then analytically integrate these Gaussian processes to form relative motion factors on position, velocity, and rotation in a manner similar to classical preintegration. They use a squared exponential kernel, which misses out on the potential benefits of using a sparse kernel. As a result, the computational complexity of their approach is $O(J^3 + J^2N)$ while ours is only $O(J + N)$ where J is the number of inducing points and N is the number of query times. In our approach, all six degrees of freedom are coupled

through the motion prior. Consequently, our approach has the potential to provide better calibrated covariance estimates. Their exponential kernel results in a fully connected factor graph, so dividing a longer trajectory into a sequence of local chunks effectively drops connections from the graph. In our approach, dividing a longer trajectory into a sequence of local GPs is less of an approximation due to the Markovian nature of the state that results from a sparse kernel. In addition, our approach can still perform continuous-time lidar odometry during IMU measurement dropout by falling back on the Gaussian process motion prior. Our approach is tightly coupled since we directly include IMU measurements in the pointcloud alignment optimization. In contrast, their approach uses the IMU to undistort the scans before the alignment optimization. Another important difference is how we compensate for motion distortion. [87] undistorts pointclouds using the upsampled preintegrated IMU measurements, whereas our approach uses the posterior of our continuous-time trajectory, which includes both IMU and lidar measurements. In Figure 9.1, we provide a qualitative example of a lidar map generated using our approach.

Finally, it should be noted that our work focuses on the back-end continuous-time state estimation and is compatible with other works that focus on the front-end pointcloud preprocessing, submap keyframing strategy, and efficient map storage improvements [156, 44]. Also, our framework supports adding additional continuous-time measurement factors such as wheel odometry and Doppler velocity measurements [155]. We provide experimental results in the autonomous driving domain and using a hand-held sensor mast demonstrating that our approach is generalizable to different domains.

9.1 Related Work

Radar-inertial odometry literature tends to focus on either low-cost consumer-grade radar for resource-constrained applications such as UAVs or automotive-grade radar for large-scale outdoor applications. Examples of prior work using consumer-grade radar include [82, 8, 115, 52, 99, 100, 42, 72]. Examples of prior work using automotive radar include [106, 164, 84]. The work closest to ours is that of Ng et al. [106], which demonstrated continuous-time radar-inertial odometry using four automotive radars. In our approach, we use a Gaussian process motion prior to enable continuous-time trajectory estimation whereas their approach uses cubic B-splines. Another important difference is that our approach uses continuous-time point-to-plane factors. In contrast, their approach uses only the Doppler velocity measurements from each sensor in conjunction with an IMU. The radar used in this thesis does not currently support Doppler velocity measurements. However, we have previously shown that our continuous-time motion prior supports incorporating these measurement factors when they are available [155]. To our knowledge, our work is the first to demonstrate radar-inertial odometry using a spinning mechanical radar. Previous works have used a combination of wheel odometry and single-axis gyroscopes [128, 103, 150] to compensate for motion distortion and as a prediction step in a filter. In contrast, our work fuses both gyroscope and accelerometer measurements with point-to-point factors using our continuous-time framework.

The radar and lidar sensors used in this thesis rely on mechanical actuation. As a consequence, these sensors suffer from a motion-distortion effect while moving. In addition, it is difficult to combine these sensors with asynchronous IMU measurements without resorting to ad-hoc interpolation schemes. These challenges motivate the use of continuous-time trajectory estimation. Figure 9.2 illustrates the high-rate and asynchronous nature of the sensor measurements.

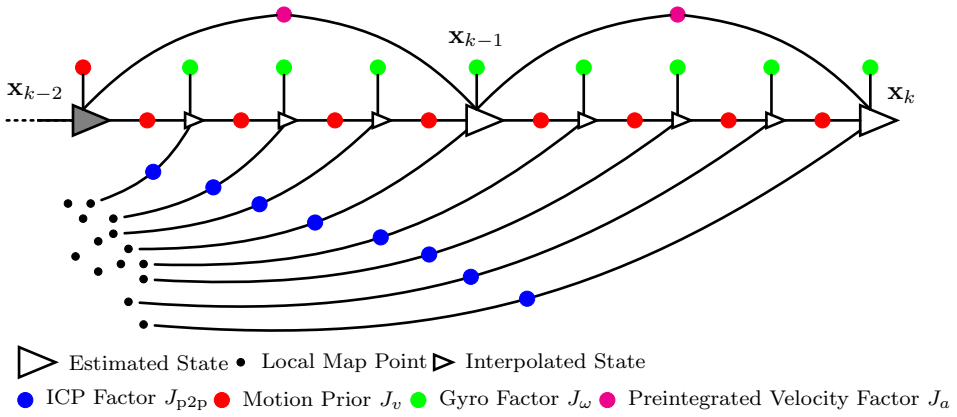


Figure 9.3: This figure depicts a factor graph of our sliding window lidar-inertial odometry using a continuous-time motion prior. The larger triangles represent the estimation times that form our sliding window. The state $\mathbf{x}(t) = \{\mathbf{T}(t), \boldsymbol{\varpi}(t), \mathbf{b}(t)\}$ includes the pose $\mathbf{T}(t)$, the body-centric velocity $\boldsymbol{\varpi}(t)$, and the IMU biases $\mathbf{b}(t)$. The grey-shaded state $\mathbf{x}_{k-2}$ is next to be marginalized and is held fixed during the optimization of the current window. The smaller triangles are interpolated states that we do not directly estimate during optimization. Instead, we construct continuous-time measurement factors using the posterior Gaussian process interpolation formula in Equation 3.31. The ICP measurement times and gyroscope measurement times may be asynchronous. The preintegrated velocity factors do not need to align with the estimated state times and could be between two interpolated states. We include a unary prior on $\mathbf{x}_{k-2}$ to denote the prior information from the sliding window filter.

9.2 Radar-Inertial and Lidar-Inertial Odometry

This section describes our lidar-inertial odometry, which is implemented as sliding-window batch trajectory estimation. The factor graph corresponding to our approach is depicted in Figure 9.3. The state $\mathbf{x}(t) = \{\mathbf{T}(t), \boldsymbol{\varpi}(t), \mathbf{b}(t)\}$ consists of the $SE(3)$ pose $\mathbf{T}_{vi}(t)$, the body-centric velocity $\boldsymbol{\varpi}_v^{vi}(t)$, and the IMU biases $\mathbf{b}(t)$. We use a white-noise-on-acceleration prior, as defined in (3.17). Our IMU measurement model is given in (8.22). Due to angular velocity being a part of the state,

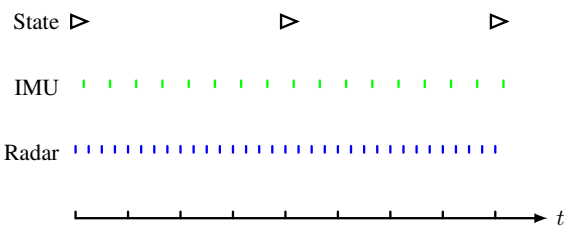


Figure 9.2: This figure illustrates our asynchronous sensor timing where states are estimated for each scan obtained by the radar. Our radar outputs measurements at 1600Hz while our lidar outputs unique timestamps at roughly 40kHz and our IMU outputs measurements at 200Hz.

the associated gyroscope error function is straightforward:

$$J_{\omega,\ell} = \frac{1}{2}\mathbf{e}_{\omega,\ell}^T \mathbf{R}_\omega^{-1} \mathbf{e}_{\omega,\ell}, \tag{9.1a}$$

$$\mathbf{e}_{\omega,\ell} = \tilde{\boldsymbol{\omega}}_\ell - \boldsymbol{\omega}(\tau_\ell) - \mathbf{b}_\omega(\tau_\ell). \tag{9.1b}$$

We preintegrate accelerometer measurements over a short temporal window $t_k \leq \tau_1 < \cdots < \tau_N < t_{k+1}$ to form a relative velocity factor,

$$\Delta\boldsymbol{\nu}(t_{k+1}, \tau_1) = \sum_{n=1}^{N} \big(\tilde{\mathbf{a}}_n + \mathbf{C}_{vi}(\tau_n)\mathbf{g}_i - \mathbf{b}_a(\tau_n)\big)\Delta t_n, \tag{9.2}$$

where the associated factor is given by

$$J_{a,k} = \frac{1}{2}\mathbf{e}_{a,k}^T \mathbf{R}_a(t_{k+1}, \tau_1)^{-1} \mathbf{e}_{a,k}, \tag{9.3a}$$

$$\mathbf{e}_{a,k} = \boldsymbol{\nu}(t_{k+1}) - \boldsymbol{\nu}(\tau_1) - \Delta\boldsymbol{\nu}(t_{k+1}, \tau_1). \tag{9.3b}$$

The covariance associated with the preintegrated velocity factor is $\mathbf{R}_a(t_{k+1}, \tau_1) = \sum_n \mathbf{R}_a \Delta t_n^2$. In error functions (9.1b) (9.3b), we use a continuous-time interpolation of the state. We use the posterior GP interpolation formula (3.31) to interpolate for the measurement times. Interpolating the state at a given measurement time effectively converts a unary measurement factor into a binary factor between the two bracketing estimation times. For example, a first-order linearization of the gyroscope error is given by

$$\mathbf{e}_{\omega,\ell} \approx \bar{\mathbf{e}}_{\omega,\ell} + \frac{\partial \mathbf{e}_{\omega,\ell}}{\partial \delta\boldsymbol{\omega}(\tau_\ell)} \left(\frac{\partial \delta\boldsymbol{\omega}(\tau_\ell)}{\partial \delta\mathbf{x}_k} \delta\mathbf{x}_k + \frac{\partial \delta\boldsymbol{\omega}(\tau_\ell)}{\partial \delta\mathbf{x}_{k+1}} \delta\mathbf{x}_{k+1} \right), \tag{9.4}$$

where we have included the Jacobians of the perturbation at the interpolated time τ_ℓ with respect to the perturbations at the bracketing estimation times (t_k, t_{k+1}). We provide these interpolation Jacobians in the Appendix. Using the posterior interpolation formula in this way is an approximation as this is not equivalent to marginalizing out the measurement times. However, we have found this approximation to be fast and to work well in practice. The computational cost of our approach scales linearly with both the number of estimation times and the number of measurement times. This is different from the approach of Le Gentil and Vidal-Calleja [87], where the computational cost of preintegration scales with the cube of the number of estimation times in the preintegration window.

We use point-to-plane factors similar to iterative closest point (ICP). The associated error function is

$$J_{\text{p2p},j} = \mathbf{e}_{\text{p2p},j}^T \mathbf{R}_{\text{p2p},j}^{-1} \mathbf{e}_{\text{p2p},j}, \tag{9.5a}$$

$$\mathbf{e}_{\text{p2p},j} = \alpha_j \mathbf{n}_j^T \mathbf{D}(\mathbf{p}_j - \mathbf{T}_{vi}(\tau_j)^{-1} \mathbf{T}_{vs} \mathbf{q}_j), \tag{9.5b}$$

where $\mathbf{q}_j$ is the query point, $\mathbf{p}_j$ is the matched point in the local map, $\mathbf{n}_j$ is an estimate of the surface normal at $\mathbf{p}_j$ given neighboring points in the map, $\mathbf{D}$ is a constant matrix removing the homogeneous component, $\mathbf{T}_{vs}$ is an extrinsic calibration between the lidar frame and the vehicle frame, and $\alpha_j = (\sigma_2 - \sigma_3)/\sigma_1$ [48, 49] is a heuristic weight to favour planar neighborhoods. Query

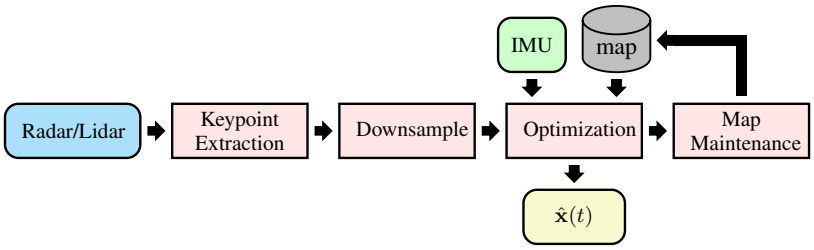


Figure 9.4: This figure depicts the simple architecture diagram for STEAM-LIO. In the radar-based pipelines, keypoints are first extracted using a constant false alarm rate (CFAR) detector. In the lidar-based pipelines, we randomly shuffle the order of the points and then downsample using a coarse voxel grid. At the optimization stage, we alternate between finding correspondences between the live undistorted pointcloud and the sliding local map, and estimating the trajectory using sliding window batch trajectory estimation. The inner loop of the optimization stage involves optimizing a nonlinear least-squares problem with Gauss-Newton. At the map maintenance stage, we add registered points to the sliding local map, and optionally cull voxels that have been unobserved for several consecutive frames.

points are matched to a sliding local voxel map centred on the current estimate of the robot's position. Once a voxel has reached its maximum number of allocated points, new points are not added. This helps to keep the state estimate from exhibiting a random walk while stationary by keeping the map fixed. Depending on the dataset, we clear voxels in the map if they have not been observed for approximately one second. We found this to be important in the Boreas dataset, especially for sequences with snowstorms where erroneous snow detections would accumulate in the map and eventually cause ICP to fail by drastically increasing the number of outlier points. Interestingly, the addition of IMU measurements made our lidar-inertial pipeline more robust to this accumulation of noise.

Our Gaussian process prior introduces a set of motion prior factors between estimation times, which penalizes the state for deviating from a constant velocity. These motion prior factors are defined in (3.25a) (3.25b). We also include motion prior factors for the IMU biases,

$$J_{v,b,k} = \frac{1}{2}\mathbf{e}_{v,b,k}^T \mathbf{Q}_{b,k}^{-1} \mathbf{e}_{v,b,k}, \tag{9.6a}$$

$$\mathbf{e}_{v,b,k} = \mathbf{b}(t_{k+1}) - \mathbf{b}(t_k), \tag{9.6b}$$

where $\mathbf{Q}_{b,k} = \boldsymbol{Q}_b \Delta t_k$ is the covariance resulting from a white-noise-on-velocity motion prior, and $\boldsymbol{Q}_b$ is the associated power spectral density matrix. The objective function that we seek to minimize is then

$$J = \sum_k J_{v,k} + \sum_j J_{\mathrm{p2p},j} + \sum_\ell J_{\omega,\ell} + \sum_k J_{a,k}. \tag{9.7}$$

We solve this nonlinear least-squares problem for the optimal perturbation to the state using Gauss-Newton. Once the solver has converged, we update the pointcloud correspondences and iterate this two-step process to convergence. In practice, we typically limit the maximum number of inner-loop Gauss-Newton iterations to 5, and the number of outer-loop iterations to 10 to enable real-time operation.

Algorithm 1 STEAM-LIO

Input: map: $\{\mathbf{p}_i\}$, new frame: $\{\mathbf{q}_j, \tau_j\}$, IMU: $\{\tilde{\boldsymbol{\omega}}_\ell, \tilde{\mathbf{a}}_\ell\}$, $\hat{\mathbf{x}}(t), \mathbf{A}, \mathbf{c}$ from previous iteration

Output: $\hat{\mathbf{x}}(t) = \{\hat{\mathbf{T}}(t), \hat{\boldsymbol{\varpi}}(t), \hat{\mathbf{b}}(t)\}$ where $t \in [t_{k-2}, t_k]$

$\mathbf{T}(t_k) \leftarrow \exp(\Delta t_k \boldsymbol{\varpi}(t_{k-1})^\wedge)\mathbf{T}(t_{k-1})$
$\boldsymbol{\varpi}(t_k) \leftarrow \boldsymbol{\varpi}(t_{k-1})$, $\mathbf{b}(t_k) \leftarrow \mathbf{b}(t_{k-1})$
$\hat{\mathbf{x}}(t), \mathbf{A}, \mathbf{c} \leftarrow \text{SlideWindow}(\hat{\mathbf{x}}(t), \mathbf{A}, \mathbf{c}, \mathbf{x}_k)$
$\{\mathbf{q}_j, \tau_j\} \leftarrow \text{Downsample}(\{\mathbf{q}_j, \tau_j\})$
$x \leftarrow 0$, $||\Delta\mathbf{x}|| \leftarrow \infty$
while $||\Delta\mathbf{x}|| > T_{\text{outer}} \wedge x < N_{\text{outer}}$ **do**
    $\{\bar{\mathbf{q}}_j\} \leftarrow \text{Undistort}(\{\mathbf{q}_j, \tau_j\}, \hat{\mathbf{x}}(t))$
    $\{\mathbf{p}_j, \mathbf{n}_j\} \leftarrow \text{Matching}(\{\mathbf{p}_i\}, \{\bar{\mathbf{q}}_j\})$
    $J \leftarrow J_v(\mathbf{x}_{k-1}, \mathbf{x}_k) + J_\omega(\mathbf{x}(t), \{\tilde{\boldsymbol{\omega}}_\ell\})$
    $+ J_a(\mathbf{x}(t), \{\tilde{\mathbf{a}}_\ell\}) + J_{\text{p2p}}(\mathbf{x}(t), \{\mathbf{p}_j, \mathbf{n}_j, \mathbf{q}_j, \tau_j\})$
    $y \leftarrow 0, ||\delta\mathbf{x}|| \leftarrow \infty, \Delta J \leftarrow \infty, \mathbf{x}_{\text{prev}} \leftarrow \hat{\mathbf{x}}(t)$
    while $||\delta\mathbf{x}|| > T_{\text{inner}} \wedge \Delta J > \delta J \wedge y < N_{\text{inner}}$ **do**
        $\mathbf{A}, \mathbf{c} \leftarrow \text{buildAndUpdateGN}(\mathbf{A}, \mathbf{c}, J, \hat{\mathbf{x}}(t))$
        $\delta\mathbf{x} \leftarrow \text{CholeskySolve}(\mathbf{A}, \mathbf{c})$
        $\hat{\mathbf{x}}(t) \leftarrow \text{UpdateState}(\hat{\mathbf{x}}(t), \delta\mathbf{x})$
        $J, \Delta J \leftarrow \text{UpdateCost}(\hat{\mathbf{x}}(t), J)$
        $y \leftarrow y + 1$
    end while
    $||\Delta\mathbf{x}|| \leftarrow \text{Dist}(\mathbf{x}_{\text{prev}}, \hat{\mathbf{x}}(t))$
    $x \leftarrow x + 1$
end while
$\{\mathbf{p}_i\} \leftarrow \text{UpdateMap}(\{\mathbf{p}_i\}, \{\mathbf{q}_j, \tau_j\}, \hat{\mathbf{x}}(t))$

9.2.1 Implementation Details

Algorithm 1 provides pseudocode for STEAM-LIO at a high level, and Figure 9.4 depicts the software architecture for our approach. We first initialize the new state using constant velocity for a new lidar frame. We then slide the estimation window forward and marginalize out states that are no longer involved in the current optimization problem. For lidar odometry, a coarse voxelization of the input pointcloud is then performed where the default is 1.5m. At each iteration of the outer loop, we first undistort the lidar frame using the posterior trajectory estimate of the previous iteration to obtain correspondences between the live frame and the local map. We then build the optimization problem given the set of lidar factors, IMU measurements, and motion prior factors derived from the Gaussian process motion. This optimization problem is then minimized using Gauss-Newton. Finally, the undistorted points are added to the sliding local map. The local map also has a coarse discretization of 1.0m, but we allow up to 20 points in each voxel with a minimum point distance of 0.1m. CT-ICP inspired this voxelization strategy [48].

To achieve real-time performance, we found it necessary to implement timestamp binning, where the original timestamp frequency is downsampled to reduce the number of state interpolations and associated Jacobians that need to be computed. For the KITTI-raw and Newer College Dataset, we downsample lidar timestamps to 5kHz. For the Boreas dataset, we downsample lidar timestamps to 400Hz. We have found that reducing the timestamp frequency in this way can retain most of the benefits of continuous-time state estimation while still operating efficiently.

9.2.2 Gravity Vector Orientation

In our approach, we estimate the orientation of the gravity vector relative to the initial map frame at startup. We do this by first estimating the orientation of the gravity frame using an initial set of accelerometer measurements,

$$J = \sum_{n} \mathbf{e}_{a,n}^{T} \mathbf{R}_{a}^{-1} \mathbf{e}_{a,n} + \ln(\mathbf{C}_{ig})^{\vee^{T}} \check{\mathbf{P}}_{g}^{-1} \ln(\mathbf{C}_{ig})^{\vee} + \mathbf{b}_{a}^{T} \check{\mathbf{P}}_{b}^{-1} \mathbf{b}_{a}, \tag{9.8a}$$

$$\mathbf{e}_{a,n} = \tilde{\mathbf{a}}_{n} - \mathbf{C}_{ig}\mathbf{g} - \mathbf{b}_{a}, \tag{9.8b}$$

where we assume that the robot is stationary at startup, and we impose a weak prior on $\mathbf{C}_{ig}$ to constrain the rotational degree of freedom not observed by the accelerometer measurements. This estimate of the gravity vector orientation then serves as a prior for the gravity vector orientation included in the state at $t = 0$. We hold our estimate of the gravity vector orientation fixed once it has been marginalized from the sliding window.

We experimented with including the gravity vector orientation in the state: $\mathbf{x}(t) = \{\mathbf{T}(t), \boldsymbol{\varpi}(t), \mathbf{b}(t), \mathbf{C}_{ig}(t)\}$. In this case, we include a motion prior factor for the gravity vector orientation,

$$J_{v,g,k} = \frac{1}{2} \mathbf{e}_{v,g,k}^{T} \mathbf{Q}_{g,k}^{-1} \mathbf{e}_{v,g,k}, \tag{9.9a}$$

$$\mathbf{e}_{v,g,k} = \ln(\mathbf{C}_{ig}(t_k)\mathbf{C}_{ig}(t_{k+1})^{-1})^{\vee}, \tag{9.9b}$$

In our experiments, we did not observe any benefit from including the gravity vector orientation in the state. However, some recent work by Nemiroff et al. [105] has shown that it can be beneficial to mapping accuracy in challenging scenarios.

9.2.3 Sliding Window Marginalization

In our approach, we perform sliding window batch trajectory estimation. The length of the sliding window is equivalent to two lidar frames or roughly 200ms. We output the pose in the middle of the newest lidar frame so that the latency is equivalent to competing approaches. In Figure 9.3, the darkly shaded state $\mathbf{x}_{k-2}$ is slated to be marginalized and is held fixed during optimization. However, there are still several continuous-time measurement factors between states $\mathbf{x}_{k-2}$ and $\mathbf{x}_{k-1}$. At each iteration of our Gauss-Newton solver, we first interpolate the state at each measurement time and update the associated measurement Jacobians before marginalizing $\mathbf{x}_{k-2}$. For example,

$$\begin{bmatrix} \mathbf{A}_{k-2,k-2} & \mathbf{A}_{k-1,k-2}^{T} & \\ \mathbf{A}_{k-1,k-2} & \mathbf{A}_{k-1,k-1} & \mathbf{A}_{k,k-1}^{T} \\ & \mathbf{A}_{k,k-1} & \mathbf{A}_{k,k} \end{bmatrix} \begin{bmatrix} \delta\mathbf{x}_{k-2}^{\star} \\ \delta\mathbf{x}_{k-1}^{\star} \\ \delta\mathbf{x}_{k}^{\star} \end{bmatrix} = \begin{bmatrix} \mathbf{c}_{k-2} \\ \mathbf{c}_{k-1} \\ \mathbf{c}_{k} \end{bmatrix} \tag{9.10}$$

becomes

$$\begin{bmatrix} \mathbf{A}_{k-1,k-1} - \mathbf{A}_{k-1,k-2}\mathbf{A}_{k-2,k-2}^{-1}\mathbf{A}_{k-1,k-2}^{T} & \mathbf{A}_{k,k-1}^{T} \\ \mathbf{A}_{k,k-1} & \mathbf{A}_{k,k} \end{bmatrix} \begin{bmatrix} \delta\mathbf{x}_{k-1}^{\star} \\ \delta\mathbf{x}_{k}^{\star} \end{bmatrix} = \begin{bmatrix} \mathbf{c}_{k-1} - \mathbf{A}_{k-1,k-2}\mathbf{A}_{k-2,k-2}^{-1}\mathbf{c}_{k-2} \\ \mathbf{c}_{k} \end{bmatrix}. \tag{9.11}$$

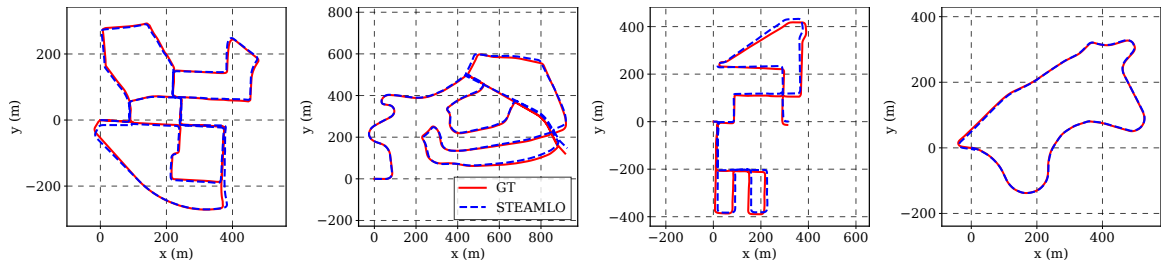


Figure 9.5: Odometry results on the KITTI-raw dataset. Sequences shown from left to right are 00, 02, 07, 08. The ground-truth trajectory is shown as a solid red line, and the STEAM-LO trajectory estimate is shown as a blue dashed line. Note that the estimated trajectory was computed online with a sliding window of 200ms and does not use any loop closures.

9.2.4 Radar-Inertial Odometry

The architecture of our radar-inertial odometry is largely the same as our lidar-inertial odometry. We note the important differences here. The radar we use in this chapter is the Navtech CIR304-H, a mechanical spinning radar providing a 360° horizontal field of view. This sensor is 2D only; as such, we do not estimate the gravity vector's orientation and remove the gravity term from the preintegrated velocity factor in (9.2). The Navtech sensor outputs a raw polar radar image corresponding to a power vs. range spectrum for each scanned azimuth. We use a constant false alarm rate (CFAR) detector with an additional constant threshold tuned to the noise floor of the sensor [7] where we retain the maximum of the left and right subwindows, a variant known as GO-CFAR [125]. The output of CFAR detection is a pointcloud that we register to a sliding local map. When a voxel in the map has not been observed for several consecutive frames (one second), we delete the points in this voxel. Without this map maintenance procedure, we have found that our radar odometry tends to fail due to the significant amount of noise in the radar pointclouds. The Navtech radar scans each azimuth only once, and as such, range measurements are corrupted by a Doppler distortion dependent on the robot's egomotion [28]. The Doppler-compensated point-to-point ICP error for radar odometry is then

$$\mathbf{e}_{\text{p2p},j} = \rho\left(\mathbf{D}\left(\mathbf{p}_j - \mathbf{T}(\tau_j)^{-1}\mathbf{T}_{vs}(\mathbf{q}_j + \Delta\mathbf{q}_j)\right)\right), \tag{9.12a}$$

$$\Delta\mathbf{q}_j = \beta\mathbf{D}^T\mathbf{a}_j\mathbf{a}_j^T\mathbf{D}\mathbf{q}_j^{\odot}\boldsymbol{\mathcal{T}}_{sv}\boldsymbol{\varpi}(\tau_j), \tag{9.12b}$$

and where $\rho(\cdot)$ is a Cauchy robust cost function, $\mathbf{D}$ is a constant matrix that removes the homogeneous component, $\mathbf{p}_j$ is a reference point in the local map, $\mathbf{T}(\tau_j)$ is the continuous-time interpolation of the robot pose, $\mathbf{T}_{vs}$ is an extrinsic transformation between the sensor and vehicle frame, $\mathbf{q}_j$ is the live query point, $\Delta\mathbf{q}_j$ is an additive correction factor to compensate for the Doppler distortion, β is Doppler distortion constant inherent to the sensor [28], $\mathbf{a}_j$ is a 3×1 unit vector in the direction of $\mathbf{q}_j$, the $\odot$ operator swaps the order of the operands associated with the skew-symmetric $\wedge$ operator [16], and $\boldsymbol{\mathcal{T}}_{sv} = \text{Ad}(\mathbf{T}_{sv})$.

Table 9.1: KITTI-raw results (22km / 0.6h): KITTI RTE (%). The average is computed over all segments of all sequences as in [48]. Note that CT-ICP optimizes one lidar frame at a time, while our algorithm optimizes two frames in a sliding window. For a fair comparison, we evaluate our algorithm using the estimated poses at the front of the window (i.e., newest timestamp).

KITTI-raw	00	01	02	03 (N/A)	04	05	06	07	08	09	10	**Overall**	**Seq. Avg.**	ΔT
CT-ICP [48]	0.51	0.81	0.55		0.43	0.27	0.28	0.35	**0.80**	0.47	**0.49**	0.55	0.50	65ms [48]
KISS-ICP [151]	0.51	0.71	0.54		**0.35**	0.31	**0.26**	**0.32**	0.83	0.49	0.58	0.55	0.49	26ms [151]
STEAM-ICP [155]	**0.49**	0.65	**0.50**		0.38	**0.26**	0.28	**0.32**	0.81	**0.46**	0.53	**0.52**	**0.47**	138ms
Constant Velocity	0.60	1.62	0.60		0.36	0.30	0.27	0.37	0.92	0.52	0.90	0.66	0.65	44ms
STEAM-LO (Ours)	**0.49**	**0.63**	0.51		0.38	**0.26**	0.30	0.33	0.84	0.49	**0.49**	0.53	**0.47**	89ms

9.3 Experimental Results

We provide experimental results on three datasets, KITTI-raw [61], the Newer College Dataset (NCD) [123], and the Boreas dataset [31]. KITTI-raw was chosen as it is a popular dataset for benchmarking lidar odometry. The raw version of the dataset contains the motion-distorted point-clouds, whereas the original version is motion-compensated using GPS poses. Since this work aims to demonstrate continuous-time state estimation using motion-distorted sensors, we present results for KITTI-raw and not the original KITTI dataset. The Newer College dataset was chosen as it has become a standard dataset for benchmarking lidar-inertial odometry. The NCD dataset is somewhat unique in that it features several sequences with aggressive high-frequency motions obtained using a handheld sensor mast. These types of trajectories are rarely observed when working with heavy ground robots. Finally, we provide results using the Boreas dataset to demonstrate continuous-time radar inertial odometry and provide a detailed comparison with lidar. We provide average runtime estimates in all experiments using an Intel Xeon CPU E5-2698 v4 with 16 threads.

For the lidar-based pipelines, we use the same parameters for the diagonal of $\boldsymbol{Q}$, the power spectral density matrix, $\text{diag}(\boldsymbol{Q}) = \{50, 50, 50, 5, 5, 5\}$. These parameters were obtained by tuning on a training split of the Boreas dataset and were verified to work well on KITTI-raw and the Newer College Dataset. The diagonal of $\boldsymbol{Q}^{-1}$ can be understood as weighting cost terms on body-centric acceleration. We separately tune the IMU measurement covariances and bias motion priors for Boreas and the Newer College Dataset. We downsample lidar timestamps to 400Hz on the Boreas dataset to achieve real-time performance. However, as we could already run in real-time, we do not downsample timestamps on KITTI-raw or the Newer College Dataset. On the Boreas dataset, we clear voxels that have not been observed for one second. We incrementally build a map on the Newer College Dataset to enable implicit loop closures by revisiting previously mapped areas.

9.3.1 KITTI-Raw Results

The KITTI dataset was collected in Karlsruhe, Germany, using an autonomous driving platform with a 64-beam Velodyne lidar and an OXTS RTK GPS. The dataset was primarily collected in an urban environment with some sequences including a brief highway portion. Table 9.1 shows our quantitative results. We compare ourselves against CT-ICP and KISS-ICP, which represent the state of the art on this dataset. We also compare ourselves against our previously published work, STEAM-ICP [155]. We include two different methods for aggregating the results across sequences. In the *Overall* column, we concatenate the subsequence errors of the KITTI metric and average across these. In the *Sequence Error* column, we simply average the results for each sequence. The

Table 9.2: Newer College dataset results (6km / 1.3h): root mean squared ATE (m). Trajectories are aligned with the ground truth using the Umeyama algorithm. ⋆ uses explicit loop closures, † results obtained from [43], ‡ uses camera.

Newer College Dataset	01-Short	02-Long	05-Quad w/ Dynamics	06-Dynamic Spinning	07-Parkland Mound	ΔT
CT-ICP⋆ [48]	0.36					430ms [48]
KISS-ICP† [151]	0.6675	1.5311	0.1040	Failed	0.2027	167ms†
FAST-LIO2† [156]	0.3775	0.3324	0.0879	0.0771	0.1483	43ms†
DLIO [44]	0.3606	**0.3268**	**0.0837**	**0.0612**	**0.1196**	36ms [44]
SLICT⋆ [108]	0.3843	0.3496	0.1155	0.0844	0.1290	
CLIO⋆‡ [96]	0.408	0.381		0.091		
Constant Velocity	0.8558	2.5792	0.3575	Failed	0.5960	163ms
STEAM-LO (Ours)	0.3398	0.4546	0.1083	0.0802	0.1537	138ms
STEAM-LO + Gyro (Ours)	0.3055	0.3340	0.1090	0.0824	0.1444	76ms
STEAM-LIO (Ours)	**0.3042**	0.3372	0.1086	0.0821	0.1444	74ms

Table 9.3: Boreas Odometry Results (102km / 4.3h): translational drift (%) / rotational drift (deg/100m). The first three columns are evaluated in SE(3) whereas the last four columns are evaluated in SE(2).

Boreas	VTR3-Lidar [30]	STEAM-LO	STEAM-LIO	STEAM-LO(SE2)	VTR3-Radar [30]	STEAM-RO	STEAM-RIO
2020-12-04	0.49 / 0.14	0.41 / 0.13	0.39 / 0.13	0.13 / 0.05	1.92 / 0.53	1.43 / 0.41	0.93 / 0.26
2021-01-26	0.51 / 0.16	0.62 / 0.21	0.53 / 0.18	0.30 / 0.11	2.27 / 0.66	1.10 / 0.33	0.61 / 0.18
2021-02-09	0.49 / 0.14	0.38 / 0.13	0.38 / 0.13	0.14 / 0.06	1.94 / 0.59	1.27 / 0.38	0.63 / 0.20
2021-03-09	0.57 / 0.17	0.47 / 0.15	0.46 / 0.15	0.13 / 0.05	2.00 / 0.59	1.24 / 0.35	0.71 / 0.19
2020-04-22	0.49 / 0.15	0.39 / 0.13	0.39 / 0.13	0.13 / 0.05	2.56 / 0.63	1.48 / 0.41	0.99 / 0.27
2021-06-29-18	0.58 / 0.17	0.48 / 0.16	0.48 / 0.16	0.14 / 0.06	1.86 / 0.56	1.55 / 0.46	1.04 / 0.29
2021-06-29-20	0.62 / 0.18	0.52 / 0.17	0.52 / 0.17	0.16 / 0.06	1.94 / 0.59	1.70 / 0.48	0.96 / 0.26
2021-09-08	0.57 / 0.17	0.47 / 0.16	0.47 / 0.16	0.16 / 0.06	1.88 / 0.57	2.01 / 0.59	1.22 / 0.35
2021-09-09	0.63 / 0.19	0.52 / 0.18	0.55 / 0.19	0.20 / 0.06	1.98 / 0.60	2.16 / 0.64	1.19 / 0.33
2021-10-05	0.59 / 0.17	0.50 / 0.16	0.49 / 0.16	0.16 / 0.06	2.87 / 0.78	2.27 / 0.63	1.01 / 0.28
2021-10-26	0.48 / 0.14	0.40 / 0.14	0.38 / 0.13	0.14 / 0.06	1.89 / 0.53	1.88 / 0.53	0.97 / 0.27
2021-11-06	0.50 / 0.15	0.40 / 0.14	0.41 / 0.14	0.15 / 0.06	1.24 / 0.34	1.86 / 0.54	1.07 / 0.29
2021-11-28	0.46 / 0.14	0.41 / 0.14	0.37 / 0.13	0.15 / 0.06	1.24 / 0.38	1.95 / 0.57	1.04 / 0.29
Seq. Avg.	0.54 / 0.16	0.46 / 0.15	0.45 / 0.15	0.16 / 0.06	2.02 / 0.58	1.68 / 0.49	0.95 / 0.27
ΔT	250ms	88ms	97ms	88ms	75ms	115ms	139ms

results show that our translational drift is slightly lower than CT-ICP and KISS-ICP but not quite as good as STEAM-ICP. However, our approach is demonstrably real-time, whereas STEAM-ICP is not. Figure 9.5 provides some qualitative examples of the trajectories estimated by our lidar odometry. This dataset does not provide raw IMU measurements, so we cannot use it to benchmark our lidar-inertial odometry. The main purpose of testing on this dataset is to show that, without an IMU, our implementation of continuous-time lidar odometry is competitive with the state of the art.

9.3.2 Newer College Dataset Results

The Newer College Dataset was collected using a handheld sensor mast at the University of Oxford. The dataset includes approximately 6km or 1.3h of data. The sensor suite includes a 64-beam Ouster lidar and an Intel Realsense camera. Both the Ouster lidar and the Intel camera have internal IMUs. We use the 100Hz IMU measurements provided by the Ouster to avoid potential time synchronization problems between the lidar and the IMU. Ground truth for this dataset was obtained by registering lidar scans in the dataset to a surveyed lidar map of the university campus. Table 9.2 shows our quantitative results for this dataset. Again, we include CT-ICP and KISS-ICP since they are well-

known lidar odometry approaches. DLIO [44] and FAST-LIO2 [156] are also included as these approaches currently represent the state of the art for lidar-inertial odometry. Finally, we include SLICT [108] and CLIO [96] as these are continuous-time approaches that use linear interpolation and B-splines, respectively. It is challenging to directly compare to other methods due to significant difference in front-end preprocessing and map storage strategies. Nevertheless, we show that our approach is competitive with the state of the art while still being real-time capable. As of writing, ours is the only continuous-time lidar-inertial odometry with confirmed real-time performance on the Newer College Dataset. SLICT quotes their average runtime as 205ms on the NTU Viral dataset [107] using two 16-beam lidars, and CLIO quotes their runtime as being 218s for a 397s sequence using a single 16-beam lidar. Note that Table 9.2 contains originally published results, except for the results provided for FAST-LIO2 and KISS-ICP where the results were obtained from [43].

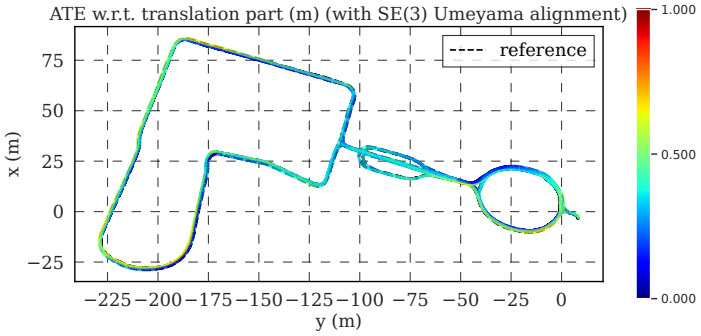


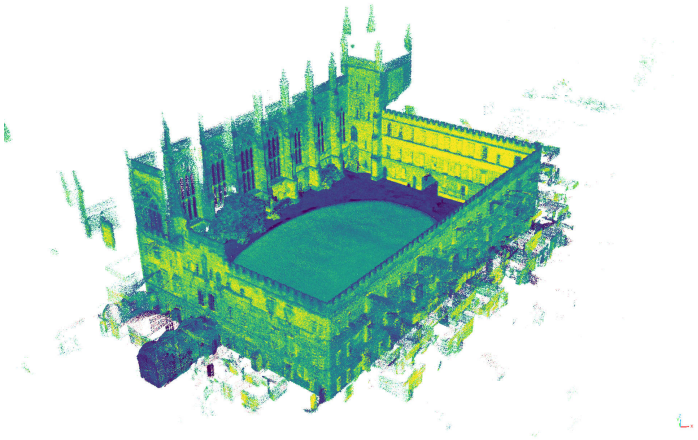
Figure 9.6: This figure depicts the trajectory estimated by STEAM-LIO during the long sequence of the Newer College Dataset. The trajectory is coloured according to the absolute trajectory error when compared against the ground truth. The estimated trajectory was aligned with the ground truth using the Umeyama algorithm [147].

Interestingly, the sequences with the most aggressive motions (05, 06) displayed the smallest differences between our continuous-time lidar-only and lidar-inertial odometry. We postulate that this was due to these sequences being collected in a rectangular quad (shown in Figure 9.7) with plenty of geometric features for point-to-plane ICP. In this dataset, adding an IMU seems to have the most noticeable improvement in sequences where brief periods lack sufficient geometric features. The majority of the lidar-inertial performance improvement seems to come from using the gyroscope with only a minor additional improvement when the accelerometer is included.

Our lidar-only approach, STEAM-LO, achieves better results on sequence 01-Short than all of the previous lidar-inertial methods. Furthermore, STEAM-LO performs the best out of the lidar-only approaches on sequence 06-Dynamic Spinning, where KISS-ICP and our constant-velocity baseline fail due to the rapid rotations observed in this sequence. It can be seen in Table 9.4 that our continuous-time approach, STEAM-LO, significantly outperforms KISS-ICP, which relies on a constant-velocity assumption. Our lidar-inertial approach, STEAM-LIO, performs best on sequence 01-Short while remaining competitive with the other lidar-inertial approaches on the other sequences. Suppose we compute an overall absolute trajectory error for the entire Newer College Dataset by concatenating the squared errors across all timestamps of all sequences. In that case, our approach (0.2946m) outperforms FAST-LIO2 (0.3152m) and DLIO (0.3048m). The performance of our approach on the Newer College Dataset highlights the value of our continuous-time technique.

In Figure 9.6, we provide a qualitative example of the trajectory estimated by our approach. In this case, the trajectory is coloured by the absolute trajectory error (ATE). The estimated trajectory is first aligned with the ground truth using the Umeyama algorithm [147] before computing the ATE as defined in [139]. Even though we do not use explicit loop closure factors, we rely on implicitly closing the loop when we revisit previously mapped areas. This allows us to achieve a low ATE for an odometry method. Usually, ATE is used to benchmark SLAM approaches and not odometry.

(a) Panoramic image of the courtyard at New College, Oxford



(b) Lidar map of the courtyard coloured by reflectivity

Figure 9.7: In this figure, we provide a qualitative example of the map produced by our lidar-inertial odometry using the first 150 seconds of the "quad with dynamics" sequence from the Newer College Dataset [123]. In order to produce this figure, we adjusted configuration parameters to produce a denser map.

We provide another qualitative example of our lidar-inertial odometry in Figure 9.7, where we plot the lidar map generated by our approach alongside a panoramic image of the courtyard of New College, Oxford. In this case, the pointcloud is coloured using the Ouster reflectivity. We used a finer voxelization to produce a denser map here. This map was produced using the *Quad with Dynamics* sequence from the Newer College Dataset, which features dynamic swinging motions of the sensor mast. Even with this dynamic motion, we can produce a high-quality map.

9.3.3 Ablation Study

In addition to the continuous-time odometry methods presented in this work, we present baseline results using a constant-velocity assumption. We approximate the body-centric velocity using the

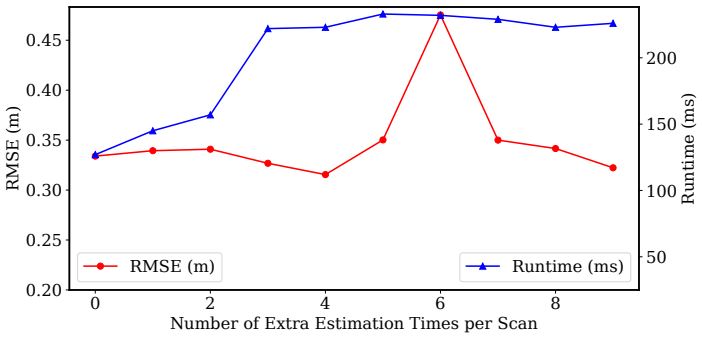


Figure 9.8: Root mean squared error vs. the number of estimation times per lidar scan for STEAM-LO for sequence 01-Short of the Newer College Dataset.

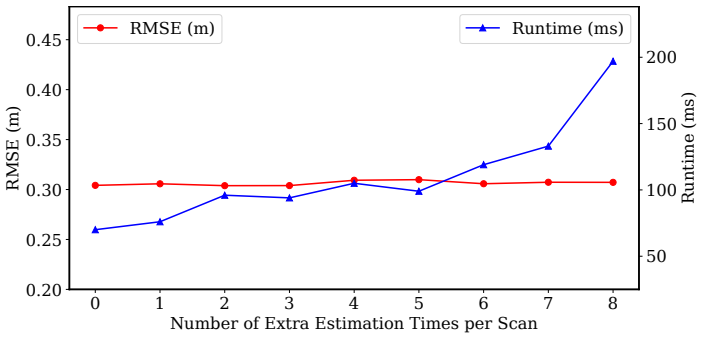


Figure 9.9: Root mean squared error vs. the number of estimation times per lidar scan for STEAM-LIO for sequence 01-Short of the Newer College Dataset.

following formula, which includes the poses at the two previous timesteps,

$$\check{\boldsymbol{\varpi}}_k \approx \frac{1}{\Delta t_{k-1}} \ln\left(\mathbf{T}_{k-1}\mathbf{T}_{k-2}^{-1}\right)^{\vee}. \tag{9.13}$$

Using this prediction of the velocity, we can deskew the pointcloud using the following formula,

$$\tilde{\mathbf{q}}_j := \exp((t_k - \tau_j)\check{\boldsymbol{\varpi}}_k^{\wedge})\mathbf{T}_{vs}\mathbf{q}_j. \tag{9.14}$$

We then reformulate our point-to-plane error function as

$$\mathbf{e}_{\mathrm{p2p},j} = \alpha_j \mathbf{n}_j^T \mathbf{D}(\mathbf{p}_j - \mathbf{T}_k^{-1}\tilde{\mathbf{q}}_j), \tag{9.15}$$

where we minimize a cost function including these point-to-plane factors using our nonlinear least-squares solver without including any explicit prior on $\mathbf{T}_k$. We refer to this approach as *Constant Velocity* in Table 9.1 and Table 9.2. On the KITTI-raw dataset, the constant-velocity approach achieves respectable results but is not quite competitive with the state of the art. However, the approach is computationally efficient. The gap between the constant-velocity approach and our continuous-time approach is much more apparent on the Newer College Dataset. Similar to KISS-ICP, our constant-velocity baseline fails on the Dynamic Spinning sequence.

Here, we analyze the effect of varying the number of additional evenly spaced estimation times for each lidar scan. The default, zero, refers to having an estimation time at the beginning and end

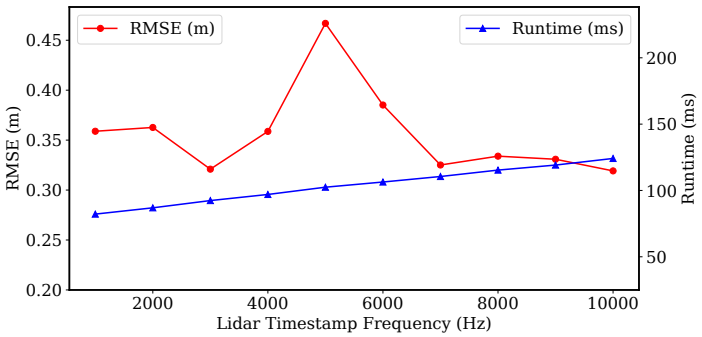


Figure 9.10: Root mean squared error (RMSE) vs. the lidar timestamp frequency for STEAM-LO for sequence 01-Short of the Newer College Dataset.

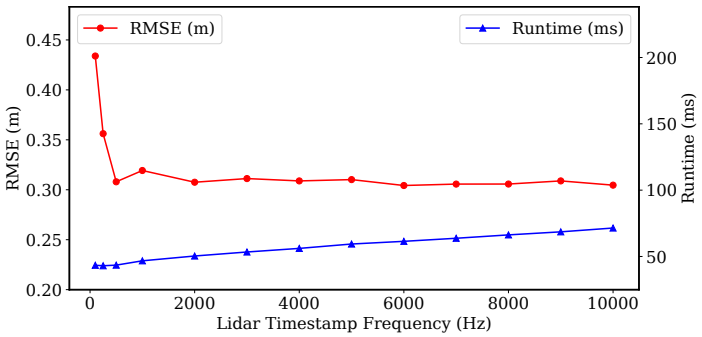


Figure 9.11: Root mean squared error (RMSE) vs. the lidar timestamp frequency for STEAM-LIO for sequence 01-Short of the Newer College Dataset.

of each scan. As shown in Figure 9.8, STEAM-LO's performance improves slightly when increasing the number of extra estimation times to four. However, the improvement does not continue when the number of estimation times is increased further. In addition, the runtime increases substantially by doing so. For STEAM-LIO, the performance is relatively flat when increasing the number of estimation times, as seen in Figure 9.9. Similar experiments varying the number of estimation times or basis functions were previously presented in [57, 75].

We also analyze the effect of downsampling the number of unique lidar timestamps. The number of points is unchanged, but the associated timestamps are rounded to reduce the effective timestamp frequency. This requires less continuous-time interpolations of the state, and fewer Jacobians need to be computed during optimization. Overall, the effect is that the continuous-time deskewing is made coarser, and the required runtime is reduced.

In Figure 9.10, STEAM-LO's performance worsens noticeably as the lidar timestamp frequency is reduced. However, at lower timestamp frequencies, STEAM-LO becomes real-time. For STEAM-LIO, the performance worsens only modestly as the timestamp frequency is reduced, and the reduction in runtime is also more modest, as depicted in Figure 9.11.

Finally, we analyze the effect of varying the parameters of the diagonal of $\boldsymbol{Q}$, the power spectral density matrix, where the default parameters are $\text{diag}(\boldsymbol{Q}) = \{50, 50, 50, 5, 5, 5\}$.

In Table 9.4, we can see that STEAM-LO is more sensitive to varying the parameters of $\boldsymbol{Q}$ than STEAM-LIO. The performance of STEAM-LO improves substantially when decreasing $\boldsymbol{Q}$. The effect of this reduction is to increase the weight of penalizing the state estimates from deviating

from a constant velocity. However, STEAM-LO fails when increasing the parameters of $\boldsymbol{Q}$.

9.3.4 Boreas Results

The Boreas dataset test set features 102km or 4.3h of driving data. See Chapter 4 for more details on the Boreas dataset. We extract 200Hz raw IMU measurements from the Applanix logs and ensure that the IMU measurements are not bias-corrected in any way by the GPS. Table 9.3 shows our quantitative results for this experiment, where we compare several variations of our approach: lidar odometry, lidar-inertial odometry, radar odometry, and radar-inertial odometry. We also compare against our previously published work Visual Teach & Repeat 3 (VTR3) [28].

It is somewhat surprising that our lidar-inertial odometry does not do much better than our lidar odometry here. We hypothesize that for relatively slow-moving ground vehicles, as in the Boreas dataset, our continuous-time lidar odometry is sufficient to compensate for the motion distortion in the pointcloud. As such, the additional inertial inputs do not significantly improve performance. On the other hand, we can see that for radar odometry, including an IMU results in a significant improvement of 43%. Our interpretation of this result is that, due to the sparsity and noisiness of the radar data, there is more room for improvement by including an IMU. We improve our results even further for our competition submission, STEAM-RIO++, described in the appendix.

Another observation is that when we evaluate our lidar odometry in $SE(2)$, we observe a significant gap in lidar and radar odometry performance. This is somewhat contrary to what has been shown in prior work where radar odometry appeared to be getting close to the performance of lidar [4]. One important caveat is that the underlying ground truth is in $SE(3)$, whereas radar odometry is estimated in $SE(2)$. As such, we have to project the ground truth from 3D to 2D before comparing it to the radar odometry estimates. This projection becomes less accurate as the trajectory length increases. The KITTI odometry metric computes the average drift over all subsequences of lengths $\{100m, 200m, \cdots, 800m\}$. Thus, it is likely that a large part of this apparent radar odometry error is due to this projection error. It appears that we need an improved set of metrics to better compare radar and lidar odometry in a fair manner. We leave this as an area of future work.

Figure 9.12 shows a qualitative example of trajectories estimated by our approach on the Boreas dataset. Our lidar odometry remains close to the ground truth, while our lidar-inertial odometry achieves a similar result. It can also be observed that our radar-inertial odometry is notably better than our radar-only odometry. We also compare the odometry metrics as a function of path length in Figure 9.13 where we observe that including an IMU results in only a minor improvement for lidar odometry but results in a significant improvement for radar odometry. In Figure 9.14, we plot the

Table 9.4: ATE results (m) on sequence 01-Short of the Newer College Dataset when varying $\text{diag}(\boldsymbol{Q}) = \{50, 50, 50, 5, 5, 5\}$.

$\boldsymbol{Q}$	STEAM-LO	STEAM-LIO
$\times 1/4$	0.3098	0.3080
$\times 1/2$	0.3287	0.3071
$\times 1$	0.3398	0.3042
$\times 2$	Failed	0.3057
$\times 4$	Failed	0.3083
$\times 8$	Failed	0.3056

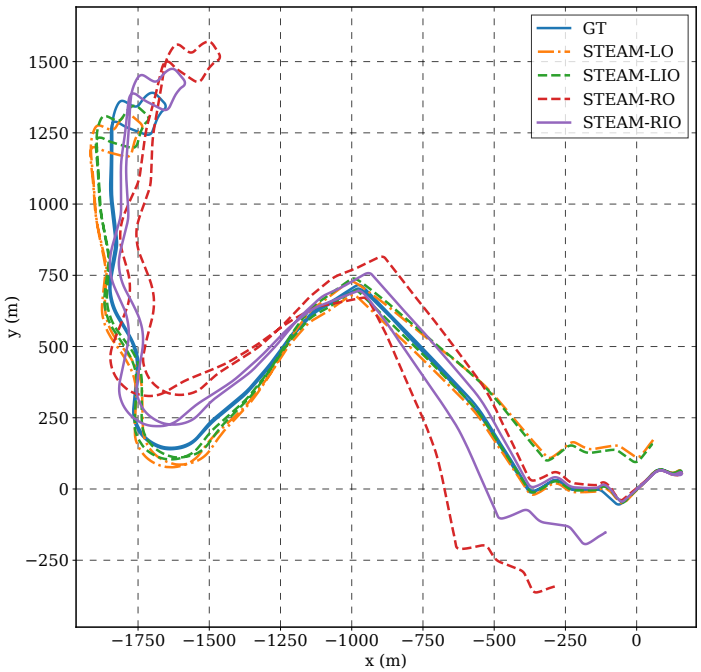


Figure 9.12: In this figure we compare the performance of odometry approaches presented in this chapter using the Boreas dataset. The depicted sequence is 2021-01-26-10-59 which was collected during a snowstorm.

odometry errors vs. time, comparing frame-to-frame odometry estimates vs. the ground truth. We also plot the estimated 3σ uncertainty bounds in red. Note that our approach is quite consistent; our estimated uncertainty does a good job of capturing the actual spread of the error. For each new lidar frame, our approach estimates the pose of the vehicle in a drifting map frame $\hat{\mathbf{T}}_k = \hat{\mathbf{T}}_{v,i}(t_k)$ where t_k corresponds to the temporal middle of the scan. We first compose two of these estimates to obtain a relative odometry pose change,

$$\hat{\mathbf{T}}_{k,k-1} = \hat{\mathbf{T}}_k \hat{\mathbf{T}}_{k-1}^{-1}. \tag{9.16}$$

The error that we compute is

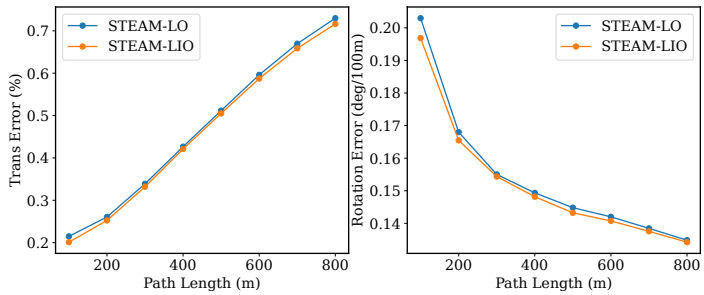
$$\boldsymbol{\xi}_{k,k-1} = \ln\left(\hat{\mathbf{T}}_{k,k-1}\mathbf{T}_{k,k-1}^{-1}\right)^{\vee}, \tag{9.17}$$

where $\mathbf{T}_{k,k-1}$ is the ground-truth odometry pose. We estimate a covariance $\hat{\mathbf{P}}_{k,k}$ for the pose of each lidar frame by interpolating the covariance over the sliding window at time t_k. See [16, §11.3.2] for details. The covariance of $\hat{\mathbf{T}}_{k,k-1}$, $\text{cov}(\hat{\mathbf{T}}_{k,k-1}) = \boldsymbol{\Sigma}_{k,k-1}$, is obtained using [16]

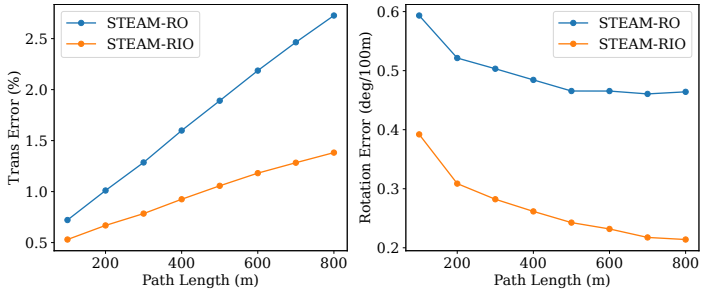
$$\boldsymbol{\Sigma}_{k,k-1} \approx \hat{\mathbf{P}}_{k,k} + \boldsymbol{\mathcal{T}}_{k,k-1}\hat{\mathbf{P}}_{k-1,k-1}\boldsymbol{\mathcal{T}}_{k,k-1}^{T}, \tag{9.18}$$

where $\boldsymbol{\mathcal{T}}_{k,k-1} = \text{Ad}(\hat{\mathbf{T}}_{k,k-1})$. 9.17 and 9.18 are then used to produce the plots in Figure 9.14. We compute the normalized estimation error squared (NEES) using

$$\text{NEES} = \sum_{k=1}^{K} \frac{\boldsymbol{\xi}_{k,k-1}^{T}\boldsymbol{\Sigma}_{k,k-1}^{-1}\boldsymbol{\xi}_{k,k-1}}{\dim(\boldsymbol{\xi}_{k,k-1})K}. \tag{9.19}$$



(a) Lidar Odometry Drift vs. Path Length



(b) Radar Odometry Drift vs. Path Length

Figure 9.13: Here we compare odometric drift vs. path length for lidar and radar odometry. Lidar-inertial odometry performs similarly to lidar odometry. However, radar-inertial odometry improves noticeably over radar odometry.

For the lidar-inertial odometry shown in Figure 9.12 and Figure 9.14, we obtain a NEES of 1.04 where an ideal value is 1.0. This means that our estimator is slightly overconfident here.

In Figure 9.15, we compare the maps generated using our lidar-inertial odometry and our radar-inertial odometry. The lidar map is coloured by height and is displayed using a top-down orthographic projection. Our lidar-inertial odometry has minimal drift, so the map it generates aligns quite well with satellite imagery. The lidar map is capturing a high level of detail. To produce the radar map in the figure, we took snapshots of the online radar map every 10m and aligned these submaps using the estimated odometry. We then removed noisy detections from the radar map by performing a radius outlier removal of points with less than two neighbors within a radius of 0.25m and a statistical outlier removal of points greater than one standard deviation above the average point-to-point distance. The radar map drifts slightly with respect to the lidar map. Note that we did not use any loop closures to generate these maps, so there is room for improvement. Even though these maps contain some drift, we showed in our prior work that maps only need to be locally consistent to enable accurate localization [30].

9.4 Conclusions

In this chapter, we showed that our Gaussian process motion prior is often sufficient to compensate for motion-distorted lidar data when there are sufficient geometric features. However, in challenging scenarios such as in the Newer College Dataset, we showed that our continuous-time lidar odometry could be augmented with IMU measurements to handle these conditions. Even in the presence of aggressive motion, the majority of the improvement resulted from including gyroscope measurements,

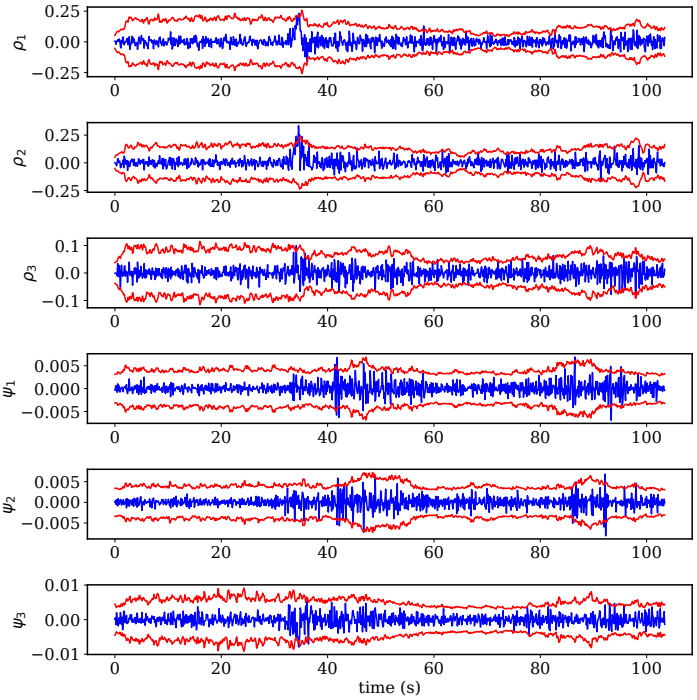
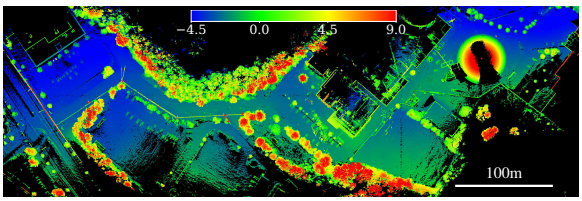


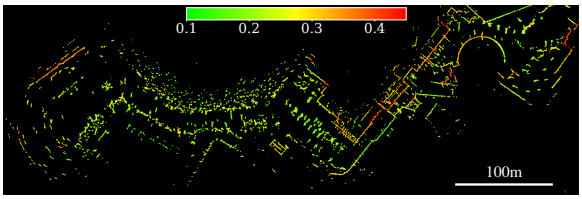
Figure 9.14: This figure plots error vs. time for our lidar-inertial odometry when compared to ground truth on the first 100 seconds of sequence 2021-01-26-10-59. The red lines denote the estimated 3σ uncertainty bounds. Each row represents a dimension of the log map of the pose error where ρ is a translational dimension and ψ is a rotational dimension.

whereas the addition of accelerometer measurements yielded only a minor improvement. We showed that we could improve our radar odometry by 43% by including inertial measurements. Contrary to previous work, we showed that there is still a significant gap between the performance of radar and lidar odometry under nominal conditions. Part of this gap may be explained by difficulties comparing 3D and 2D odometry estimates. Improved metrics for this purpose is an area of future work. Including body-centric acceleration in the state is another area of future work.

(a) Satellite image of UTIAS



(b) Lidar map of UTIAS (colored by height)



(c) Radar map of UTIAS (colored by intensity)

Figure 9.15: This figure displays maps generated of the University of Toronto Institute for Aerospace Studies (UTIAS). The data was obtained using the first 134 seconds of the 2021-02-09 Boreas sequence.

Chapter 10

Conclusion

The main goal of this thesis was to show that radar can be a viable alternative to lidar as a localization sensor in the autonomous driving domain. Toward this goal, we created the Boreas dataset, enabling us to compare lidar and radar localization across varying seasons and weather conditions. We showed that centimetre-accurate localization of radar to radar maps and radar to lidar maps is possible. In doing so, we established that the challenges inherent to working with radar, such as sparse or noisy measurements, can be overcome. This thesis also contributed to advancing the state of the art in radar odometry, reducing the benchmark translational drift from 2% to 0.6%.

However, this thesis did not demonstrate results in adverse weather conditions where radar showed a clear advantage over lidar. Other researchers have shown that radar can outperform lidar in artificially created dense smoke conditions [113]. However, collecting adverse weather datasets in the real world would benefit the community greatly by stress-testing our systems and potentially showing a case where radar outperforms lidar; this is a clear area for future work. Nevertheless, radar-based localization may prove valuable as a low-cost alternative to lidar-based localization, where multiple cheap automotive radars can be used instead of the Navtech radar.

The radar-based localization methods presented in this work have numerous potential applications. In areas where robustness to adverse conditions is critical, radar may become integral to robotics deployments. These potential applications include mining and disaster response. Marine radar may also be important for localizing surface vessels in GPS-denied environments. Low-cost single-chip radar may also be useful for indoor robotics applications. Will we ever have autonomous vehicles in Canada operating in white-out, blizzard-like conditions? If we do, they will need radar to make it happen.

We also presented a continuous-time lidar-inertial odometry method that is exceptionally robust to agile motion. This work may be useful in quadruped robots, two-legged robots, or fast-moving drones.

We showed that IMUs do not necessarily need to be treated as inputs to a motion model and then preintegrated. Further, we showed that treating them as a measurement of the state may be a fortuitous direction for further research. As future researchers seek to improve existing systems by applying these methods in highly dynamic scenarios, our investigation may be a useful starting point.

We continue to use the Boreas dataset extensively in our work, adding new data and improving

the existing data. Among the broader research community, the Boreas dataset is starting to be employed in others' works and was also used as part of a radar odometry competition[1]. We hope the Boreas dataset will continue to be used to benchmark and measure progress toward improving radar odometry and localization.

10.1 Future Work

As we showed in **Chapter 7**, lidar localization can still outperform radar even under moderate levels of precipitation. However, there remain conditions where we expect lidar to fail. For example, Park et al. [113] demonstrated the superiority of radar in dense smoke conditions. It would be interesting to attempt to collect datasets that include extreme weather events such as heavy rain, blizzards, dense fog or smoke, or dust storms. In practice, collecting datasets such as these is challenging due to the rarity of these events. However, these datasets would undoubtedly prove valuable to the community for identifying and studying lidar-based localization failure modes and finding ways to leverage radar or other sensors, such as IMUs, to overcome these failure modes. It may be possible to simply rely on simulation environments to test our systems for robustness to adverse weather. However, evaluating the sim-to-real gap for a simulator is challenging when we have no real dataset in those conditions to compare it against. So, at the very least, datasets collected in extreme weather conditions could be used to validate simulators attempting to generate such conditions. Radar simulation is also an active area of research where one interesting problem is to learn to generate novel viewpoints based on a real dataset towards training and testing autonomous agents in a closed-loop simulator.

There has been plenty of work in developing radar-based autonomy. However, relatively few system papers demonstrate these algorithms being applied in closed-loop robotic systems. There is an opportunity to perform these experiments and report the results to the community.

Radar odometry and localization continue to be active areas of research. More performance may yet be obtained from spinning mechanical radar odometry. However, we appear to be hitting a point of diminishing returns. Furthermore, the current state of the art for radar odometry is likely already sufficient to enable radar SLAM or radar localization. More work appears to be needed in the areas of automotive radar, low-cost radar sensors, and marine radar. Radar SLAM continues to receive interest, and some of the remaining challenges include robustness to radar noise and sparsity. There is also work to be done in the area of Doppler-enabled spinning mechanical radar, which employs alternating up-chirps and down-chirps in a triangle-wave pattern for the frequency modulation of the radar.

In the area of mapping and localization in general, potential areas for future research include addressing highly dynamic motions and sparse or degenerate geometry such as tunnels. As the field has matured, the focus has shifted towards higher-level concepts such as localizability or certifying map quality. In safety-critical applications, it would also be ideal to certify that the outputs of our estimators are optimal, another active area of research.

In **Chapter 8**, we presented an approach to lidar-inertial odometry where the IMU is treated as a measurement of the state. This framework would also work well for the combination of several asynchronous IMUs. There is also further work to be done on treating IMUs as a measurement of

[1] `https://sites.google.com/view/radar-robotics/competition`

the state. One issue we encountered was that our local Gaussian process (GP) formulation did not lend itself easily to preintegrating imu-as-a-measurement factors along with motion prior factors. One potential solution might be to formulate a singularity-free global GP representation for SE(3).

To date, relatively few published works use a spinning mechanical radar for object detection either in isolation or in combination with other sensors such as lidar or cameras. There is an opportunity to do further work in this area.

Appendix A

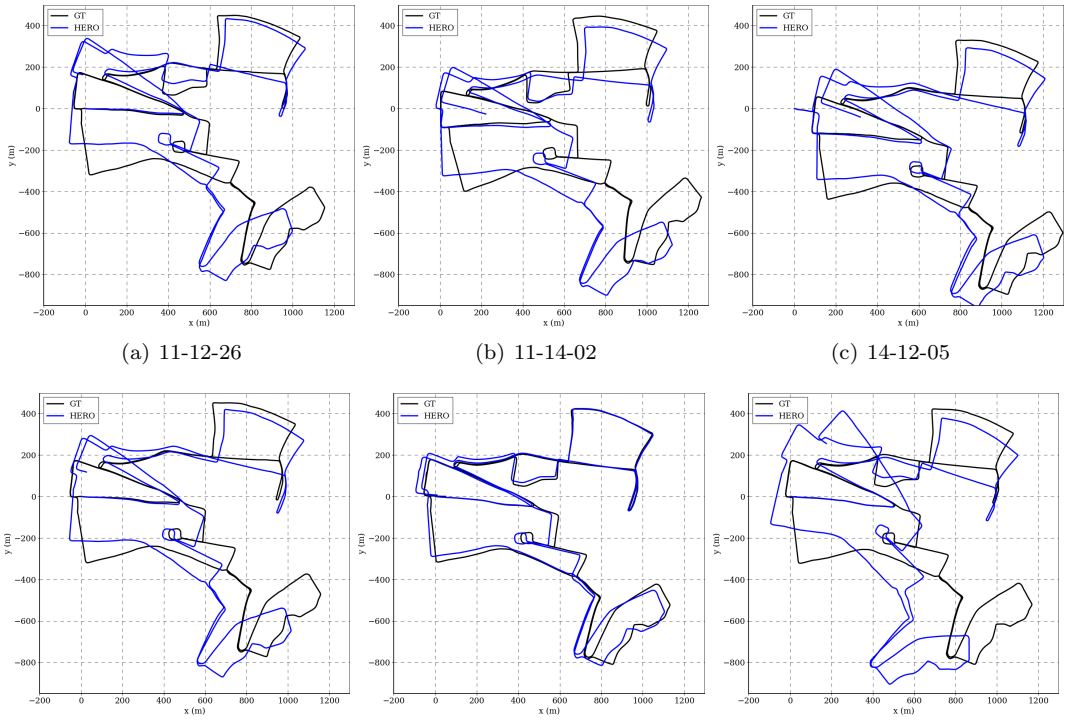
Supplementary Results for HERO

Table A.1: Evaluation on 7 sequences from the Oxford Dataset. Performance is reported as translational drift (%) / rotational drift (deg/1000m) using the common KITTI odometry metric. This table is restricted to algorithms that have been tested on similar sequences. Cen RO, Kung RO, and MC-RANSAC are hand-crafted algorithms. Masking by Moving, Under the Radar, and HERO are all learning-based algorithms. HERO is currently the only self-supervised learning-based radar odometry approach. *Tested on all 32 sequences. **Uses dense correlation between scans.

Method	**Evaluation**	**Sequences**							**Mean**
		10-14-02	11-12-26	11-14-02	14-12-05	15-13-06	16-11-53	17-11-46	
Cen RO [36]	[18]	N/A	N/A	N/A	N/A	N/A	N/A	N/A	3.72/9.5
Kung RO* [85]	[85]	N/A	N/A	N/A	N/A	N/A	N/A	N/A	1.96/6.0
Masking by Moving** [20]	[20]	N/A	N/A	N/A	N/A	N/A	N/A	N/A	1.16/3.0
Under the Radar [18]	[18]	N/A	N/A	N/A	N/A	N/A	N/A	N/A	2.05/6.7
MC-RANSAC [28]	Ours	3.43/11.2	3.72/12.9	3.27/10.9	3.41/10.7	3.07/9.8	3.12/9.9	3.21/11.0	3.31/10.9
HERO (Ours)	Ours	1.86/5.9	1.85/6.2	1.93/6.5	1.96/6.8	1.80/6.1	2.43/7.1	2.08/7.1	1.99/6.5

Table A.2: An evaluation on 8 sequences from the Oxford Dataset. This experiment required a new train/validation/test split. 16-13-42 was used for validation. Performance is reported as translational drift (%) / rotational drift (deg/100m) using the common KITTI odometry metric. *Uses loop closures.

Method	**Evaluation**	**Sequences**								**Mean**
		10-11-46	10-12-32	16-11-53	16-13-09	17-13-26	18-14-14	18-14-46	18-15-20	
Hong RO [71]	[71]	2.16/0.6	2.32/0.7	2.49/0.7	2.62/0.7	2.27/0.6	2.29/0.7	2.12/0.6	2.25/0.7	2.32/0.7
Hong SLAM* [71]	[71]	1.96/0.7	1.98/0.6	1.81/0.6	1.48/0.5	1.71/0.5	2.22/0.7	1.68/0.5	1.77/0.6	1.83/0.6
CFEAR [3]	[3]	1.65/0.48	1.64/0.48	1.99/0.53	1.86/0.52	1.66/0.48	1.71/0.49	1.79/0.5	1.75/0.51	1.76/0.50
HERO (Ours)	Ours	2.14/0.71	1.77/0.62	2.01/0.61	1.75/0.59	2.04/0.73	1.83/0.61	1.97/0.65	2.20/0.77	1.96/0.66



(a) 11-12-26 (b) 11-14-02 (c) 14-12-05

(d) 15-13-06 (e) 16-11-53 (f) 17-11-46

Figure A.1: These results illustrate the performance of HERO during each of the test sequences. See Figure 6.4 for HERO's performance on 10-14-02.

Appendix B

Supplementary Results for VTR3

Table B.1: $SE(2)$ Metric Localization RMSE Results (Reference Sequence: 2020-11-26)

	Lidar-to-Lidar		
	lateral (m)	longitudinal (m)	heading (deg)
2020-12-04	0.060	0.059	0.035
2021-01-26	0.023	0.026	0.040
2021-02-09	0.030	0.031	0.041
2021-03-09	0.021	0.028	0.035
2021-06-29	0.025	0.056	0.050
2021-09-08	0.030	0.036	0.048
2021-10-05	0.028	0.038	0.041
2021-10-26	0.032	0.042	0.041
2021-11-06	0.030	0.032	0.039
2021-11-28	0.025	0.041	0.035
mean	**0.031**	**0.039**	**0.040**

	Radar-to-Radar		
	lateral (m)	longitudinal (m)	heading (deg)
2020-12-04	0.072	0.082	0.211
2021-01-26	0.048	0.055	0.227
2021-02-09	0.053	0.051	0.235
2021-03-09	0.051	0.053	0.233
2021-06-29	0.069	0.095	0.246
2021-09-08	0.067	0.110	0.269
2021-10-05	0.069	0.109	0.288
2021-10-26	0.060	0.119	0.283
2021-11-06	0.062	0.155	0.256
2021-11-28	0.058	0.190	0.436
mean	**0.061**	**0.102**	**0.268**

	Radar-to-Lidar		
	lateral (m)	longitudinal (m)	heading (deg)
2020-12-04	0.074	0.135	0.135
2021-01-26	0.095	0.128	0.183
2021-02-09	0.061	0.125	0.135
2021-03-09	0.057	0.123	0.135
2021-06-29	0.069	0.122	0.139
2021-09-08	0.063	0.108	0.161
2021-10-05	0.061	0.074	0.147
2021-10-26	0.052	0.064	0.183
2021-11-06	0.051	0.057	0.183
2021-11-28	0.053	0.068	0.310
mean	**0.064**	**0.100**	**0.171**

Table B.2: $SE(2)$ Odometry Results

	Radar	
	Translation (%)	Rotation (deg/100m)
2020-12-04	1.92	0.53
2021-01-26	2.27	0.66
2021-02-09	1.94	0.59
2021-03-09	2.00	0.59
2020-04-22	2.56	0.63
2021-06-29-18	1.86	0.56
2021-06-29-20	1.94	0.59
2021-09-08	1.88	0.57
2021-09-09	1.98	0.60
2021-10-05	2.87	0.78
2021-10-26	1.89	0.53
2021-11-06	1.24	0.34
2021-11-28	1.24	0.38
mean	**2.02**	**0.58**

Table B.3: $SE(3)$ Odometry Results

	Lidar	
	Translation (%)	Rotation (deg/100m)
2020-12-04	0.49	0.14
2021-01-26	0.51	0.16
2021-02-09	0.49	0.14
2021-03-09	0.57	0.17
2020-04-22	0.49	0.15
2021-06-29-18	0.58	0.17
2021-06-29-20	0.62	0.18
2021-09-08	0.57	0.17
2021-09-09	0.63	0.19
2021-10-05	0.59	0.17
2021-10-26	0.48	0.14
2021-11-06	0.50	0.15
2021-11-28	0.46	0.14
mean	**0.54**	**0.16**

Table B.4: $SE(3)$ Metric Localization RMSE ResultsReference Sequence: 2020-11-26

	Lidar-to-Lidar					
	lateral (m)	longitudinal (m)	vertical (m)	roll (deg)	pitch (deg)	heading (deg)
2020-12-04	0.060	0.059	0.100	0.021	0.037	0.034
2021-01-26	0.023	0.026	0.080	0.030	0.042	0.040
2021-02-09	0.030	0.031	0.030	0.025	0.045	0.041
2021-03-09	0.021	0.028	0.034	0.024	0.045	0.034
2021-06-29	0.025	0.056	0.046	0.028	0.048	0.050
2021-09-08	0.030	0.036	0.054	0.025	0.046	0.047
2021-10-05	0.028	0.038	0.049	0.025	0.045	0.041
2021-10-26	0.032	0.042	0.037	0.024	0.043	0.040
2021-11-06	0.030	0.032	0.052	0.025	0.041	0.040
2021-11-28	0.025	0.041	0.067	0.025	0.044	0.034
mean	**0.031**	**0.039**	**0.055**	**0.025**	**0.043**	**0.040**

Appendix C

Supplementary Results for STEAM-RIO

STEAM-RIO was submitted to be part of the 2024 Radar in Robotics competition. In order to achieve competitive results, we increased the length of the sliding window from two scans to four. We also switched from a Cauchy loss to a Huber loss with a more restrictive threshold to filter out outliers. We increased the weight given to gyroscope measurements by decreasing their associated measurement covariance. We also incorporated a version of keyframing where new radar frames were not added to the sliding local map unless the vehicle travelled at least one metre. Putting together all of these changes, we arrive at our competition submission, STEAM-RIO++ which yields a 34% improvement in translational drift over STEAM-RIO. In the competition, we placed third, where the 2nd place submission was CFEAR with 0.61% drift [2]. Notably, CFEAR did not use an IMU, and their approach is computationally very efficient. In order to achieve their highest-level competition peformance, they greatly increased the length of their sliding window and employed a coarse-to-fine registration approach. The first-place entry CFEAR++ achieved 0.51% translation error[91]. CFEAR++ is based on CFEAR, however the authors also used an IMU as well as a semantic segmentation model to remove distracting features in order to improve performance. Our approach is competitive with the state of the art while being capable of supporting additional asynchronous measurement factors such as wheel encoder measurements. CFEAR employs a point-to-line loss and including such a loss in our framework is an area of future work. It has been shown in the context of lidar that point-to-plane losses tend to outperform point-to-point, and so it is conceivable that this might also hold for radar.

Table C.1: Boreas Odometry Supplementary Results (102km / 4.3h): translational drift (%) / rotational drift (deg/100m).

Boreas	STEAM-RO	STEAM-RIO	STEAM-RIO++
2020-12-04	1.43 / 0.41	0.93 / 0.26	0.76 / 0.20
2021-01-26	1.10 / 0.33	0.61 / 0.18	0.50 / 0.24
2021-02-09	1.27 / 0.38	0.63 / 0.20	0.40 / 0.13
2021-03-09	1.24 / 0.35	0.71 / 0.19	0.58 / 0.17
2020-04-22	1.48 / 0.41	0.99 / 0.27	0.67 / 0.18
2021-06-29-18	1.55 / 0.46	1.04 / 0.29	0.66 / 0.19
2021-06-29-20	1.70 / 0.48	0.96 / 0.26	0.75 / 0.20
2021-09-08	2.01 / 0.59	1.22 / 0.35	0.74 / 0.21
2021-09-09	2.16 / 0.64	1.19 / 0.33	0.56 / 0.15
2021-10-05	2.27 / 0.63	1.01 / 0.28	0.58 / 0.16
2021-10-26	1.88 / 0.53	0.97 / 0.27	0.63 / 0.18
2021-11-06	1.86 / 0.54	1.07 / 0.29	0.74 / 0.21
2021-11-28	1.95 / 0.57	1.04 / 0.29	0.56 / 0.16
AVG	1.68 / 0.49	0.95 / 0.27	0.62 / 0.18
ΔT	115ms	139ms	153ms

Appendix D

Preintegration Using a Schur Complement

The method of preintegration presented in section 8.2.1 is mathematically equivalent to marginalizing out the unwanted states from the full Bayesian posterior. Marginalization can also be performed using a Schur complement. We will use the Schur complement to efficiently marginalize out unwanted states from our continuous-time formulation. By exploiting sparsity, this can be performed in $O(K)$ time, which is the same time complexity as the classic approach presented in section 8.2.1.

We consider the factor graph shown in Figure 8.1, which could potentially be a result of our continuous-time state estimation with binary motion prior factors, unary measurement factors, and a unary prior factor on the initial state $\mathbf{x}_0$. Equivalently, the Gauss-Newton system of equations associated with Figure 8.1 can be written in the following form:

$$\underbrace{\left(\mathbf{A}^{-T}\mathbf{Q}^{-1}\mathbf{A} + \mathbf{C}^{T}\mathbf{R}^{-1}\mathbf{C}\right)}_{\mathbf{L}}\hat{\mathbf{x}} = \underbrace{\mathbf{A}^{-T}\mathbf{Q}^{-1}\check{\mathbf{x}} + \mathbf{C}^{T}\mathbf{R}^{-1}\mathbf{y}}_{\mathbf{r}}, \tag{D.1}$$

where $\mathbf{L}$ is block-tridiagonal,

$$\mathbf{L} = \begin{bmatrix} \mathbf{L}_{0,0} & \mathbf{L}_{0,1:3} & & & \\ \mathbf{L}_{0,1:3}^{T} & \mathbf{L}_{1:3,1:3} & \mathbf{L}_{1:3,4} & & \\ & \mathbf{L}_{1:3,4}^{T} & \mathbf{L}_{4,4} & \mathbf{L}_{4,5:7} & \\ & & \mathbf{L}_{4,5:7}^{T} & \mathbf{L}_{5:7,5:7} & \mathbf{L}_{5:7,8} \\ & & & \mathbf{L}_{5:7,8}^{T} & \mathbf{L}_{8,8} \end{bmatrix} = \left[\begin{array}{c|ccc|c|ccc|c} * & * & & & & & & & \\ \hline * & * & * & & & & & & \\ & * & * & * & & & & & \\ & & * & * & * & & & & \\ \hline & & & * & * & * & & & \\ \hline & & & & * & * & * & & \\ & & & & & * & * & * & \\ & & & & & & * & * & * \\ \hline & & & & & & & * & * \end{array}\right]. \tag{D.2}$$

Here, we consider the case where we would like to marginalize the full posterior such that we only

retain states $\{\mathbf{x}_0, \mathbf{x}_4, \mathbf{x}_8\}$. After marginalizing out the unwanted states, our system becomes

$$\mathbf{L}_{\text{small}}\hat{\mathbf{x}}_{\text{small}} = \mathbf{r}_{\text{small}}, \tag{D.3}$$

where by the Schur complement,

$$\mathbf{L}_{\text{small}} = \begin{bmatrix} \mathbf{L}_{0,0} & & \\ & \mathbf{L}_{4,4} & \\ & & \mathbf{L}_{8,8} \end{bmatrix} - \begin{bmatrix} \mathbf{L}_{0,1:3} & \\ \mathbf{L}_{1:3,4}^T & \mathbf{L}_{4,5:7} \\ & \mathbf{L}_{5:7,8}^T \end{bmatrix} \begin{bmatrix} \mathbf{L}_{1:3,1:3} & \\ & \mathbf{L}_{5:7,5:7} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{L}_{0,1:3}^T & \mathbf{L}_{1:3,4} & \\ & \mathbf{L}_{4,5:7}^T & \mathbf{L}_{5:7,8} \end{bmatrix}$$

$$= \left[\begin{array}{c|c|c} * & & \\ \hline & * & \\ \hline & & * \end{array}\right] - \left[\begin{array}{c|c|c} * & & \\ \hline & * & * \\ \hline & & * \end{array}\right] \times \left[\begin{array}{cc|ccc} * & * & & & \\ * & * & * & & \\ & * & * & & \\ \hline & & & * & * \\ & & & * & * & * \\ & & & & * & * \end{array}\right]^{-1} \left[\begin{array}{c|c|c} * & & \\ \hline & * & \\ \hline & & * \end{array}\right].$$

$\mathbf{L}_{\text{small}}$ can be computed efficiently by exploiting the primary and secondary sparsity. Note that

$$\begin{bmatrix} \mathbf{L}_{1:3,1:3} & \\ & \mathbf{L}_{5:7,5:7} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{L}_{0,1:3}^T & \mathbf{L}_{1:3,4} & \\ & \mathbf{L}_{4,5:7}^T & \mathbf{L}_{5:7,8} \end{bmatrix} = \begin{bmatrix} \mathbf{L}_{1:3,1:3} \setminus \mathbf{L}_{0,1:3}^T & \mathbf{L}_{1:3,1:3} \setminus \mathbf{L}_{1:3,4} & \\ & \mathbf{L}_{5:7,5:7} \setminus \mathbf{L}_{4,5:7}^T & \mathbf{L}_{5:7,5:7} \setminus \mathbf{L}_{5:7,8} \end{bmatrix}, \tag{D.4}$$

where each entry similar to $\mathbf{L}_{1:3,1:3} \setminus \mathbf{L}_{1:3,4}$ is shorthand for solving $\mathbf{L}_{1:3,1:3}\boldsymbol{\ell} = \mathbf{L}_{1:3,4}$ for $\boldsymbol{\ell}$. Each of these terms can be solved in linear time thanks to $\mathbf{L}_{1:3,1:3}$ and $\mathbf{L}_{5:7,5:7}$ being block-tridiagonal. Thus, $\mathbf{L}_{\text{small}}$ can be constructed in linear time, the same time complexity as the classic approach. Furthermore, it can be shown that the resulting matrix $\mathbf{L}_{\text{small}}$ is block-tridiagonal. In summary, Schur complement preintegration is a generalization of classic preintegration that can handle both binary motion prior factors and unary measurement factors while retaining the same linear time complexity as classic preintegration.

Appendix E

Analytical Gradients for Training the Singer Prior

Following the method presented by Wong et al. [153] for training the parameters of the Singer prior, we found it necessary to derive the analytical gradients of our objective with respect to the desired parameters. Since these gradients were not provided in [153], we provide them here instead for the convenience of the reader. Starting with the objective from (8.15), the discrete-time covariance $\mathbf{Q}_k$ of the Singer prior can be written as the product of two factors where

$$\mathbf{Q}_k = \underbrace{\begin{bmatrix} \boldsymbol{\sigma}^2 & & \\ & \boldsymbol{\sigma}^2 & \\ & & \boldsymbol{\sigma}^2 \end{bmatrix}}_{\mathbf{Q}_{\boldsymbol{\sigma}^2}} \mathbf{Q}(\Delta t_k, \boldsymbol{\alpha}). \tag{E.1}$$

The components of $\mathbf{Q}(\Delta t_k, \boldsymbol{\alpha})$ are provided by Wong et al. [153] and are repeated here,

$$\mathbf{Q}(\Delta t_k, \boldsymbol{\alpha}) = \begin{bmatrix} \mathbf{Q}_{11} & \mathbf{Q}_{12} & \mathbf{Q}_{13} \\ \mathbf{Q}_{12}^T & \mathbf{Q}_{22} & \mathbf{Q}_{23} \\ \mathbf{Q}_{13}^T & \mathbf{Q}_{23}^T & \mathbf{Q}_{33} \end{bmatrix}, \tag{E.2}$$

where

$$\mathbf{Q}_{11} = \frac{1}{2}\boldsymbol{\alpha}^{-5}\Big(\mathbf{1} - e^{-2\boldsymbol{\alpha}\Delta t_k} + 2\boldsymbol{\alpha}\Delta t_k + \frac{2}{3}\boldsymbol{\alpha}^3\Delta t_k^3 - 2\boldsymbol{\alpha}^2\Delta t_k^2 - 4\boldsymbol{\alpha}\Delta t_k e^{-\boldsymbol{\alpha}\Delta t_k}\Big), \tag{E.3a}$$

$$\mathbf{Q}_{12} = \frac{1}{2}\boldsymbol{\alpha}^{-4}\Big(e^{-2\boldsymbol{\alpha}\Delta t_k} + \mathbf{1} - 2e^{-\boldsymbol{\alpha}\Delta t_k} + 2\boldsymbol{\alpha}\Delta t_k e^{-\boldsymbol{\alpha}\Delta t_k} - 2\boldsymbol{\alpha}\Delta t_k + \boldsymbol{\alpha}^2\Delta t_k^2\Big), \tag{E.3b}$$

$$\mathbf{Q}_{13} = \frac{1}{2}\boldsymbol{\alpha}^{-3}\left(\mathbf{1} - e^{-2\boldsymbol{\alpha}\Delta t_k} - 2\boldsymbol{\alpha}\Delta t_k e^{-\boldsymbol{\alpha}\Delta t_k}\right), \tag{E.3c}$$

$$\mathbf{Q}_{22} = \frac{1}{2}\boldsymbol{\alpha}^{-3}\left(4e^{-\boldsymbol{\alpha}\Delta t_k} - 3\cdot\mathbf{1} - e^{-2\boldsymbol{\alpha}\Delta t_k} + 2\boldsymbol{\alpha}\Delta t_k\right), \tag{E.3d}$$

$$\mathbf{Q}_{23} = \frac{1}{2}\boldsymbol{\alpha}^{-2}\left(e^{-2\boldsymbol{\alpha}\Delta t_k} + \mathbf{1} - 2e^{-\boldsymbol{\alpha}\Delta t_k}\right), \tag{E.3e}$$

$$\mathbf{Q}_{33} = \frac{1}{2}\boldsymbol{\alpha}^{-1}\left(\mathbf{1} - e^{-2\boldsymbol{\alpha}\Delta t_k}\right). \tag{E.3f}$$

The motion error is given by

$$\mathbf{e}_k = \mathbf{x}_k - \mathbf{\Phi}(t_k, t_{k-1})\mathbf{x}_{k-1}, \tag{E.4}$$

where

$$\mathbf{\Phi}(t_k, t_{k-1}) = \begin{bmatrix} \mathbf{1} & \Delta t_k \mathbf{1} & (\boldsymbol{\alpha}\Delta t_k - \mathbf{1} + \exp(-\boldsymbol{\alpha}\Delta t_k))\boldsymbol{\alpha}^{-2} \\ \mathbf{0} & \mathbf{1} & (\mathbf{1} - \exp(-\boldsymbol{\alpha}\Delta t_k))\boldsymbol{\alpha}^{-1} \\ \mathbf{0} & \mathbf{0} & \exp(-\boldsymbol{\alpha}\Delta t_k) \end{bmatrix} \tag{E.5}$$

is the state transition function. $\boldsymbol{\sigma}^2$ and $\boldsymbol{\alpha}$ are both diagonal matrices, whose size depends on the dimension of the state. For example, for a 6D state, $\boldsymbol{\sigma}^2 = \text{diag}(\sigma_1^2, \sigma_2^2, \sigma_3^2, \sigma_4^2, \sigma_5^2, \sigma_6^2)$. The gradients of the objective with respect to the components of $\boldsymbol{\sigma}^2$ and $\boldsymbol{\alpha}$ are then

$$\frac{\partial J_t}{\partial \alpha_i} = \frac{1}{2}\sum_k \left\{ 2\mathbf{e}_k^T \mathbf{Q}_k^{-1} \frac{\partial \mathbf{e}_k}{\partial \alpha_i} - \mathbf{e}_k^T \mathbf{Q}_k^{-1} \frac{\partial \mathbf{Q}_k}{\partial \alpha_i} \mathbf{Q}_k^{-1} \mathbf{e}_k + \text{ tr}\left(\mathbf{Q}_k^{-1} \frac{\partial \mathbf{Q}_k}{\partial \alpha_i} \right) \right\}, \tag{E.6a}$$

$$\frac{\partial J_t}{\partial \sigma_i^2} = \frac{3K}{2\sigma_i^2} - \frac{1}{2}\sum_k \frac{1}{\sigma_i^4} \mathbf{e}_k^T \mathbf{Q}(\Delta t_k, \boldsymbol{\alpha})^{-1} \frac{\partial \mathbf{Q}_{\boldsymbol{\sigma}^2}}{\partial \sigma_i^2} \mathbf{e}_k, \tag{E.6b}$$

for each α_i and σ_i^2, respectively, where $J = \sum_{t=1}^{T} J_t$. The partial derivatives of $\mathbf{Q}(\Delta t_k, \boldsymbol{\alpha})$ and $\mathbf{e}_k$ with respect to α_i are then given by

$$\frac{\partial \mathbf{Q}_{11}}{\partial \alpha_i} = \left[-\frac{2\Delta t_k^3}{3\alpha_i^3} + \frac{\Delta t_k^2 (2e^{-\alpha_i \Delta t_k} + 3)}{\alpha_i^4} + \frac{5(e^{-2\alpha_i \Delta t_k} - 1)}{2\alpha_i^6} + \frac{\Delta t_k (e^{-2\alpha_i \Delta t_k} + 8e^{-\alpha_i \Delta t_k} - 4)}{\alpha_i^5} \right] \boldsymbol{\delta}_{ii}, \tag{E.7a}$$

$$\frac{\partial \mathbf{Q}_{12}}{\partial \alpha_i} = \left[-\frac{\Delta t_k^2 (e^{-\alpha_i \Delta t_k} + 1)}{\alpha_i^3} + \frac{\Delta t_k (3 - e^{-2\alpha_i \Delta t_k} - 2e^{-\alpha_i \Delta t_k})}{\alpha_i^4} + \frac{4e^{-\alpha_i \Delta t_k} - 2e^{-2\alpha_i \Delta t_k} - 2}{\alpha_i^5} \right] \boldsymbol{\delta}_{ii}, \tag{E.7b}$$

$$\frac{\partial \mathbf{Q}_{13}}{\partial \alpha_i} = \left[\frac{\Delta t_k^2 e^{-\alpha_i \Delta t_k}}{\alpha_i^2} + \frac{3(e^{-2\alpha_i \Delta t_k} - 1)}{2\alpha_i^4} + \frac{\Delta t_k (e^{-2\alpha_i \Delta t_k} + 2e^{-\alpha_i \Delta t_k})}{\alpha_i^3} \right] \boldsymbol{\delta}_{ii},$$

$$\frac{\partial \mathbf{Q}_{22}}{\partial \alpha_i} = \left[\frac{3e^{-2\alpha_i \Delta t_k} - 12e^{-\alpha_i \Delta t_k} + 9}{2\alpha_i^4} + \frac{\Delta t_k (e^{-2\alpha_i \Delta t_k} - 2e^{-\alpha_i \Delta t_k} - 2)}{\alpha_i^3} \right] \boldsymbol{\delta}_{ii}, \tag{E.7c}$$

$$\frac{\partial \mathbf{Q}_{23}}{\partial \alpha_i} = \left[\frac{2e^{-\alpha_i \Delta t_k} - e^{-2\alpha_i \Delta t_k} - 1}{\alpha_i^3} + \frac{\Delta t_k (e^{-\alpha_i \Delta t_k} - e^{-2\alpha_i \Delta t_k})}{\alpha_i^2} \right] \boldsymbol{\delta}_{ii}, \tag{E.7d}$$

$$\frac{\partial \mathbf{Q}_{33}}{\partial \alpha_i} = \left[\frac{e^{-2\alpha_i \Delta t_k} - 1}{2\alpha_i^2} + \frac{\Delta t_k e^{-2\alpha_i \Delta t_k}}{\alpha_i} \right] \boldsymbol{\delta}_{ii}, \tag{E.7e}$$

$$\frac{\partial \mathbf{e}_k}{\partial \alpha_i} = -\begin{bmatrix} \left(\frac{2(1 - e^{-\alpha_i \Delta t_k})}{\alpha_i^3} - \frac{\Delta t_k (e^{-\alpha_i \Delta t_k} + 1)}{\alpha_i^2} \right) \boldsymbol{\delta}_{ii} \\ \left(\frac{e^{-\alpha_i \Delta t_k} - 1}{\alpha_i^2} + \frac{\Delta t_k e^{-\alpha_i \Delta t_k}}{\alpha_i} \right) \boldsymbol{\delta}_{ii} \\ \left(-\Delta t_k e^{-\alpha_i \Delta t_k} \right) \boldsymbol{\delta}_{ii} \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \mathbf{x}_k, \tag{E.7f}$$

where $\boldsymbol{\delta}_{ii}$ is the Kronecker delta. We can use these gradients to learn the parameters of the Singer prior using gradient descent. In order to speed up training, we can solve for the optimal value of σ_i^2 at each iteration of gradient descent:

$$\sigma_i^{2^\star} = \frac{1}{3KT} \sum_t \sum_k \mathbf{e}_{k,t}^T \mathbf{Q}(\Delta t_{k,t}, \boldsymbol{\alpha})^{-1} \frac{\partial \mathbf{Q}_{\boldsymbol{\sigma}^2}}{\partial \sigma_i^2} \mathbf{e}_{k,t}. \tag{E.8}$$

Note that $\mathbf{Q}_k$ is numerically unstable for $\alpha < 1.0$. In this case, we use a Taylor series expansion about $\alpha = 0$ as an approximation. The Jacobian $\frac{\partial \mathbf{Q}_k}{\partial \alpha_i}$ is also numerically unstable for $\alpha < 4.0$. In this case, we can approximate the components of this matrix with either a Laurent series or Taylor series as $\alpha \to 0$.

The previous gradients work well for learning the parameters of a Gaussian process in simulation where the ground truth measurements of the state are noiseless. In reality, our source of ground truth will have some measurement covariance that may be estimated or taken from the datasheet of the sensor being used. In this case, computing the gradients of the objective with respect to the components of $\boldsymbol{\sigma}^2$ and $\boldsymbol{\alpha}$ is slightly more involved. We follow the approach presented by Wong et al. [153]. Now, our objective function looks at the entire trajectory at once,

$$J = -\ln p(\mathbf{y}|\boldsymbol{\sigma}, \boldsymbol{\alpha}) = \frac{1}{2}\mathbf{e}^T\mathbf{Q}^{-1}\mathbf{e} + \frac{1}{2}\ln|\mathbf{Q}| + \frac{n}{2}\ln 2\pi, \tag{E.9}$$

where $\mathbf{e}$ is a stacked version of all the individual error terms from each timestep $\mathbf{e}_k$, and

$$\mathbf{Q} = \begin{bmatrix} \boldsymbol{\Sigma}_{0,0} & \boldsymbol{\Sigma}_{0,1} & & \\ \boldsymbol{\Sigma}_{1,0}^T & \boldsymbol{\Sigma}_{1,1} & \boldsymbol{\Sigma}_{1,2} & \\ & \boldsymbol{\Sigma}_{1,2}^T & \ddots & \ddots \\ & & \ddots & \boldsymbol{\Sigma}_{K,K} \end{bmatrix}, \tag{E.10}$$

where

$$\boldsymbol{\Sigma}_{k,k} \approx \mathbf{R}_k + \boldsymbol{\Phi}(t_k, t_{k-1})\mathbf{R}_{k-1}\boldsymbol{\Phi}(t_k, t_{k-1})^T + \mathbf{Q}_k, \tag{E.11a}$$

$$\boldsymbol{\Sigma}_{k,k+1} \approx -\mathbf{R}_k\boldsymbol{\Phi}(t_{k+1}, t_k)^T, \tag{E.11b}$$

and $\mathbf{R}_k$ is the measurement covariance associated with local variable $\mathbf{x}_k$. The gradient of the objective function J with respect to GP parameter θ is

$$\frac{\partial J}{\partial \theta} = -\frac{1}{2}\mathbf{e}^T\mathbf{Q}^{-1}\frac{\partial \mathbf{Q}}{\partial \theta}\mathbf{Q}^{-1}\mathbf{e} + \mathbf{e}^T\mathbf{Q}^{-1}\frac{\partial \mathbf{e}}{\partial \theta} + \frac{1}{2}\mathrm{tr}\left(\mathbf{Q}^{-1}\frac{\partial \mathbf{Q}}{\partial \theta}\right) \tag{E.12}$$

where each of the Jacobians is evaluated using the current value of θ. We can compute the trace in $O(K)$ time by first computing only the block-tridiagonal components of $\mathbf{Q}^{-1}$. Since $\mathbf{Q}$ is itself block-tridiagonal, we can compute the blocks of the inverse that we need in $O(K)$ time [16]. Then, we can compute the trace of the matrix product in $O(K)$ time by only computing the elements along the diagonal of the matrix product. $\mathbf{Q}^{-1}\mathbf{e}$ can also be evaluated in $O(K)$ time by solving $\mathbf{Q}\mathbf{x} = \mathbf{e}$ for $\mathbf{x}$ using a sparse Cholesky solver. Using the gradient in (E.12) for each parameter, we can learn the parameters of the Gaussian process by minimizing the negative log likelihood using gradient descent.

Appendix F

IMU-as-Input Lidar-Inertial Baseline Jacobians

Perturbations to the state variables are defined as $\mathbf{C}_{iv} = \overline{\mathbf{C}}_{iv} \exp(\delta\phi^{\wedge})$, $\mathbf{r}_i^{vi} = \bar{\mathbf{r}}_i^{vi} + \overline{\mathbf{C}}_{iv}\delta\mathbf{r}$, $\mathbf{v}_i^{vi} = \bar{\mathbf{v}}_i^{vi} + \overline{\mathbf{C}}_{iv}\delta\mathbf{v}$, $\mathbf{b} = \bar{\mathbf{b}} + \delta\mathbf{b}$. The Jacobians of the point-to-plane error function (8.29) with respect to perturbations to the state variables are provided here,

$$\frac{\partial \mathbf{e}_j}{\partial \delta\mathbf{x}} = \begin{bmatrix} \frac{\partial \mathbf{e}_j}{\partial \mathbf{r}_i^{vi}(\tau_j)} & \frac{\partial \mathbf{e}_j}{\partial \delta\mathbf{C}_{iv}(\tau_j)} \end{bmatrix} \times \begin{bmatrix} \frac{\partial \mathbf{r}_i^{vi}(\tau_j)}{\partial \mathbf{r}_\ell} \frac{\partial \mathbf{r}_\ell}{\partial \delta\mathbf{x}} + \frac{\partial \mathbf{r}_i^{vi}(\tau_j)}{\partial \mathbf{r}_{\ell+1}} \frac{\partial \mathbf{r}_{\ell+1}}{\partial \delta\mathbf{x}} \\ \frac{\partial \delta\mathbf{C}_{iv}(\tau_j)}{\partial \delta\mathbf{C}_\ell} \frac{\partial \delta\mathbf{C}_\ell}{\partial \delta\mathbf{x}} + \frac{\partial \delta\mathbf{C}_{iv}(\tau_j)}{\partial \delta\mathbf{C}_{\ell+1}} \frac{\partial \delta\mathbf{C}_{\ell+1}}{\partial \delta\mathbf{x}} \end{bmatrix}, \tag{F.1}$$

where

$$\frac{\partial \mathbf{e}_j}{\partial \mathbf{r}_i^{vi}(\tau_j)} = -\mathbf{n}_j^T, \quad \frac{\partial \mathbf{e}_j}{\partial \delta\mathbf{C}_{iv}(\tau_j)} = \mathbf{n}_j^T \left(\overline{\mathbf{C}}_{iv}(\tau_j)(\mathbf{C}_{vs}\mathbf{q}_j + \mathbf{r}_v^{sv})^{\wedge} \right), \tag{F.2a}$$

$$\frac{\partial \mathbf{r}_i^{vi}(\tau_j)}{\partial \mathbf{r}_\ell} = (1-\alpha)\mathbf{1}, \quad \frac{\partial \mathbf{r}_i^{vi}(\tau_j)}{\partial \mathbf{r}_{\ell+1}} = \alpha\mathbf{1}, \tag{F.2b}$$

$$\frac{\partial \delta\mathbf{C}_{iv}(\tau_j)}{\partial \delta\mathbf{C}_\ell} = \mathbf{1} - \mathbf{A}(\alpha, \boldsymbol{\phi}), \quad \frac{\partial \delta\mathbf{C}_{iv}(\tau_j)}{\partial \delta\mathbf{C}_{\ell+1}} = \mathbf{A}(\alpha, \boldsymbol{\phi}), \tag{F.2c}$$

where $\mathbf{A}(\alpha, \boldsymbol{\phi}) = \alpha\mathbf{J}_r(\alpha\boldsymbol{\phi})\mathbf{J}_r(\boldsymbol{\phi})^{-1}$, $\boldsymbol{\phi} = \ln(\mathbf{C}_\ell^T\mathbf{C}_{\ell+1})^{\vee}$, and

$$\frac{\partial \delta \mathbf{C}_\ell}{\partial \delta \mathbf{C}_i} = \Delta \overline{\mathbf{C}}_{i\ell}^T, \text{ where } \quad \Delta \mathbf{C}_{i\ell} = \prod_{k=i}^{\ell-1} \exp\left(\Delta t_k (\tilde{\boldsymbol{\omega}}_k - \mathbf{b}_\omega(t_k))^\wedge\right), \tag{F.3a}$$

$$\frac{\partial \delta \mathbf{C}_\ell}{\partial \delta \mathbf{b}_\omega(t_i)} = -\sum_{k=i}^{\ell-1} \Delta \overline{\mathbf{C}}_{k+1,\ell}^T \mathbf{J}_r(\boldsymbol{\phi}_k) \Delta t_k, \text{ where } \quad \boldsymbol{\phi}_k = \Delta t_k (\tilde{\boldsymbol{\omega}}_k - \mathbf{b}_\omega(t_k)), \tag{F.3b}$$

$$\frac{\partial \mathbf{v}_\ell}{\partial \delta \mathbf{v}_i} = \mathbf{C}_i, \tag{F.3c}$$

$$\frac{\partial \mathbf{v}_\ell}{\partial \delta \mathbf{C}_i} = -\sum_{k=i}^{\ell-1} \mathbf{C}_k (\tilde{\mathbf{a}}_k - \mathbf{b}_a(t_k))^\wedge \Delta \overline{\mathbf{C}}_{ik}^T \Delta t_k, \tag{F.3d}$$

$$\frac{\partial \mathbf{v}_\ell}{\partial \delta \mathbf{b}_\omega(t_i)} = -\sum_{k=i}^{\ell-1} \mathbf{C}_k (\tilde{\mathbf{a}}_k - \mathbf{b}_a(t_k))^\wedge \frac{\partial \delta \mathbf{C}_k}{\partial \delta \mathbf{b}_\omega(t_i)} \Delta t_k, \tag{F.3e}$$

$$\frac{\partial \mathbf{v}_\ell}{\partial \delta \mathbf{b}_a(t_i)} = -\sum_{k=i}^{\ell-1} \mathbf{C}_k \Delta t_k, \tag{F.3f}$$

$$\frac{\partial \mathbf{r}_\ell}{\partial \delta \mathbf{v}_i} = \mathbf{C}_i \Delta t_{ij}, \tag{F.3g}$$

$$\frac{\partial \mathbf{r}_\ell}{\partial \delta \mathbf{C}_i} = \sum_{k=i}^{\ell-1} \left[\frac{\partial \mathbf{v}_k}{\partial \delta \mathbf{C}_i} \Delta t_k - \frac{1}{2} \mathbf{C}_k (\tilde{\mathbf{a}}_k - \mathbf{b}_a(t_k))^\wedge \Delta \overline{\mathbf{C}}_{ik}^T \Delta t_k^2 \right], \tag{F.3h}$$

$$\frac{\partial \mathbf{r}_\ell}{\partial \delta \mathbf{b}_\omega(t_i)} = \sum_{k=i}^{\ell-1} \left[\frac{\partial \mathbf{v}_k}{\partial \delta \mathbf{b}_\omega(t_i)} \Delta t_k - \frac{1}{2} \mathbf{C}_k (\tilde{\mathbf{a}}_k - \mathbf{b}_a(t_k))^\wedge \frac{\partial \delta \mathbf{C}_k}{\partial \delta \mathbf{b}_\omega(t_i)} \Delta t_k^2 \right], \tag{F.3i}$$

$$\frac{\partial \mathbf{r}_\ell}{\partial \delta \mathbf{b}_a(t_i)} = \sum_{k=i}^{\ell-1} \left[\frac{\partial \mathbf{v}_k}{\partial \delta \mathbf{b}_a(t_i)} \Delta t_k - \frac{1}{2} \mathbf{C}_k \Delta t_k^2 \right]. \tag{F.3j}$$

Appendix G

Interpolation Jacobians

We build continuous-time measurement factors by making use of the posterior Gaussian process interpolation formula. In order to do this, we need to compute the Jacobians of the perturbation to the interpolated state $\delta\mathbf{x}(\tau)$ with respect to the state perturbations at the bracketing estimation times $\delta\mathbf{x}_k, \delta\mathbf{x}_{k+1}$. Perturbations to the state at estimation times are defined as

$$\mathbf{T}_k = \exp(\boldsymbol{\epsilon}_k^{\wedge})\mathbf{T}_{\mathrm{op},k}, \tag{G.1a}$$

$$\boldsymbol{\varpi}_k = \boldsymbol{\varpi}_{\mathrm{op},k} + \boldsymbol{\eta}_k. \tag{G.1b}$$

In order to compute the interpolation Jacobians, we first need to linearize some expressions contained in (3.31). When we evaluate the local Markovian variable at the endpoints of the local GP, we get the following results,

$$\hat{\boldsymbol{\gamma}}_k(t_k) = \begin{bmatrix} \mathbf{0} \\ \hat{\boldsymbol{\varpi}}_k \end{bmatrix}, \tag{G.2a}$$

$$\hat{\boldsymbol{\gamma}}_k(t_{k+1}) = \begin{bmatrix} \ln\left(\hat{\mathbf{T}}_{k+1}\hat{\mathbf{T}}_k^{-1}\right)^{\vee} \\ \boldsymbol{\mathcal{J}}\left(\ln\left(\hat{\mathbf{T}}_{k+1}\hat{\mathbf{T}}_k^{-1}\right)^{\vee}\right)^{-1}\hat{\boldsymbol{\varpi}}_{k+1} \end{bmatrix}. \tag{G.2b}$$

Next, we linearize

$$\ln\left(\mathbf{T}_{k+1}\mathbf{T}_k^{-1}\right)^{\vee} \approx \ln\left(\mathbf{T}_{\mathrm{op},k+1}\mathbf{T}_{\mathrm{op},k}^{-1}\right)^{\vee} + \boldsymbol{\mathcal{J}}_{\mathrm{op},k+1,k}^{-1}(\boldsymbol{\epsilon}_{k+1} - \boldsymbol{\mathcal{T}}_{\mathrm{op},k+1,k}\boldsymbol{\epsilon}_k), \tag{G.3}$$

where we have assumed that $\boldsymbol{\epsilon}_{k+1} - \boldsymbol{\mathcal{T}}_{\mathrm{op},k+1,k}\boldsymbol{\epsilon}_k$ is small and we have defined

$$\boldsymbol{\mathcal{T}}_{\mathrm{op},k+1,k} = \boldsymbol{\mathcal{T}}_{\mathrm{op},k+1}\boldsymbol{\mathcal{T}}_{\mathrm{op},k}^{-1}, \tag{G.4a}$$

$$\boldsymbol{\mathcal{J}}_{\mathrm{op},k+1,k} = \boldsymbol{\mathcal{J}}\left(\ln(\boldsymbol{\mathcal{T}}_{\mathrm{op},k+1}\boldsymbol{\mathcal{T}}_{\mathrm{op},k}^{-1})^{\vee}\right), \tag{G.4b}$$

and $\boldsymbol{\mathcal{T}}_{\mathrm{op},k} = \mathrm{Ad}(\mathbf{T}_{\mathrm{op},k})$. We also make the following linearization

$$\boldsymbol{\mathcal{J}}\left(\ln\left(\hat{\mathbf{T}}_{k+1}\hat{\mathbf{T}}_k^{-1}\right)^{\vee}\right)^{-1} \approx \boldsymbol{\mathcal{J}}_{\mathrm{op},k+1,k}^{-1} - \frac{1}{2}\left(\boldsymbol{\mathcal{J}}_{\mathrm{op},k+1,k}^{-1}(\boldsymbol{\epsilon}_{k+1} - \boldsymbol{\mathcal{T}}_{\mathrm{op},k+1,k}\boldsymbol{\epsilon}_k)\right)^{\curlywedge}, \tag{G.5}$$

where we have again assumed that $\boldsymbol{\epsilon}_{k+1} - \boldsymbol{\mathcal{T}}_{\text{op},k+1,k}\boldsymbol{\epsilon}_k$ is small and we have approximated the inverse left Jacobian with $\boldsymbol{\mathcal{J}}^{-1}(\mathbf{x}) \approx \mathbf{1} - \frac{1}{2}\mathbf{x}^{\curlywedge}$. The interpolated local variables between estimation times t_k, t_{k+1} are defined as

$$\boldsymbol{\xi}_k(\tau) = \boldsymbol{\Lambda}_1(\tau)\hat{\boldsymbol{\gamma}}_k(t_k) + \boldsymbol{\Psi}_1(\tau)\hat{\boldsymbol{\gamma}}_k(t_{k+1}), \tag{G.6a}$$

$$\dot{\boldsymbol{\xi}}_k(\tau) = \boldsymbol{\Lambda}_2(\tau)\hat{\boldsymbol{\gamma}}_k(t_k) + \boldsymbol{\Psi}_2(\tau)\hat{\boldsymbol{\gamma}}_k(t_{k+1}). \tag{G.6b}$$

The general formula for obtaining the interpolation Jacobians for perturbations to the pose and body-centric velocity is as follows:

$$\frac{\partial \delta\mathbf{T}(\tau)}{\partial \mathbf{x}} = \boldsymbol{\mathcal{J}}_{\text{op},\tau,k}\frac{\partial \boldsymbol{\xi}_k(\tau)}{\partial \mathbf{x}} + \boldsymbol{\mathcal{T}}_{\text{op},\tau,k}\frac{\partial \boldsymbol{\epsilon}_k}{\partial \mathbf{x}}, \tag{G.7a}$$

$$\frac{\partial \delta\boldsymbol{\varpi}(\tau)}{\partial \mathbf{x}} = \boldsymbol{\mathcal{J}}_{\text{op},\tau,k}\frac{\partial \dot{\boldsymbol{\xi}}_k(\tau)}{\partial \mathbf{x}} - \frac{1}{2}\dot{\boldsymbol{\xi}}^{\curlywedge}_{\text{op},\tau}\frac{\partial \boldsymbol{\xi}_k(\tau)}{\partial \mathbf{x}}, \tag{G.7b}$$

where the Jacobians of the local variable $\boldsymbol{\xi}_k(\tau)$ with respect to state perturbations at the bracketing times are given by

$$\frac{\partial \boldsymbol{\xi}_k(\tau)}{\partial \boldsymbol{\epsilon}_{k+1}} = \boldsymbol{\Psi}_{11}\boldsymbol{\mathcal{J}}^{-1}_{\text{op},k+1,k} + \frac{1}{2}\boldsymbol{\Psi}_{12}\boldsymbol{\varpi}^{\curlywedge}_{\text{op},k+1}\boldsymbol{\mathcal{J}}^{-1}_{\text{op},k+1,k}, \tag{G.8a}$$

$$\frac{\partial \boldsymbol{\xi}_k(\tau)}{\partial \boldsymbol{\epsilon}_k} = -\left(\frac{\partial \boldsymbol{\xi}_k(\tau)}{\partial \boldsymbol{\epsilon}_{k+1}}\right)\boldsymbol{\mathcal{T}}_{\text{op},k+1,k}, \tag{G.8b}$$

$$\frac{\partial \boldsymbol{\xi}_k(\tau)}{\partial \boldsymbol{\eta}_k} = \boldsymbol{\Lambda}_{12}, \tag{G.8c}$$

$$\frac{\partial \boldsymbol{\xi}_k(\tau)}{\partial \boldsymbol{\eta}_{k+1}} = \boldsymbol{\Psi}_{12}\boldsymbol{\mathcal{J}}^{-1}_{\text{op},k+1,k}. \tag{G.8d}$$

The Jacobians of $\dot{\boldsymbol{\xi}}_k(\tau)$ have the same form except that we use the second row of the interpolation matrices, $\boldsymbol{\Psi}_{21}$ instead of $\boldsymbol{\Psi}_{11}$, for example.

Bibliography

[1] Oxbotica and Navtech launch Terran360: a world-first radar-based localisation system, howpublished = `http://web.archive.org/web/20080207010024/http://www.808multimedia.com/winnt/kernel.htm`, note = Accessed: 2024-03-24.

[2] Daniel Adolfsson and Maximilian Hilger. An evaluation of CFEAR radar odometry. *arXiv preprint arXiv:2404.01781*, 2024.

[3] Daniel Adolfsson, Martin Magnusson, Anas Alhashimi, Achim J. Lilienthal, and Henrik Andreasson. CFEAR radarodometry - conservative filtering for efficient and accurate radar odometry. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5462–5469, 2021. doi: 10.1109/IROS51168.2021.9636253.

[4] Daniel Adolfsson, Martin Magnusson, Anas Alhashimi, Achim J Lilienthal, and Henrik Andreasson. Lidar-level localization with radar? the CFEAR approach to accurate, fast, and robust large-scale radar odometry in diverse environments. *IEEE Transactions on robotics*, 39 (2):1476–1495, 2022.

[5] Roberto Aldera, Daniele De Martini, Matthew Gadd, and Paul Newman. Fast radar motion estimation with a learnt focus of attention using weak supervision. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1190–1196. IEEE, 2019.

[6] Roberto Aldera, Daniele De Martini, Matthew Gadd, and Paul Newman. What could go wrong? Introspective radar odometry in challenging environments. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2835–2842. IEEE, 2019.

[7] Anas Alhashimi, Daniel Adolfsson, Henrik Andreasson, Achim Lilienthal, and Martin Magnusson. BFAR: improving radar odometry estimation using a bounded false alarm rate detector. *Autonomous Robots*, 48(8):1–16, 2024.

[8] Yasin Almalioglu, Mehmet Turan, Chris Xiaoxuan Lu, Niki Trigoni, and Andrew Markham. Milli-RIO: Ego-motion estimation with low-cost millimetre-wave radar. *IEEE Sensors Journal*, 21(3):3314–3323, 2020.

[9] Sean Anderson and Timothy D Barfoot. RANSAC for motion-distorted 3D visual sensors. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2093–2099. IEEE, 2013.

[10] Sean Anderson and Timothy D Barfoot. Full STEAM ahead: Exactly sparse Gaussian process regression for batch continuous-time trajectory estimation on SE(3). In *2015 IEEE/RSJ*

International Conference on Intelligent Robots and Systems (IROS), pages 157–164. IEEE, 2015.

[11] Sean Anderson, Timothy D Barfoot, Chi Hay Tong, and Simo Särkkä. Batch nonlinear continuous-time trajectory estimation as exactly sparse Gaussian process regression. *Autonomous Robots*, 39(3):221–238, 2015.

[12] Applanix. www.applanix.com, 2022.

[13] K Somani Arun, Thomas S Huang, and Steven D Blostein. Least-squares fitting of two 3-D point sets. *IEEE Transactions on pattern analysis and machine intelligence*, (5):698–700, 1987.

[14] Hernán Badino, Daniel Huber, and Takeo Kanade. Visual topometric localization. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 794–799, 2011. doi: 10.1109/IVS.2011.5940504.

[15] Tim D Barfoot, Chi Hay Tong, and Simo Särkkä. Batch continuous-time trajectory estimation as exactly sparse Gaussian process regression. In *Robotics: Science and Systems*, volume 10, pages 1–10, Berkeley, USA, 12-16 July 2014.

[16] Timothy D Barfoot. *State Estimation for Robotics, 2nd Edition*. Cambridge University Press, 2024.

[17] Timothy D Barfoot, James R Forbes, and David J Yoon. Exactly sparse Gaussian variational inference with application to derivative-free batch nonlinear state estimation. *The International Journal of Robotics Research*, 39(13):1473–1502, 2020.

[18] Dan Barnes and Ingmar Posner. Under the radar: Learning to predict robust keypoints for odometry estimation and metric localisation in radar. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9484–9490, 2020.

[19] Dan Barnes, Matthew Gadd, Paul Murcutt, Paul Newman, and Ingmar Posner. The Oxford Radar RobotCar Dataset: A Radar Extension to the Oxford RobotCar Dataset. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6433–6438, 2020.

[20] Dan Barnes, Rob Weston, and Ingmar Posner. Masking by moving: Learning distraction-free radar odometry from pose information. In *Conference on Robot Learning*, pages 303–316, 2020.

[21] Jens Behley and Cyrill Stachniss. Efficient surfel-based SLAM using 3D laser range data in urban environments. In *Robotics: Science and Systems*, 2018.

[22] Michael Bosse and Robert Zlot. Map matching and data association for large-scale two-dimensional laser scan-based SLAM. *The International Journal of Robotics Research*, 27(6): 667–691, 2008.

[23] Michael Bosse, Robert Zlot, and Paul Flick. Zebedee: Design of a spring-mounted 3-d range sensor with application to mobile mapping. *IEEE Transactions on Robotics*, 28(5):1104–1119, 2012.

[24] Graham Brooker, Mark Bishop, and Steve Scheding. Millimetre waves for robotics. In *Australian Conference for Robotics and Automation*, 2001.

[25] Graham M Brooker, Steven Scheding, Mark V Bishop, and Ross C Hennessy. Development and application of millimeter wave radar sensors for underground mining. *IEEE Sensors journal*, 5(6):1270–1280, 2005.

[26] Martin Brossard, Axel Barrau, Paul Chauchat, and Silvére Bonnabel. Associating uncertainty to extended poses for on Lie group IMU preintegration with rotating earth. *IEEE Transactions on Robotics*, 38(2):998–1015, 2022. doi: 10.1109/TRO.2021.3100156.

[27] Keenan Burnett. Radar to Lidar Calibrator, 2020.

[28] Keenan Burnett, Angela P Schoellig, and Timothy D Barfoot. Do we need to compensate for motion distortion and Doppler effects in spinning radar navigation? *IEEE Robotics and Automation Letters*, 6(2):771–778, 2021.

[29] Keenan Burnett, David J Yoon, Angela P Schoellig, and Timothy D Barfoot. Radar odometry combining probabilistic estimation and unsupervised feature learning. In *Robotics: Science and Systems (RSS)*, 2021.

[30] Keenan Burnett, Yuchen Wu, David J Yoon, Angela P Schoellig, and Timothy D Barfoot. Are we ready for radar to replace lidar in all-weather mapping and localization? *IEEE Robotics and Automation Letters*, 7(4):10328–10335, 2022.

[31] Keenan Burnett, David J Yoon, Yuchen Wu, Andrew Z Li, Haowei Zhang, Shichen Lu, Jingxing Qian, Wei-Kang Tseng, Andrew Lambert, Keith YK Leung, et al. Boreas: A multi-season autonomous driving dataset. *The International Journal of Robotics Research*, 42(1-2):33–42, 2023.

[32] Keenan Burnett, Angela P. Schoellig, and Timothy D. Barfoot. Continuous-time radar-inertial and lidar-inertial odometry using a Gaussian process motion prior. *IEEE Transactions on Robotics*, 41:1059–1076, 2025. doi: 10.1109/TRO.2024.3521856.

[33] Keenan Burnett, Angela P. Schoellig, and Timothy D. Barfoot. Imu as an input versus a measurement of the state in inertial-aided state estimation. *Robotica*, page 1–21, 2025. doi: 10.1017/S0263574724002121.

[34] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11621–11631, 2020.

[35] Jonas Callmer, David Törnqvist, Fredrik Gustafsson, Henrik Svensson, and Pelle Carlbom. Radar SLAM using visual features. *EURASIP Journal on Advances in Signal Processing*, 2011.

[36] Sarah H Cen and Paul Newman. Precise ego-motion estimation with millimeter-wave radar under diverse and challenging conditions. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018.

[37] Sarah H Cen and Paul Newman. Radar-only ego-motion estimation in difficult settings via graph matching. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 298–304. IEEE, 2019.

[38] Manjari Chandran and Paul Newman. Motion estimation from map quality with millimeter wave radar. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 808–813, 2006. doi: 10.1109/IROS.2006.281673.

[39] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3D tracking and forecasting with rich maps. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8748–8757, 2019.

[40] Nicholas Charron, Stephen Phillips, and Steven L Waslander. De-noising of Lidar Point Clouds Corrupted by Snowfall. In *2018 15th Conference on Computer and Robot Vision (CRV)*, pages 254–261, 2018.

[41] Paul Checchin, Franck Gérossier, Christophe Blanc, Roland Chapuis, and Laurent Trassoudaine. Radar scan matching SLAM using the Fourier-Mellin transform. In *Field and Service Robotics: Results of the 7th International Conference*, pages 151–161. Springer, 2010.

[42] Hongyu Chen, Yimin Liu, and Yuwei Cheng. DRIO: Robust radar-inertial odometry in dynamic environments. *IEEE Robotics and Automation Letters*, 2023.

[43] Kenny Chen, Ryan Nemiroff, and Brett T Lopez. Direct lidar-inertial odometry and mapping: Perceptive and connective SLAM. *arXiv preprint arXiv:2305.01843*, 2023.

[44] Kenny Chen, Ryan Nemiroff, and Brett T Lopez. Direct lidar-inertial odometry: Lightweight lio with continuous-time motion correction. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3983–3989. IEEE, 2023.

[45] Xieyuanli Chen, Andres Milioto, Emanuele Palazzolo, Philippe Giguère, Jens Behley, and Cyrill Stachniss. SuMa++: Efficient lidar-based semantic SLAM. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4530–4537, 2019. doi: 10.1109/IROS40897.2019.8967704.

[46] Steve Clark and Hugh Durrant-Whyte. Autonomous land vehicle navigation using millimeter wave radar. In *1998 IEEE Conference on Robotics and Automation (ICRA)*, volume 4, pages 3697–3702. IEEE, 1998.

[47] Daniele De Martini, Matthew Gadd, and Paul Newman. kRadar++: Coarse-to-Fine FMCW Scanning Radar Localisation. *Sensors*, 20(21):6002, 2020.

[48] Pierre Dellenbach, Jean-Emmanuel Deschaud, Bastien Jacquet, and François Goulette. CT-ICP: Real-time elastic lidar odometry with loop closure. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 5580–5586. IEEE, 2022.

[49] Jean-Emmanuel Deschaud. IMLS-SLAM: Scan-to-model matching based on 3D data. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2480–2485, 2018.

[50] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Self-improving visual odometry. *arXiv preprint arXiv:1812.03245*, 2018.

[51] Gamini Dissanayake, Paul Newman, Steve Clark, Hugh F Durrant-Whyte, and Michael Csorba. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on robotics and automation*, 17(3):229–241, 2001.

[52] Christopher Doer and Gert F Trommer. Radar visual inertial odometry and radar thermal inertial odometry: Robust navigation even in challenging visual conditions. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 331–338. IEEE, 2021.

[53] David Droeschel and Sven Behnke. Efficient continuous-time SLAM for 3D lidar-based online mapping. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5000–5007. IEEE, 2018.

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

[55] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. On-manifold preintegration for real-time visual–inertial odometry. *IEEE Transactions on Robotics*, 33(1):1–21, 2017. doi: 10.1109/TRO.2016.2597321.

[56] Paul Furgale and Timothy D Barfoot. Visual teach and repeat for long-range rover autonomy. *Journal of field robotics*, 27(5):534–560, 2010.

[57] Paul Furgale, Chi Hay Tong, Timothy D Barfoot, and Gabe Sibley. Continuous-time batch trajectory estimation using temporal basis functions. *The International Journal of Robotics Research*, 34(14):1688–1710, 2015.

[58] Matthew Gadd, Daniele De Martini, and Paul Newman. Look around you: Sequence-based radar place recognition with learned rotational invariance. In *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pages 270–276, 2020.

[59] Pengen Gao, Shengkai Zhang, Wei Wang, and Chris Xiaoxuan Lu. Accurate automotive radar based metric localization with explicit Doppler compensation. *arXiv preprint arXiv:2112.14887*, 2021.

[60] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *CVPR*, 2012.

[61] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[62] Zoubin Ghahramani and Geoffrey E Hinton. Parameter Estimation for Linear Dynamical Systems. Technical Report CRG-TR-96-2, U. of Toronto, 1996.

[63] Zoubin Ghahramani and Sam T Roweis. Learning nonlinear dynamical systems using an EM algorithm. In *Advances in Neural Information Processing Systems*, 1999.

[64] Mona Gridseth and Timothy D. Barfoot. Keeping an eye on things: Deep learned features for long-term visual localization. *IEEE Robotics and Automation Letters*, 7(2):1016–1023, 2022. doi: 10.1109/LRA.2021.3136867.

[65] Kyle Harlow, Hyesu Jang, Timothy D. Barfoot, Ayoung Kim, and Christoffer Heckman. A new wave in robotics: Survey on recent mmwave radar applications in robotics. *IEEE Transactions on Robotics*, 40:4544–4560, 2024.

[66] Johan Hedborg, Per-Erik Forssén, Michael Felsberg, and Erik Ringaby. Rolling shutter bundle adjustment. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1434–1441. IEEE, 2012.

[67] Michael Helmberger, Kristian Morin, Beda Berner, Nitish Kumar, Giovanni Cioffi, and Davide Scaramuzza. The Hilti SLAM challenge dataset. *IEEE Robotics and Automation Letters*, 7 (3):7518–7525, 2022.

[68] Daniel Casado Herraez, Matthias Zeller, Le Chang, Ignacio Vizzo, Michael Heidingsfeld, and Cyrill Stachniss. Radar-only odometry and mapping for autonomous vehicles. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10275–10282, 2024.

[69] Martin Holder, Sven Hellwig, and Hermann Winner. Real-time pose graph SLAM based on radar. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1145–1151, 2019. doi: 10.1109/IVS.2019.8813841.

[70] Ziyang Hong, Yvan Petillot, and Sen Wang. RadarSLAM: Radar based large-scale SLAM in all weathers. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5164–5170. IEEE, 2020.

[71] Ziyang Hong, Yvan Petillot, Andrew Wallace, and Sen Wang. RadarSLAM: A robust simultaneous localization and mapping system for all weather conditions. *The international journal of robotics research*, 41(5):519–542, 2022.

[72] Jui-Te Huang, Ruoyang Xu, Akshay Hinduja, and Michael Kaess. Multi-radar inertial odometry for 3D state estimation using mmwave imaging radar. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 12006–12012, 2024.

[73] Xinyu Huang, Xinjing Cheng, Qichuan Geng, Binbin Cao, Dingfu Zhou, Peng Wang, Yuanqing Lin, and Ruigang Yang. The ApolloScape dataset for autonomous driving. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 954–960, 2018.

[74] Jinyong Jeong, Younggun Cho, Young-Sik Shin, Hyunchul Roh, and Ayoung Kim. Complex urban dataset with multi-level sensors from highly diverse urban environments. *The International Journal of Robotics Research*, 38(6):642–657, 2019.

[75] Jacob Johnson, Joshua Mangelson, Timothy Barfoot, and Randal Beard. Continuous-time trajectory estimation: A comparative study between Gaussian process and spline-based approaches. *arXiv preprint arXiv:2402.00399*, 2024.

[76] Matthew James Johnson, David Duvenaud, Alexander B Wiltschko, Sandeep R Datta, and Ryan P Adams. Composing graphical models with neural networks for structured representations and fast inference. *Advances in Neural Information Processing Systems*, 2016.

[77] Ebi Jose and Martin David Adams. Relative radar cross section based feature identification with millimeter wave radar for outdoor SLAM. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 425–430. IEEE, 2004.

[78] Dominik Kellner, Michael Barjenbruch, Jens Klappstein, Jürgen Dickmann, and Klaus Dietmayer. Instantaneous ego-motion estimation using Doppler radar. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pages 869–874. IEEE, 2013.

[79] Giseop Kim, Yeong Sang Park, Younghun Cho, Jinyong Jeong, and Ayoung Kim. MulRan: Multimodal Range Dataset for Urban Place Recognition. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6246–6253, 2020.

[80] Diederik Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations*, 2013.

[81] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[82] Andrew Kramer, Carl Stahoviak, Angel Santamaria-Navarro, Ali-akbar Agha-mohammadi, and Christoffer Heckman. Radar-inertial ego-velocity estimation for visually degraded environments. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5739–5746. IEEE, 2020.

[83] Philipp Krüsi, Bastian Bücheler, François Pomerleau, Ulrich Schwesinger, Roland Siegwart, and Paul Furgale. Lighting-invariant adaptive route following using iterative closest point matching. *JFR*, 2015.

[84] Vladimír Kubelka, Emil Fritz, and Martin Magnusson. Do we need scan-matching in radar odometry? In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13710–13716, 2024.

[85] Pou-Chun Kung, Chieh-Chih Wang, and Wen-Chieh Lin. A Normal Distribution Transform-Based Radar Odometry Designed For Scanning and Automotive Radars. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021.

[86] Xiaolei Lang, Chao Chen, Kai Tang, Yukai Ma, Jiajun Lv, Yong Liu, and Xingxing Zuo. Coco-LIC: Continuous-time tightly-coupled lidar-inertial-camera odometry using non-uniform b-spline. *IEEE Robotics and Automation Letters*, 2023.

[87] Cedric Le Gentil and Teresa Vidal-Calleja. Continuous latent state preintegration for inertial-aided systems. *The International Journal of Robotics Research*, 42(10):874–900, 2023.

[88] Cedric Le Gentil, Teresa Vidal-Calleja, and Shoudong Huang. IN2LAAMA: Inertial lidar localization autocalibration and mapping. *IEEE Transactions on Robotics*, 37(1):275–290, 2020.

[89] Dongjae Lee, Minwoo Jung, Wooseong Yang, and Ayoung Kim. Lidar odometry survey: recent advancements and remaining challenges. *Intelligent Service Robotics*, 17(2):95–118, 2024.

[90] Jesse Levinson and Sebastian Thrun. Robust vehicle localization in urban environments using probabilistic maps. In *2010 IEEE International Conference on Robotics and Automation*, pages 4372–4378, 2010. doi: 10.1109/ROBOT.2010.5509700.

[91] Siru Li, Ziyang Hong, Yushuai Chen, Liang Hu, and Jiahu Qin. Get it for free: Radar segmentation without expert labels and its application in odometry and localization. *arXiv preprint arXiv:2409.18434*, 2024.

[92] Katherine Liu, Kyel Ok, William Vega-Brown, and Nicholas Roy. Deep inference for covariance estimation: Learning Gaussian noise models for state estimation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1436–1443, 2018. doi: 10.1109/ICRA.2018.8461047.

[93] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[94] Chris Xiaoxuan Lu, Muhamad Risqi U Saputra, Peijun Zhao, Yasin Almalioglu, Pedro PB de Gusmao, Changhao Chen, Ke Sun, Niki Trigoni, and Andrew Markham. milliEgo: single-chip mmwave radar aided egomotion estimation via deep sensor fusion. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, pages 109–122, 2020.

[95] Todd Lupton and Salah Sukkarieh. Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions. *IEEE Transactions on Robotics*, 28(1):61–76, 2012. doi: 10.1109/TRO.2011.2170332.

[96] Jiajun Lv, Xiaolei Lang, Jinhong Xu, Mengmeng Wang, Yong Liu, and Xingxing Zuo. Continuous-time fixed-lag smoothing for lidar-inertial-camera SLAM. *IEEE/ASME Transactions on Mechatronics*, 2023.

[97] Will Maddern, Geoffrey Pascoe, Chris Linegar, and Paul Newman. 1 Year, 1000 km: The Oxford RobotCar dataset. *The International Journal of Robotics Research*, 36(1):3–15, 2017.

[98] Mathworks. mathworks.com, 2022.

[99] Jan Michalczyk, Roland Jung, and Stephan Weiss. Tightly-coupled EKF-based radar-inertial odometry. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12336–12343. IEEE, 2022.

[100] Jan Michalczyk, Roland Jung, Christian Brommer, and Stephan Weiss. Multi-state tightly-coupled EKF-based radar-inertial odometry with persistent landmarks. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4011–4017. IEEE, 2023.

[101] Michael J Milford and Gordon F Wyeth. SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights. In *2012 IEEE international conference on robotics and automation*, pages 1643–1649. IEEE, 2012.

[102] John Mullane, Ba-Ngu Vo, Martin D. Adams, and Ba-Tuong Vo. A random-finite-set approach to bayesian SLAM. *IEEE Transactions on Robotics*, 27(2):268–282, 2011. doi: 10.1109/TRO.2010.2101370.

[103] John Mullane, Samuel Keller, and Martin Adams. Random set versus vector based SLAM in the presence of high clutter. In *IEEE ICRA*, 2012.

[104] Radford M Neal and Geoffrey E Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in Graphical Models*, pages 355–368. Springer, 1998.

[105] Ryan Nemiroff, Kenny Chen, and Brett T Lopez. Joint on-manifold gravity and accelerometer intrinsics estimation. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023.

[106] Yin Zhi Ng, Benjamin Choi, Robby Tan, and Lionel Heng. Continuous-time radar-inertial odometry for automotive radars. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 323–330. IEEE, 2021.

[107] Thien-Minh Nguyen, Shenghai Yuan, Muqing Cao, Yang Lyu, Thien H Nguyen, and Lihua Xie. NTU Viral: A visual-inertial-ranging-lidar dataset, from an aerial vehicle viewpoint. *The International Journal of Robotics Research*, 41(3):270–280, 2022.

[108] Thien-Minh Nguyen, Daniel Duberg, Patric Jensfelt, Shenghai Yuan, and Lihua Xie. Slict: Multi-input multi-scale surfel-based lidar-inertial continuous-time odometry and mapping. *IEEE Robotics and Automation Letters*, 8(4):2102–2109, 2023.

[109] United States Department of Transportation National Highway Traffic Safety Administration. Drowsy driving 2015. 2017.

[110] United States Department of Transportation National Highway Traffic Safety Administration. Distracted driving in 2021. 2023.

[111] United States Department of Transportation National Highway Traffic Safety Administration. Overview of motor vehicle traffic crashes in 2021. 2023.

[112] Chanoh Park, Peyman Moghadam, Jason L Williams, Soohwan Kim, Sridha Sridharan, and Clinton Fookes. Elasticity meets continuous-time: Map-centric dense 3D lidar SLAM. *IEEE Transactions on Robotics*, 38(2):978–997, 2021.

[113] Yeong Sang Park, Joowan Kim, and Ayoung Kim. Radar localization and mapping for indoor disaster environments via multi-modal registration to prior lidar map. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1307–1314, 2019. doi: 10.1109/IROS40897.2019.8967633.

[114] Yeong Sang Park, Young-Sik Shin, and Ayoung Kim. PhaRaO: Direct radar odometry using phase correlation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2617–2623, 2020.

[115] Yeong Sang Park, Young-Sik Shin, Joowan Kim, and Ayoung Kim. 3D ego-motion estimation using low-cost mmwave radars via radar velocity factor for pose-graph SLAM. *IEEE Robotics and Automation Letters*, 6(4):7691–7698, 2021. doi: 10.1109/LRA.2021.3099365.

[116] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems*, 32:8026–8037, 2019.

[117] Michael Paton, Kirk MacTavish, Michael Warren, and Timothy D. Barfoot. Bridging the appearance gap: Multi-experience localization for long-term visual teach and repeat. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1918–1925, 2016. doi: 10.1109/IROS.2016.7759303.

[118] Michael Paton, Kirk MacTavish, Laszlo-Peter Berczi, Sebastian Kai van Es, and Timothy D Barfoot. I can see for miles and miles: An extended field test of visual teach and repeat 2.0. In *Field and Service Robotics: Results of the 11th International Conference*, pages 415–431. Springer, 2018.

[119] Alonso Patron-Perez, Steven Lovegrove, and Gabe Sibley. A spline-based trajectory representation for sensor fusion and rolling shutter cameras. *International Journal of Computer Vision*, 113(3):208–219, 2015.

[120] Matthew Pitropov, Danson Evan Garcia, Jason Rebello, Michael Smart, Carlos Wang, Krzysztof Czarnecki, and Steven Waslander. Canadian adverse driving conditions dataset. *The International Journal of Robotics Research*, 40(4-5):681–690, 2021.

[121] François Pomerleau, Francis Colas, Roland Siegwart, and Stéphane Magnenat. Comparing ICP variants on real-world data sets: Open-source library and experimental protocol. *Autonomous robots*, 34:133–148, 2013.

[122] Jan Quenzel and Sven Behnke. Real-time multi-adaptive-resolution-surfel 6D lidar odometry using continuous-time trajectory optimization. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5499–5506. IEEE, 2021.

[123] Milad Ramezani, Yiduo Wang, Marco Camurri, David Wisth, Matias Mattamala, and Maurice Fallon. The newer college dataset: Handheld lidar, inertial and vision with ground truth. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4353–4360. IEEE, 2020.

[124] Matthias Rapp, Michael Barjenbruch, Markus Hahn, Jürgen Dickmann, and Klaus Dietmayer. Probabilistic ego-motion estimation using multiple automotive radar sensors. *Robotics and Autonomous Systems*, 2017.

[125] Hermann Rohling. Radar CFAR thresholding in clutter and multiple target situations. *IEEE transactions on aerospace and electronic systems*, (4):608–621, 1983.

[126] Hermann Rohling. Ordered statistic CFAR technique-an overview. In *2011 12th International Radar Symposium (IRS)*, pages 631–638. IEEE, 2011.

[127] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Comp.-Assisted Intervention*, 2015.

[128] R Rouveure, MO Monod, and P Faure. High resolution mapping of the environment with a ground-based radar imager. In *RADAR*. IEEE, 2009.

[129] Raphaël Rouveure, Patrice Faure, and Marie-Odile Monod. PELICAN: Panoramic millimeter-wave radar for perception in mobile robotics applications, part 1: Principles of FMCW radar and of 2D image construction. *Robotics and Autonomous Systems*, 81:1–16, 2016. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2016.04.001.

[130] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 International conference on computer vision*, pages 2564–2571. IEEE, 2011.

[131] Ştefan Săftescu, Matthew Gadd, Daniele De Martini, Dan Barnes, and Paul Newman. Kidnapped radar: Topological radar localisation using rotationally-invariant metric learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4358–4364, 2020.

[132] Scale. www.scale.com, 2022.

[133] F. Schuster, C. G. Keller, M. Rapp, M. Haueis, and C. Curio. Landmark based radar SLAM using graph optimization. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 2559–2564, 2016. doi: 10.1109/ITSC.2016.7795967.

[134] Aleksandr Segal, Dirk Haehnel, and Sebastian Thrun. Generalized-ICP. In *Robotics: science and systems*, page 435. Seattle, WA, 2009.

[135] Tixiao Shan, Brendan Englot, Drew Meyers, Wei Wang, Carlo Ratti, and Daniela Rus. LIO-SAM: Tightly-coupled lidar inertial odometry via smoothing and mapping. In *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 5135–5142. IEEE, 2020.

[136] Marcel Sheeny, Emanuele De Pellegrin, Saptarshi Mukherjee, Alireza Ahrabian, Sen Wang, and Andrew Wallace. RADIATE: A radar dataset for automotive perception in bad weather. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–7, 2021.

[137] Robert H Shumway and David S Stoffer. An approach to time series smoothing and forecasting using the EM algorithm. *Journal of Time Series Analysis*, 3(4):253–264, 1982.

[138] Merrill I Skolnik. Radar handbook second edition. *McGrawHill*, 1990.

[139] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 573–580. IEEE, 2012.

[140] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2446–2454, 2020.

[141] Andrea Tagliabue, Jesus Tordesillas, Xiaoyi Cai, Angel Santamaria-Navarro, Jonathan P How, Luca Carlone, and Ali-akbar Agha-mohammadi. LION: Lidar-inertial observability-aware navigator for vision-denied environments. In *Experimental Robotics: The 17th International Symposium*, pages 380–390. Springer, 2021.

[142] William Talbot, Julian Nubert, Turcan Tuna, Cesar Cadena, Frederike Dümbgen, Jesus Tordesillas, Timothy D. Barfoot, and Marco Hutter. Continuous-time state estimation methods in robotics: A survey. *arXiv preprint arXiv:2411.03951*, 2024.

[143] Tim Y Tang, David J Yoon, and Timothy D Barfoot. A white-noise-on-jerk motion prior for continuous-time trajectory estimation on se(3). *IEEE Robotics and Automation Letters*, 4(2): 594–601, 2019. doi: 10.1109/LRA.2019.2891492.

[144] Tim Y Tang, Daniele De Martini, Shangzhe Wu, and Paul Newman. Self-Supervised Localisation between Range Sensors and Overhead Imagery. In *2020 Robotics: Science and Systems*, 2020.

[145] Tim Y Tang, Daniele De Martini, and Paul Newman. Get to the point: Learning lidar place recognition and metric localisation using overhead imagery. In *Robotics: Science and Systems*, 2021.

[146] Tim Yuqing Tang, Daniele De Martini, Dan Barnes, and Paul Newman. RSL-Net: Localising in satellite images from a radar on the ground. *IEEE Robotics and Automation Letters*, 5(2): 1087–1094, 2020.

[147] Shinji Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 13(04):376–380, 1991.

[148] Understand. Understand.ai Anonymizer, 2022.

[149] Damien Vivet, Paul Checchin, and Roland Chapuis. Localization and mapping using only a rotating FMCW radar sensor. *Sensors*, 13(4):4527–4552, 2013.

[150] Damien Vivet, Franck Gérossier, Paul Checchin, Laurent Trassoudaine, and Roland Chapuis. Mobile ground-based radar sensor for localization and mapping: An evaluation of two approaches. *International Journal of Advanced Robotic Systems*, 10(8):307, 2013.

[151] Ignacio Vizzo, Tiziano Guadagnino, Benedikt Mersch, Louis Wiesmann, Jens Behley, and Cyrill Stachniss. Kiss-icp: In defense of point-to-point icp–simple, accurate, and robust registration if done the right way. *IEEE Robotics and Automation Letters*, 8(2):1029–1036, 2023.

[152] Ryan W. Wolcott and Ryan M. Eustice. Fast lidar localization using multiresolution Gaussian mixture maps. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2814–2821, 2015. doi: 10.1109/ICRA.2015.7139582.

[153] Jeremy N Wong, David J Yoon, Angela P Schoellig, and Timothy D Barfoot. A data-driven motion prior for continuous-time trajectory estimation on se(3). *IEEE Robotics and Automation Letters*, 5(2):1429–1436, 2020. doi: 10.1109/LRA.2020.2969153.

[154] Jeremy N Wong, David J Yoon, Angela P Schoellig, and Timothy D Barfoot. Variational inference with parameter learning applied to vehicle trajectory estimation. *IEEE Robotics and Automation Letters*, 5(4):5291–5298, 2020.

[155] Yuchen Wu, David J Yoon, Keenan Burnett, Soeren Kammel, Yi Chen, Heethesh Vhavle, and Timothy D Barfoot. Picking up speed: Continuous-time lidar-only odometry using Doppler velocity measurements. *IEEE Robotics and Automation Letters*, 8(1):264–271, 2022.

[156] Wei Xu, Yixi Cai, Dongjiao He, Jiarong Lin, and Fu Zhang. Fast-lio2: Fast direct lidar-inertial odometry. *IEEE Transactions on Robotics*, 38(4):2053–2073, 2022.

[157] Haoyang Ye, Yuying Chen, and Ming Liu. Tightly coupled 3D lidar inertial odometry and mapping. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3144–3150. IEEE, 2019.

[158] Huan Yin, Yue Wang, Li Tang, and Rong Xiong. Radar-on-lidar: metric radar localization on prior lidar maps. In *2020 IEEE International Conference on Real-time Computing and Robotics (RCAR)*. IEEE, 2020.

[159] Huan Yin, Runjian Chen, Yue Wang, and Rong Xiong. Rall: end-to-end radar localization on lidar map using differentiable measurement model. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[160] David J. Yoon, Haowei Zhang, Mona Gridseth, Hugues Thomas, and Timothy D. Barfoot. Unsupervised learning of lidar features for use ina probabilistic trajectory estimator. *IEEE Robotics and Automation Letters*, 6(2):2130–2138, 2021. doi: 10.1109/LRA.2021.3060407.

[161] Ji Zhang and Sanjiv Singh. Low-drift and real-time lidar odometry and mapping. *Autonomous Robots*, 41:401–416, 2017.

[162] Xin Zheng and Jianke Zhu. Traj-LO: In defense of lidar-only odometry using an effective continuous-time trajectory. *IEEE Robotics and Automation Letters*, 9(2):1961–1968, 2024.

[163] Xin Zheng and Jianke Zhu. Traj-LIO: A resilient multi-lidar multi-IMU state estimator through sparse Gaussian process. *arXiv preprint arXiv:2402.09189*, 2024.

[164] Yuan Zhuang, Binliang Wang, Jianzhu Huai, and Miao Li. 4D iRIOM: 4D imaging radar inertial odometry and mapping. *IEEE Robotics and Automation Letters*, 2023.