Continuous-Time Radar-Inertial and Lidar-Inertial Odometry using a Gaussian Process Motion Prior

Keenan Burnett, *Graduate Student Member, IEEE*, Angela P. Schoellig, *Member, IEEE*,
Timothy D. Barfoot, *Fellow, IEEE*

Abstract—In this work, we demonstrate continuous-time radar-inertial and lidar-inertial odometry using a Gaussian process motion prior. Using a sparse prior, we demonstrate improved computational complexity during preintegration and interpolation. We use a white-noise-on-acceleration motion prior and treat the gyroscope as a direct measurement of the state while preintegrating accelerometer measurements to form relative velocity factors. Our odometry is implemented using sliding-window batch trajectory estimation. To our knowledge, our work is the first to demonstrate radar-inertial odometry with a spinning mechanical radar using both gyroscope and accelerometer measurements. We improve the performance of our radar odometry by 43% by incorporating an IMU. Our approach is efficient and we demonstrate real-time performance. Code for this paper can be found at: https://github.com/utiasASRL/steam_icp

Index Terms—Localization, Range Sensing, Mapping, Continuous-Time

I. INTRODUCTION

IN large-scale outdoor mapping and localization, focus has shifted towards improving robustness and reliability under challenging conditions such as sparse or degenerate geometry, aggressive motion, and adverse weather. Radar is a promising alternative to lidar as its longer wavelength enables it to be robust to dust, fog, rain, and snow. Although it has been shown in prior work that lidar localization can still function under moderate levels of precipitation [1], it is possible that radar will perform better than lidar under more extreme weather conditions. Furthermore, radar-based localization may still prove valuable as a redundant backup system in safety-critical applications. Our goal in this work 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 so as to address adverse weather conditions. Our secondary goal is to reduce the performance gap between radar and lidar odometry through the incorporation of 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
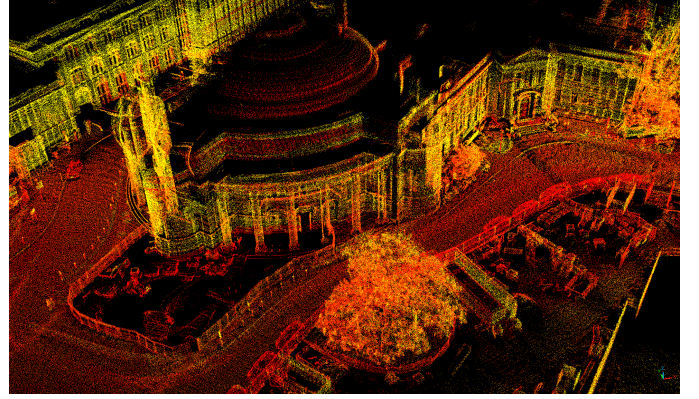
Fig. 1. A lidar map generated of the University of Toronto obtained during a sequence of the Boreas dataset. This high-quality map is generated as a byproduct of our odometry pipeline. The pointcloud is colored by intensity.

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 the high rate of IMUs, the number of measurement times can become quite large and consequently the computational requirements can become too expensive for real-time operation.

In order to address this problem, Lupton and Sukkarieh [2] proposed to preintegrate IMU measurements between pairs of consecutive camera measurements in order to combine them into a single relative motion factor. This significantly improves runtime since we now only need to estimate the state at each camera measurement. Forster et al. [3] then showed how to perform on-manifold preintegration within the space of 3D rotations, $SO(3)$. Recently, Brossard et al. [4] 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 motion of the robot causes the pointclouds to become distorted due to the scanning-while-moving nature of the sensor. In our previous work, we addressed this motion-distortion effect

using continuous-time point-to-point factors and a Gaussian process motion prior [1]. This allowed us to estimate the pose and body-centric velocity of the sensor while simultaneously 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 [5] when the motion of a robot is relatively smooth, such as in the case of heavy ground robots. However, when working with highly dynamic motions, such as in the case of drones or walking robots, continuous-time approaches present considerable advantages. We will demonstrate this in the experimental results section using the Newer College Dataset.

Our aim is 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 work, we choose to use a white-noise-on-acceleration motion prior [6]. 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 [7] or the Singer prior [8], both of which include body-centric acceleration in the state. In our experiments, we observed 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 and thus we opted to not include it.

Another work that employed Gaussian processes in continuous-time state estimation is that of Le Gentil and Vidal-Calleja [9]. 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, and 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 a period

of IMU measurement dropout by falling back on the Gaussian process motion prior. We consider our approach to be tightly coupled since we include IMU measurements in the pointcloud alignment optimization directly whereas their approach uses the IMU to first undistort the scans prior to the alignment optimization. Another important difference is how we compensate for motion distortion. [9] 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 1, we provide an example of a 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 [10], [11]. Also, our framework supports adding additional continuous-time measurement factors such as wheel odometry and Doppler velocity measurements [12]. 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. To summarize, our contributions are 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 [13], Boreas [14] and the Newer College Dataset [15].
- We demonstrate radar-inertial odometry with a spinning mechanical radar using both gyroscope and accelerometer measurements. To our knowledge, this has not been previously demonstrated in the literature.
- We provide a detailed comparison of the performance of lidar-inertial and radar-inertial odometry across varying seasonal and weather conditions.

II. RELATED WORK

For a detailed literature review on lidar odometry and lidar-inertial odometry, we refer readers to the recent survey by Lee et al. [16].

A. 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 [17]. Due to the large number of points produced by modern lidar sensors ($\sim$100k points per scan), these methods incur a heavy computational load. In order to enable real-time operation, recent methods rely on coarse voxelization and efficient data structures for map storage and retrieval [5], [10], [11], [18]. 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,
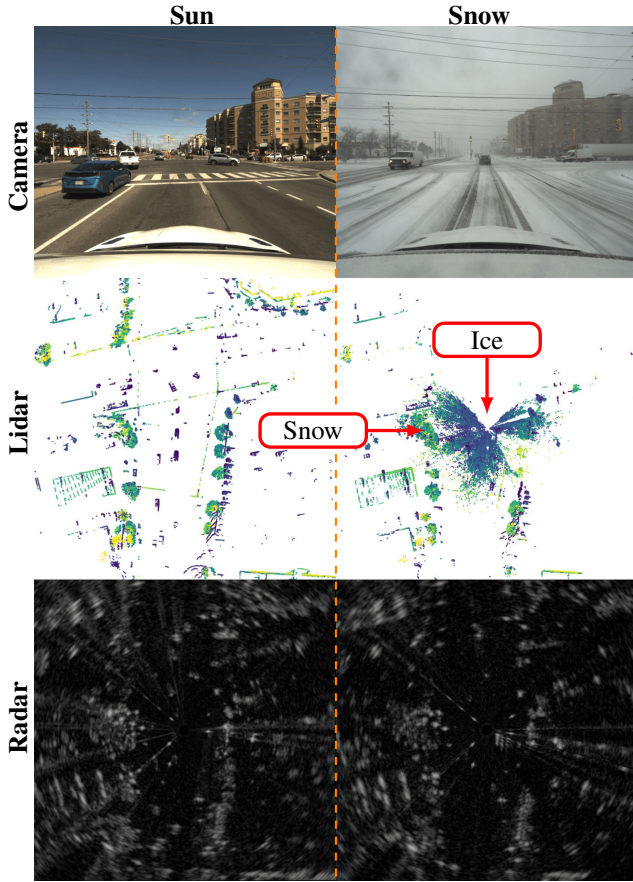
Fig. 2. This figure illustrates the differences in camera, lidar, and radar data during a sunny day and during a snowstorm. The lidar data here is colored by elevation. During snowfall, the lidar data becomes corrupted with noise and a section becomes blocked by ice but the radar data appears unaffected.

point-to-plane, and Generalized ICP [17], [19]. Our work can be considered a direct method since we do not make use of feature extraction and matching.

Examples of feature-based methods include LOAM [20], which matches edge and plane features, and SuMa++ [21], 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.

B. 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 [20]. 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 incorporate IMU data into the pointcloud alignment optimization directly.

LOAM [20] 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 [22] and LION [23] 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 [11] is a recent example of a loosely coupled approach where an IMU is used to undistort lidar data. Preintegrated IMU measurements are then subsequently combined with pose estimates from ICP using a hierarchical geometric observer. LIOM [24] is an example of a tightly coupled lidar-inertial odometry using a factor graph. FAST-LIO2 [10] is another tightly coupled approach that uses an iterated extended Kalman filter. In our tightly coupled approach, we combine continuous-time point-to-plane factors with direct gyroscope factors and preintegrated velocity factors using a fixed-lag smoother and a Gaussian process motion prior.

Even after incorporating an IMU, some challenges remain such as handling harsh enviromental conditions such as dust, fog, rain, and snow that can adversely affect lidar data. Figure 2 depicts an example where lidar data is affected by snow and ice build-up. In order to tackle these problems, radar is being investigated as a potential alternative to lidar.

C. Radar Odometry

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 in order to boost the signal-to-noise ratio between the detected target power and background clutter power [25], [26]. Due to the noise inherent to radar, it is challenging to directly apply methods designed for lidar.

Several methods have been proposed to deal with the high amount of noise in radar measurements. Jose and Adams [27] proposed to estimate the probability of target presence and to include radar cross section in their simultaneous localization and mapping (SLAM) setup. Checchin et al. [28] proposed to densely match radar scans using the Fourier-Mellin transform, a correlative scan-matching approach. Vivet et al. [29] and Kellner et al. [30] proposed to use the relative Doppler velocity measurements to estimate the instantaneous egomotion. Callmer et al. [31] proposed to leverage features originally designed for vision to enable landmark-based SLAM. Schuster et al. [32] subsequently refined this approach by designing bespoke radar feature descriptors. Cen and Newman demonstrated accurate large-scale radar odometry using a spinning mechanical radar [33]. Hong et al. [34] demonstrated large-scale radar SLAM in all weather conditions. Recently, there has been a resurgence of research into improving and refining radar-based localization. Currently, the state of the art for radar odometry 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 [35]. For a more detailed review of radar-based localization, we refer readers to the survey by Harlow et al. [36].

Recent works have benefited from advancements in radar sensors where frequency modulated continuous wave (FMCW)

radar sensors now possess improved range resolution, azimuth resolution, and some even provide target elevation in 3D. Furthermore, many FMCW radar sensors also provide a relative Doppler velocity measurement for each target. However, these new radar sensors still produce sparse and noisy pointclouds. As such, there has been a concerted effort to develop and improve radar-inertial odometry to address these challenges.

D. Radar-Inertial Odometry

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 [37]–[44]. Examples of prior work using automotive radar include [45]–[47]. The work that is closest to ours is that of Ng et al. [45], 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 whereas their approach uses only the Doppler velocity measurements from each sensor in conjunction with an IMU. The radar that we use in this work 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 [12]. 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 [48]–[50] to compensate for motion distortion and as a prediction step in a filter while our work fuses both gyroscope and accelerometer measurements with point-to-point radar factors using our continuous-time framework.

The radar and lidar sensors used in this work both rely on mechanical actuation to cover the entire field of view around the robot. 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 3 illustrates the high-rate and asynchronous nature of the sensor measurements.

E. Continuous-Time State Estimation

There are two main classes of continuous-time approaches, parametric approaches relying 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 samples of the trajectory. 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
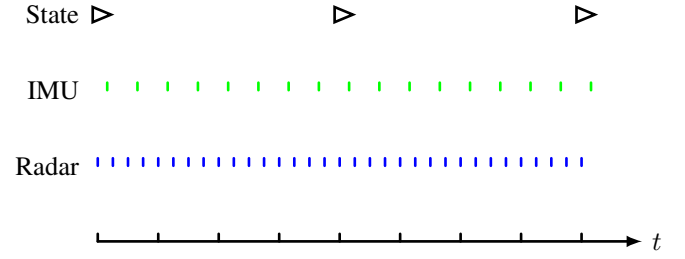

Fig. 3. This figure illustrates our asynchronous sensor timing where states are estimated for each scan obtained by the radar. Our radar outputs measurements at 1600Hz while our lidar outputs unique timestamps at roughly 40kHz and our IMU outputs measurements at 200Hz.

be included to penalize acceleration between pairs of poses. Examples of linear interpolation approaches include [51]–[53]. CT-ICP [54] is another 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.

Parametric approaches seek to represent the trajectory using a finite set of temporal basis functions. Previous examples of parametric approaches include [55], [56]. Recent examples of parametric approaches applied to lidar odometry and lidar-inertial odometry include [57]–[61] 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 [62]. Instead of the usual cubic time complexity for Gaussian process regression, this approach enables linear time complexity. 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 [6]. Recently, Le Gentil and Vidal-Calleja employed Gaussian processes in order to model the linear acceleration and angular velocities in continuous-time given a set of IMU measurements [9], [63]. 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
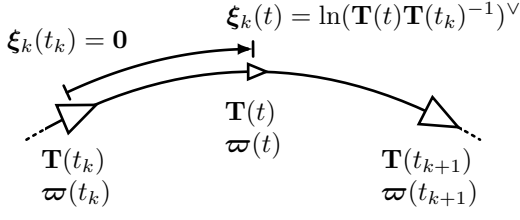
Fig. 4. This figure illustrates the local variable $\boldsymbol{\xi}_k(t)$ which is defined in the tangent space of the pose at time t_k. The larger triangles denote the estimated states while the smaller triangle denotes the interpolated state at time t.

be learned from a training set using maximum likelihood, enabling a data-driven approach to fine-tune the GP to each application. Determining the spacing of control points is an important engineering challenge in the use of B-splines which can be avoided by using Gaussian processes instead. For a comparison of splines and Gaussian processes, we refer to Johnson et al. [64]. For a survey on continuous-time state estimation, we refer to Talbot et al. [65].

III. CONTINUOUS-TIME TRAJECTORY ESTIMATION

In this section, we review relevant prior work [6], [62] on continuous-time trajectory estimation using Gaussian processes. We begin with the following nonlinear time-varying stochastic differential equation,

$$\dot{\mathbf{T}}(t) = \boldsymbol{\varpi}(t)^\wedge \mathbf{T}(t) \tag{1a}$$

$$\dot{\boldsymbol{\varpi}}(t) = \mathbf{w}'(t), \quad \mathbf{w}'(t) \sim \mathcal{GP}(\mathbf{0}, \boldsymbol{Q}'\delta(t-t')) \tag{1b}$$

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 [6]. 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

$$\mathbf{T}(t) = \exp(\boldsymbol{\xi}_k(t)^\wedge)\mathbf{T}(t_k). \tag{2}$$

The local kinematic equations are then defined as

$$\ddot{\boldsymbol{\xi}}_k(t) = \mathbf{w}_k(t), \quad \mathbf{w}_k(t) \sim \mathcal{GP}(\mathbf{0}, \boldsymbol{Q}\delta(t-t')). \tag{3}$$

This approximation of (1) 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

$$\boldsymbol{\gamma}_k(t) = \begin{bmatrix} \boldsymbol{\xi}_k(t) \\ \dot{\boldsymbol{\xi}}_k(t) \end{bmatrix}. \tag{4}$$

The local LTI SDE defined by (3), (4) is then stochastically integrated to arrive at the following local GP:

$$\boldsymbol{\gamma}_k(t) \sim \mathcal{GP}(\boldsymbol{\Phi}(t,t_k)\check{\boldsymbol{\gamma}}_k(t_k)), \tag{5}$$
$$\boldsymbol{\Phi}(t,t_k)\check{\mathbf{P}}(t_k)\boldsymbol{\Phi}(t,t_k)^T + \mathbf{Q}_k),$$

where

$$\boldsymbol{\Phi}(t,t_k) = \begin{bmatrix} \mathbf{1} & (t-t_k)\mathbf{1} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} \tag{6}$$

is the transition function,

$$\mathbf{Q}_k = \begin{bmatrix} \frac{1}{3}(t-t_k)^3 \boldsymbol{Q} & \frac{1}{2}(t-t_k)^2 \boldsymbol{Q} \\ \frac{1}{2}(t-t_k)^2 \boldsymbol{Q} & (t-t_k)\boldsymbol{Q} \end{bmatrix} \tag{7}$$

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

$$\check{\mathbf{P}} = \text{cov}(\delta\mathbf{x}) = \mathbf{A}\mathbf{Q}\mathbf{A}^T, \tag{8}$$

where

$$\mathbf{A}^{-1} = \begin{bmatrix} \mathbf{1} & & & \\ -\boldsymbol{\Phi}(t_1,t_0) & \ddots & & \\ & \ddots & \mathbf{1} & \\ & & -\boldsymbol{\Phi}(t_K,t_{K-1}) & \mathbf{1} \end{bmatrix} \tag{9}$$

is the inverse of the lifted transition matrix and $\mathbf{Q} = \text{diag}(\check{\mathbf{P}}_0, \mathbf{Q}_1, \cdots, \mathbf{Q}_K)$. 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.

In order to convert our continuous-time formulation into a factor graph, we construct a sequence of motion prior factors between pairs of estimation times,

$$J_{v,k} = \frac{1}{2}\mathbf{e}_{v,k}^T \mathbf{Q}_k^{-1} \mathbf{e}_{v,k}, \tag{10a}$$
$$\mathbf{e}_{v,k} = \boldsymbol{\gamma}_k(t_{k+1}) - \check{\boldsymbol{\gamma}}_k(t_{k+1})$$
$$- \boldsymbol{\Phi}(t_{k+1},t_k)(\boldsymbol{\gamma}_k(t_k) - \check{\boldsymbol{\gamma}}_k(t_k)), \tag{10b}$$

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 (10b) simplifies to

$$\mathbf{e}_{v,k} = \boldsymbol{\gamma}_k(t_{k+1}) - \boldsymbol{\Phi}(t_{k+1},t_k)\boldsymbol{\gamma}_k(t_k). \tag{11}$$

Now, to translate this prior factor, which is defined in terms of the local variables, into the global variables, we first rearrange (2) as

$$\boldsymbol{\xi}_k(t) = \ln(\mathbf{T}(t)\mathbf{T}(t_k)^{-1})^\vee. \tag{12}$$

We then convert from body-centric velocity using:

$$\dot{\boldsymbol{\xi}}_k(t) = \boldsymbol{\mathcal{J}}(\boldsymbol{\xi}_k(t))^{-1}\boldsymbol{\varpi}(t), \tag{13}$$

where $\boldsymbol{\mathcal{J}}$ is the left-hand Jacobian of $SE(3)$. From (12), (13), we can then define the local Markovian variable in terms of the global variables with

$$\boldsymbol{\gamma}(t) = \begin{bmatrix} \ln(\mathbf{T}(t)\mathbf{T}(t_k)^{-1})^\vee \\ \boldsymbol{\mathcal{J}}\left(\ln(\mathbf{T}(t)\mathbf{T}(t_k)^{-1})^\vee\right)^{-1}\boldsymbol{\varpi}(t) \end{bmatrix}. \tag{14}$$
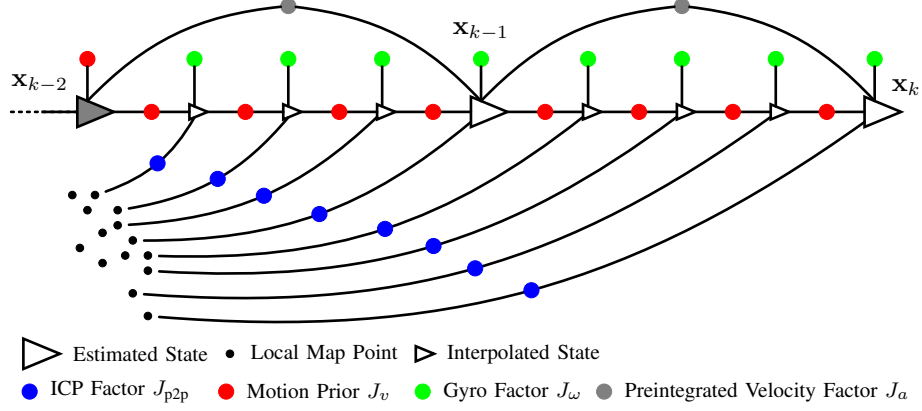
Fig. 5. This figure depicts a factor graph of our sliding window lidar-inertial odometry using a continuous-time motion prior. The larger triangles represent the estimation times that form our sliding window. The state $\mathbf{x}(t) = \{\mathbf{T}(t), \boldsymbol{\varpi}(t), \mathbf{b}(t)\}$ includes 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 the optimization process. Instead, we construct continuous-time measurement factors using the posterior Gaussian process interpolation formula in Equation 16. 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 instead. We include a unary prior on $\mathbf{x}_{k-2}$ to denote the prior information from the sliding window filter.

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

$$
\mathbf{e}_{v,k} = \begin{bmatrix} \ln(\mathbf{T}_{k+1}\mathbf{T}_k^{-1})^\vee - (t_{k+1} - t_k)\boldsymbol{\varpi}_k \\ \mathcal{J}\left(\ln(\mathbf{T}_{k+1}\mathbf{T}_k^{-1})^\vee\right)^{-1}\boldsymbol{\varpi}_{k+1} - \boldsymbol{\varpi}_k \end{bmatrix}, \quad (15)
$$

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

$$
\begin{aligned}
\hat{\mathbf{T}}(\tau) =& \exp\left((\boldsymbol{\Lambda}_1(\tau)\hat{\boldsymbol{\gamma}}_k(t_k) + \boldsymbol{\Psi}_1(\tau)\hat{\boldsymbol{\gamma}}_k(t_{k+1}))^\wedge\right)\hat{\mathbf{T}}_k, \\
\hat{\boldsymbol{\varpi}}(\tau) =& \mathcal{J}(\ln(\hat{\mathbf{T}}(\tau)\hat{\mathbf{T}}_k^{-1})^\vee) \\
& \times (\boldsymbol{\Lambda}_2(\tau)\hat{\boldsymbol{\gamma}}_k(t_k) + \boldsymbol{\Psi}_2(\tau)\hat{\boldsymbol{\gamma}}_k(t_{k+1}))
\end{aligned} \quad (16)
$$

are the interpolation equations involving only the two states bracketing the desired interpolation time: $t_k < \tau < t_{k+1}$. The interpolation matrices that result from the standard GP interpolation formula are

$$
\boldsymbol{\Psi}(\tau) = \mathbf{Q}_\tau \boldsymbol{\Phi}(t_{k+1}, \tau)^T \mathbf{Q}_{k+1}^{-1}, \quad (17a)
$$

$$
\boldsymbol{\Lambda}(\tau) = \boldsymbol{\Phi}(\tau, t_k) - \boldsymbol{\Psi}(\tau)\boldsymbol{\Phi}(t_{k+1}, t_k). \quad (17b)
$$

where $\boldsymbol{\Psi}(\tau) = [\boldsymbol{\Psi}_1(\tau)^T \ \boldsymbol{\Psi}_2(\tau)^T]^T$ and $\boldsymbol{\Lambda}(\tau) = [\boldsymbol{\Lambda}_1(\tau)^T \ \boldsymbol{\Lambda}_2(\tau)^T]^T$. When performing continuous-time trajectory estimation, we use the posterior interpolation formulas to build measurement factors at times between our desired estimation times.

IV. RADAR-INERTIAL AND LIDAR-INERTIAL ODOMETRY

In this section, we describe our lidar-inertial odometry, which is implemented as sliding-window batch trajectory estimation. The factor graph corresponding to our approach is depicted in Figure 5. 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 $\mathbf{b}(t)$. In our notation, $\boldsymbol{\varpi}_v^{vi}$ is a

6×1 vector containing the body-centric linear velocity $\boldsymbol{\nu}_v^{vi}$ and angular velocity $\boldsymbol{\omega}_v^{vi}$. We use a white-noise-on-acceleration prior, as defined in (1). Our IMU measurement model is

$$
\begin{bmatrix} \tilde{\mathbf{a}} \\ \tilde{\boldsymbol{\omega}} \end{bmatrix} = \begin{bmatrix} \mathbf{a}_v^{vi} - \mathbf{C}_{vi}\mathbf{g}_i \\ \boldsymbol{\omega}_v^{vi} \end{bmatrix} + \begin{bmatrix} \mathbf{b}_a \\ \mathbf{b}_\omega \end{bmatrix} + \begin{bmatrix} \mathbf{w}_a \\ \mathbf{w}_\omega \end{bmatrix} \quad (18)
$$

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 being a part of the state, the associated gyroscope error function is straightforward:

$$
J_{\omega,\ell} = \frac{1}{2}\mathbf{e}_{\omega,\ell}^T \mathbf{R}_\omega^{-1}\mathbf{e}_{\omega,\ell}, \quad (19a)
$$

$$
\mathbf{e}_{\omega,\ell} = \tilde{\boldsymbol{\omega}}_\ell - \boldsymbol{\omega}(\tau_\ell) - \mathbf{b}_\omega(\tau_\ell). \quad (19b)
$$

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

$$
\Delta\boldsymbol{\nu}(t_{k+1}, \tau_1) = \sum_{n=1}^N \left(\tilde{\mathbf{a}}_n + \mathbf{C}_{vi}(\tau_n)\mathbf{g}_i - \mathbf{b}_a(\tau_n)\right)\Delta t_n, \quad (20)
$$

where the associated factor is given by

$$
J_{a,k} = \frac{1}{2}\mathbf{e}_{a,k}^T \mathbf{R}_a(t_{k+1}, \tau_1)^{-1}\mathbf{e}_{a,k}, \quad (21a)
$$

$$
\mathbf{e}_{a,k} = \boldsymbol{\nu}(t_{k+1}) - \boldsymbol{\nu}(\tau_1) - \Delta\boldsymbol{\nu}(t_{k+1}, \tau_1). \quad (21b)
$$

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 (19b) (21b), we use a continuous-time interpolation of the state. In order to interpolate for the measurement times, we use the posterior GP interpolation formula (16). 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

$$
\mathbf{e}_{\omega,\ell} \approx \bar{\mathbf{e}}_{\omega,\ell} + \frac{\partial\mathbf{e}_{\omega,\ell}}{\partial\delta\boldsymbol{\omega}(\tau_\ell)}\left(\frac{\partial\delta\boldsymbol{\omega}(\tau_\ell)}{\partial\delta\mathbf{x}_k}\delta\mathbf{x}_k + \frac{\partial\delta\boldsymbol{\omega}(\tau_\ell)}{\partial\delta\mathbf{x}_{k+1}}\delta\mathbf{x}_{k+1}\right), \quad (22)
$$

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 estimation times and the number of measurement times. This is different from the approach of Le Gentil and Vidal-Calleja [9] 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

$$J_{\text{p2p},j} = \mathbf{e}_{\text{p2p},j}^T \mathbf{R}_{\text{p2p},j}^{-1} \mathbf{e}_{\text{p2p},j}, \tag{23a}$$

$$\mathbf{e}_{\text{p2p},j} = \alpha_j \mathbf{n}_j^T \mathbf{D}(\mathbf{p}_j - \mathbf{T}_{vi}(\tau_j)^{-1}\mathbf{T}_{vs}\mathbf{q}_j), \tag{23b}$$

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 $\alpha = (\sigma_2 - \sigma_3)/\sigma_1$ [18], [54] is a heuristic weight to favour planar neighborhoods. Query points are matched to a sliding local voxel map centered 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 to it. 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 penalize the state for deviating from a constant velocity. These motion prior factors are defined in (10a) (10b). We also include motion prior factors for the IMU biases,

$$J_{v,b,k} = \frac{1}{2}\mathbf{e}_{v,b,k}^T \mathbf{Q}_{b,k}^{-1}\mathbf{e}_{v,b,k}, \tag{24a}$$

$$\mathbf{e}_{v,b,k} = \mathbf{b}(t_{k+1}) - \mathbf{b}(t_k), \tag{24b}$$

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. The objective function that we seek to minimize is then

$$J = \sum_k J_{v,k} + \sum_j J_{\text{p2p},j} + \sum_\ell J_{\omega,\ell} + \sum_k J_{a,k}. \tag{25}$$

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

---

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 $t \in [t_{k-2}, t_k]$

1: $\mathbf{T}(t_k) \leftarrow \exp(\Delta t_k \boldsymbol{\varpi}(t_{k-1})^\wedge)\mathbf{T}(t_{k-1})$
2: $\boldsymbol{\varpi}(t_k) \leftarrow \boldsymbol{\varpi}(t_{k-1})$, $\mathbf{b}(t_k) \leftarrow \mathbf{b}(t_{k-1})$
3: $\hat{\mathbf{x}}(t), \mathbf{A}, \mathbf{c} \leftarrow \text{SlideWindow}(\hat{\mathbf{x}}(t), \mathbf{A}, \mathbf{c}, \mathbf{x}_k)$
4: $\{\mathbf{q}_j, \tau_j\} \leftarrow \text{Downsample}(\{\mathbf{q}_j, \tau_j\})$
5: $x \leftarrow 0$, $||\Delta\mathbf{x}|| \leftarrow \infty$
6: **while** $||\Delta\mathbf{x}|| > T_{\text{outer}} \wedge x < N_{\text{outer}}$ **do**
7: $\quad \{\bar{\mathbf{q}}_j\} \leftarrow \text{Undistort}(\{\mathbf{q}_j, \tau_j\}, \hat{\mathbf{x}}(t))$
8: $\quad \{\mathbf{p}_j, \mathbf{n}_j\} \leftarrow \text{Matching}(\{\mathbf{p}_i\}, \{\bar{\mathbf{q}}_j\})$
9: $\quad J \leftarrow J_v(\mathbf{x}_{k-1}, \mathbf{x}_k) + J_\omega(\mathbf{x}(t), \{\tilde{\boldsymbol{\omega}}_\ell\})$
$\quad\quad + J_a(\mathbf{x}(t), \{\tilde{\mathbf{a}}_\ell\}) + J_{\text{p2p}}(\mathbf{x}(t), \{\mathbf{p}_j, \mathbf{n}_j, \mathbf{q}_j, \tau_j\})$
10: $\quad y \leftarrow 0, ||\delta\mathbf{x}|| \leftarrow \infty, \Delta J \leftarrow \infty, \mathbf{x}_{\text{prev}} \leftarrow \hat{\mathbf{x}}(t)$
11: $\quad$ **while** $||\delta\mathbf{x}|| > T_{\text{inner}} \wedge \Delta J > \delta J \wedge y < N_{\text{inner}}$ **do**
12: $\quad\quad \mathbf{A}, \mathbf{c} \leftarrow \text{buildAndUpdateGN}(\mathbf{A}, \mathbf{c}, J, \hat{\mathbf{x}}(t))$
13: $\quad\quad \delta\mathbf{x} \leftarrow \text{CholeskySolve}(\mathbf{A}, \mathbf{c})$
14: $\quad\quad \hat{\mathbf{x}}(t) \leftarrow \text{UpdateState}(\hat{\mathbf{x}}(t), \delta\mathbf{x})$
15: $\quad\quad J, \Delta J \leftarrow \text{UpdateCost}(\hat{\mathbf{x}}(t), J)$
16: $\quad\quad y \leftarrow y + 1$
17: $\quad$ **end while**
18: $\quad ||\Delta\mathbf{x}|| \leftarrow \text{Dist}(\mathbf{x}_{\text{prev}}, \hat{\mathbf{x}}(t))$
19: $\quad x \leftarrow x + 1$
20: **end while**
21: $\{\mathbf{p}_i\} \leftarrow \text{UpdateMap}(\{\mathbf{p}_i\}, \{\mathbf{q}_j, \tau_j\}, \hat{\mathbf{x}}(t))$

---

practice, we typically limit the maximum number of inner-loop Gauss-Newton iterations to 5, and the number of outer-loop iterations to 10 in order to enable real-time operation.

A. Implementation Details

Algorithm 1 provides pseudocode for STEAM-LIO at a high level and Figure 6 depicts the software architecture for our approach. For a new lidar frame, we first initialize the new state using constant velocity. 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 in order 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. This voxelization strategy is inspired by CT-ICP [18].

In order 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
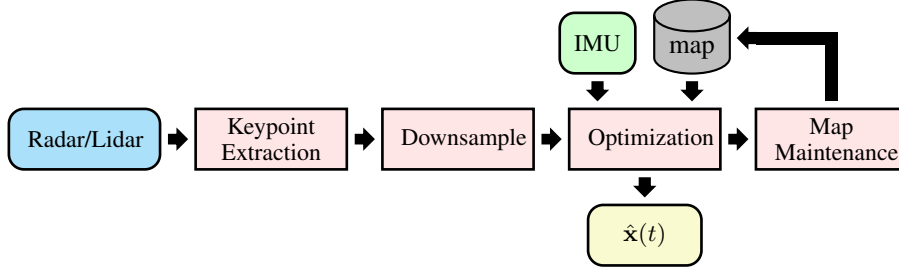
Fig. 6. This figure depicts the simple architecture diagram for STEAM-LIO. In the radar-based pipelines, keypoints are first extracted using a constant false alarm rate (CFAR) detector. In the lidar-based pipelines, we randomly shuffle the order of the points and then downsample using a coarse voxel 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.

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 by reducing the timestamp frequency in this way, we can retain most of the benefits of continuous-time state estimation while still operating efficiently.

B. 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,

$$J = \sum_n \mathbf{e}_{a,n}^T \mathbf{R}_a^{-1} \mathbf{e}_{a,n} + \ln(\mathbf{C}_{ig})^{\vee T} \check{\mathbf{P}}_g^{-1} \ln(\mathbf{C}_{ig})^{\vee} \quad (26a)$$
$$+ \mathbf{b}_a^T \check{\mathbf{P}}_b^{-1} \mathbf{b}_a, \quad$$
$$\mathbf{e}_{a,n} = \tilde{\mathbf{a}}_n - \mathbf{C}_{ig}\mathbf{g} - \mathbf{b}_a, \quad (26b)$$

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,

$$J_{v,g,k} = \frac{1}{2}\mathbf{e}_{v,g,k}^T \mathbf{Q}_{g,k}^{-1} \mathbf{e}_{v,g,k}, \quad (27a)$$
$$\mathbf{e}_{v,g,k} = \ln(\mathbf{C}_{ig}(t_k)\mathbf{C}_{ig}(t_{k+1})^{-1})^{\vee}, \quad (27b)$$

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. [66] has shown that it can be beneficial to mapping accuracy in challenging scenarios.

C. 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 at

the middle of the newest lidar frame so that the latency is equivalent to competing approaches. In Figure 5, 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 $\mathbf{x}_{k-1}$. As such, 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,

$$\begin{bmatrix} \mathbf{A}_{k-2,k-2} & \mathbf{A}_{k-1,k-2}^T & \\ \mathbf{A}_{k-1,k-2} & \mathbf{A}_{k-1,k-1} & \mathbf{A}_{k,k-1}^T \\ & \mathbf{A}_{k,k-1} & \mathbf{A}_{k,k} \end{bmatrix} \begin{bmatrix} \delta\mathbf{x}_{k-2}^{\star} \\ \delta\mathbf{x}_{k-1}^{\star} \\ \delta\mathbf{x}_k^{\star} \end{bmatrix} = \begin{bmatrix} \mathbf{c}_{k-2} \\ \mathbf{c}_{k-1} \\ \mathbf{c}_k \end{bmatrix}$$
$$(28)$$

becomes

$$\begin{bmatrix} \mathbf{A}_{k-1,k-1} - \mathbf{A}_{k-1,k-2}\mathbf{A}_{k-2,k-2}^{-1}\mathbf{A}_{k-1,k-2}^T \cdots \\ \mathbf{A}_{k,k-1} \end{bmatrix} \quad (29)$$
$$\begin{bmatrix} \mathbf{A}_{k,k-1}^T \\ \mathbf{A}_{k,k} \end{bmatrix} \begin{bmatrix} \delta\mathbf{x}_{k-1}^{\star} \\ \delta\mathbf{x}_k^{\star} \end{bmatrix} = \begin{bmatrix} \mathbf{c}_{k-1} - \mathbf{A}_{k-1,k-2}\mathbf{A}_{k-2,k-2}^{-1}\mathbf{c}_{k-2} \\ \mathbf{c}_k \end{bmatrix}.$$

D. 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 that we use in this work is the Navtech CIR304-H, a mechanical spinning radar that provides a $360°$ horizontal field of view. This sensor is 2D only, as such we do not estimate the orientation of the gravity vector and we remove the gravity term from the preintegrated velocity factor in (20). 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 [67] where we retain the maximum of the left and right subwindows, a variant known as GO-CFAR [68]. 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 has a tendency to fail due to the significant amount of noise present in the radar pointclouds. The Navtech radar
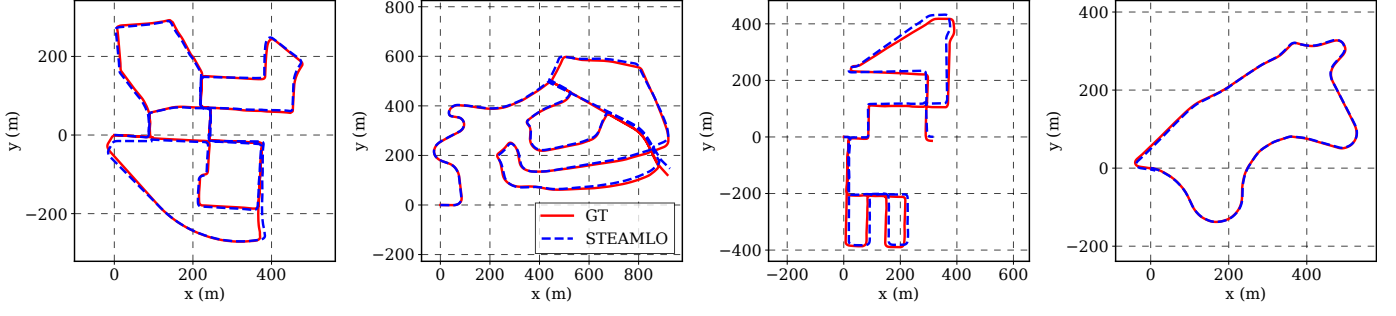
Fig. 7. Odometry results on the KITTI-raw dataset. Sequences shown from left to right are 00, 02, 07, 08. The ground-truth trajectory is shown as a solid red line and the STEAM-LO trajectory estimate is shown as a blue dashed line. Note that the estimated trajectory was computed in an online fashion with a sliding window of 200ms and does not make use of any loop closures.

scans each azimuth only once and as such range measurements are corrupted by a Doppler distortion dependent on the robot's egomotion [69]. The Doppler-compensated point-to-point ICP error for radar odometry is then

$$\mathbf{e}_{\text{p2p},j} = \rho \left(\mathbf{D} \left(\mathbf{p}_j - \mathbf{T}(\tau_j)^{-1} \mathbf{T}_{vs}(\mathbf{q}_j + \Delta\mathbf{q}_j) \right) \right), \quad (30a)$$

$$\text{where } \Delta\mathbf{q}_j = \beta \mathbf{D}^T \mathbf{a}_j \mathbf{a}_j^T \mathbf{D} \mathbf{q}_j^{\odot} \boldsymbol{\mathcal{T}}_{sv} \boldsymbol{\varpi}(\tau_j), \quad (30b)$$

and where $\rho(\cdot)$ is a Cauchy robust cost function, $\mathbf{D}$ is 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 [69], $\mathbf{a}_j$ 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 [70], and $\boldsymbol{\mathcal{T}}_{sv} = \text{Ad}(\mathbf{T}_{sv})$.

V. EXPERIMENTAL RESULTS

We provide experimental results on three datasets, KITTI-raw [13], the Newer College Dataset (NCD) [15], and the Boreas dataset [14]. KITTI-raw was chosen as it is a popular dataset for benchmarking lidar odometry. The raw version of the dataset contains the motion-distorted pointclouds whereas the original version of the dataset is motion-compensated using GPS poses. Since the purpose of this work is 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 in order to demonstrate continuous-time radar inertial odometry and to provide a detailed comparison with lidar. In all experiments, we provide average runtime estimates 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 tune the IMU measurement covariances and bias motion priors for Boreas and the Newer College Dataset separately. We downsample lidar timestamps to 400Hz on the Boreas dataset in order to achieve real-time performance. However, we do not downsample timestamps on KITTI-raw or the Newer College Dataset as we were already able to run in real-time. On the Boreas dataset, we clear voxels that have not been observed for one second. On the Newer College Dataset, we incrementally build a map to enable implicit loop closures by revisiting previously mapped areas.

A. KITTI-Raw Results

The KITTI dataset was collected in Karlsruhe, Germany using an autonomous driving platform equipped 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 I 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 [12]. 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 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 7 provides some qualitative examples of the trajectories estimated by our lidar odometry. This dataset does not provide raw IMU measurements, as such 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.

TABLE I

KITTI-RAW RESULTS (22KM / 0.6H): KITTI RTE (%). THE AVERAGE IS COMPUTED OVER ALL SEGMENTS OF ALL SEQUENCES AS IN [18]. 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 [18]	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 [18]
KISS-ICP [5]	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 [5]
STEAM-ICP [12]	**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

TABLE II

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 [71], ‡ USES CAMERA.

Newer College Dataset	01-Short	02-Long	05-Quad w/ Dynamics	06-Dynamic Spinning	07-Parkland Mound	ΔT
CT-ICP⋆ [18]	0.36					430ms [18]
KISS-ICP† [5]	0.6675	1.5311	0.1040	Failed	0.2027	167ms†
FAST-LIO2† [10]	0.3775	0.3324	0.0879	0.0771	0.1483	43ms†
DLIO [11]	0.3606	**0.3268**	**0.0837**	**0.0612**	**0.1196**	36ms [11]
SLICT⋆ [52]	0.3843	0.3496	0.1155	0.0844	0.1290	
CLIO⋆‡ [60]	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 III

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 [1]	STEAM-LO	STEAM-LIO	STEAM-LO(SE2)	VTR3-Radar [1]	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

B. 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 so that we can 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 II shows our quantitative results for this dataset. Again, we include CT-ICP and KISS-ICP since they are well-known lidar odometry approaches. DLIO [11] and FAST-LIO2 [10] are also included as these approaches currently represent the state of the art for lidar-inertial odometry. Finally, we include SLICT [52] and CLIO [60] as these are continuous-time approaches that use linear interpolation and B-splines, respectively. It is challenging to make a direct comparison 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. We note that, 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
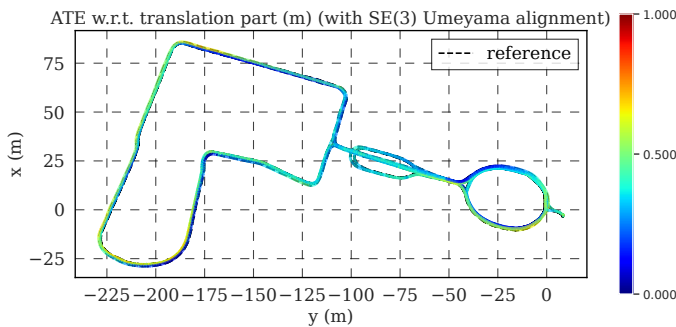
Fig. 8. This figure depicts the trajectory estimated by STEAM-LIO during the long sequence of the Newer College Dataset. The trajectory is colored according to the absolute trajectory error when compared against the ground truth. The estimated trajectory was aligned with the ground truth using the Umeyama algorithm [73].

dataset [72] 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 II contains originally published results, except for the results provided for FAST-LIO2 and KISS-ICP where the results were obtained from [71].

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) with plenty of geometric features for point-to-plane ICP. In this dataset, the addition of an IMU seems to have the most noticeable improvement in sequences where there are brief periods lacking 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 IV that our continuous-time approach, STEAM-LO, significantly outperforms KISS-ICP, which relies on a constant-velocity assumption. Our lidar-inertial approach, STEAM-LIO achieves the best performance on sequence 01-Short by a significant margin while remaining competitive with the other lidar-inertial approaches on the other sequences. In fact, if we compute an overall absolute trajectory error for the entire Newer College Dataset by concatenating the squared errors across all timestamps of all sequences, our approach (0.2946m) actually 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 8, we provide a qualitative example of the trajectory estimated by our approach. In this case, the trajectory is colored by the absolute trajectory error (ATE). The estimated trajectory is first aligned with the ground truth using

(a) Panoramic image of the courtyard at New College, Oxford
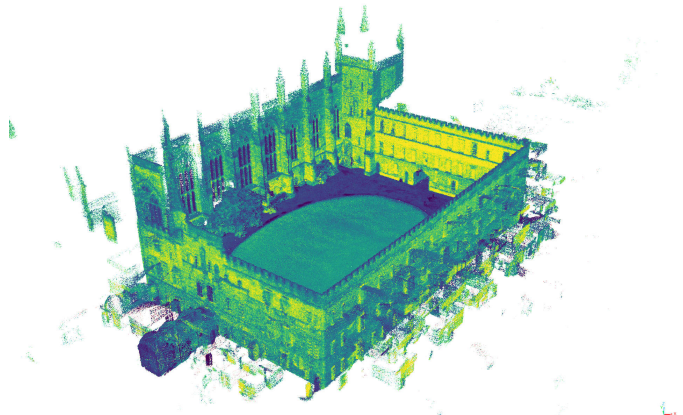
(b) Lidar map of the courtyard colored by reflectivity

Fig. 9. 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 [15]. In order to produce this figure, we adjusted configuration parameters to produce a denser map.

the Umeyama algorithm [73] before computing the ATE as described in [74]. Even though we do not make use of 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.

We provide another qualitative example of our lidar-inertial odometry in Figure 9 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 colored 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 are able to produce a crisp high-quality map.

C. Ablation Study

In addition to the continuous-time odometry methods presented in this work, we also present baseline results using a constant-velocity assumption. We approximate the body-centric velocity using the following formula, which includes the poses at the two previous timesteps,

$$\breve{\varpi}_k \approx \frac{1}{\Delta t_{k-1}} \ln \left(\mathbf{T}_{k-1} \mathbf{T}_{k-2}^{-1} \right)^{\vee}. \tag{31}$$
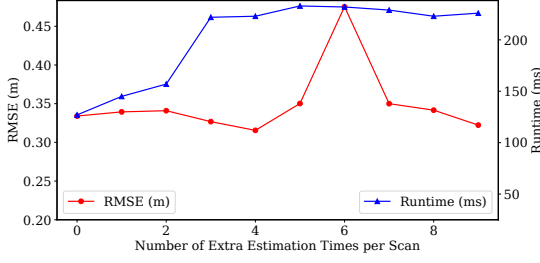
Fig. 10. Root mean squared error vs. the number of estimation times per lidar scan for STEAM-LO for sequence 01-Short of the Newer College Dataset.
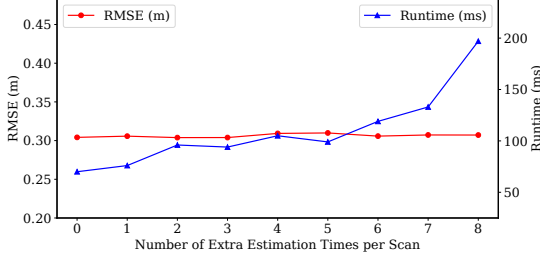

Fig. 12. Root mean squared error (RMSE) vs. the lidar timestamp frequency for STEAM-LO for sequence 01-Short of the Newer College Dataset.

Fig. 11. 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.

Fig. 13. Root mean squared error (RMSE) vs. the lidar timestamp frequency for STEAM-LIO for sequence 01-Short of the Newer College Dataset.

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

$$\tilde{\mathbf{q}}_j := \exp((t_k - \tau_j)\breve{\boldsymbol{\varpi}}_k^\wedge)\mathbf{T}_{vs}\mathbf{q}_j. \tag{32}$$

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

$$\mathbf{e}_{\mathrm{p2p},j} = \alpha_j \mathbf{n}_j^T \mathbf{D}(\mathbf{p}_j - \mathbf{T}_k^{-1}\tilde{\mathbf{q}}_j), \tag{33}$$

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 I and Table II. 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. On the Newer College Dataset, the gap between the constant-velocity approach and our continuous-time approach is much more apparent. 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 where the default, zero, refers to having an estimation time at the beginning and at the end of each scan. As shown in Figure 10, STEAM-LO's performance improves slightly when increasing the number of extra estimation times to four. However, the improvement does not continue when increasing the number of estimation times 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 can be seen in Figure 11. Similar experiments varying the number of estimation times or basis functions were previously presented in [55], [64].

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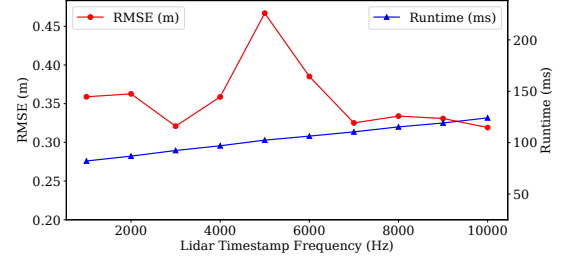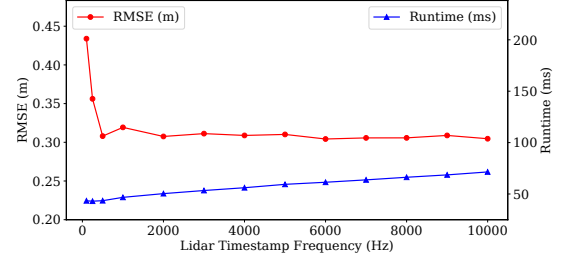
but the associated timestamps are rounded so that the effective timestamp frequency is reduced. This has the effect of requiring less continuous-time interpolations of the state and also less Jacobians need to be computed during optimization. Overall, the effect is that the continuous-time deskewing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

In Figure 12, 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 13.

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 $\mathrm{diag}(\boldsymbol{Q}) = \{50, 50, 50, 5, 5, 5\}$.

In Table IV, 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, when increasing the parameters of $\boldsymbol{Q}$, STEAM-LO fails.

TABLE IV
ATE RESULTS (M) ON SEQUENCE 01-SHORT OF THE NEWER COLLEGE DATASET WHEN VARYING $\mathrm{DIAG}(\boldsymbol{Q}) = \{50, 50, 50, 5, 5, 5\}$.

$\boldsymbol{Q}$	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

Fig. 14. This figure depicts our data collection platform Boreas which includes a Velodyne Alpha-Prime 128-beam lidar, a Navtech CIR304-H radar, an Applanix GNSS/IMU, and a FLIR Blackfly S camera.
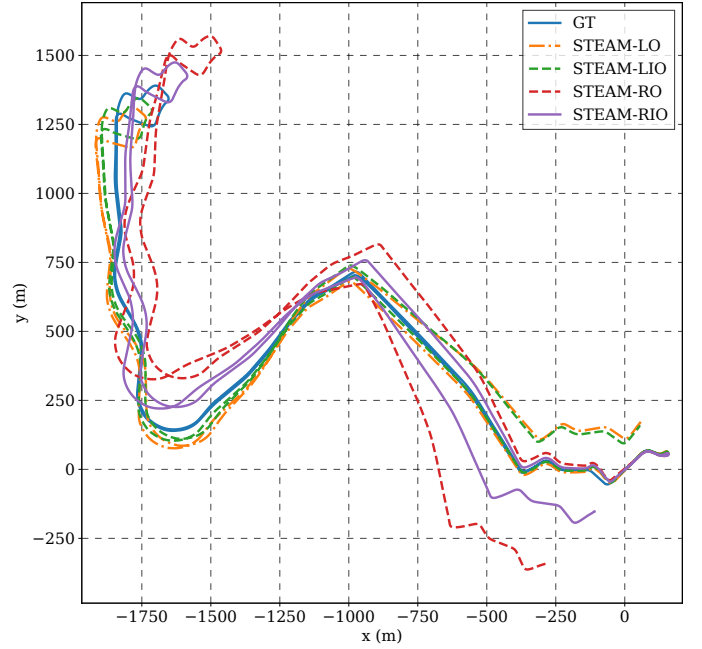

Fig. 15. In this figure we compare the performance of odometry approaches presented in this work using the Boreas dataset. The depicted sequence is 2021-01-26-10-59 which was collected during a snowstorm.

D. Boreas Results

The Boreas dataset was collected at the University of Toronto by driving a repeated route over the course of one year. The dataset features varying seasonal and weather conditions. The sensor suite, depicted in Figure 14 includes a Navtech CIR304-H radar, a 128-beam Velodyne lidar, and an Applanix GNSS/INS. Ground-truth poses were obtained by post-processing all GPS, IMU, and wheel encoder measurements as well as using a subscription for GPS corrections. In total, the test set features 102km or 4.3h of driving data. We extract 200Hz raw IMU measurements from the Applanix logs and take care to ensure that the IMU measurements are not bias-corrected in any way by the GPS. Table III 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) [69].

It is somewhat surprising to see that our lidar-inertial odometry does not do much better than our lidar odometry here. Our hypothesis is 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 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 the performance of lidar and radar odometry. 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 [35]. One important caveat here is that the underlying ground truth is in $SE(3)$ whereas radar odometry is being 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 15 shows a qualitative example of trajectories estimated by our approach on the Boreas dataset. We observe that 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 16 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 17, we plot the odometry errors vs. time where we compare 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

(a) Lidar Odometry Drift vs. Path Length

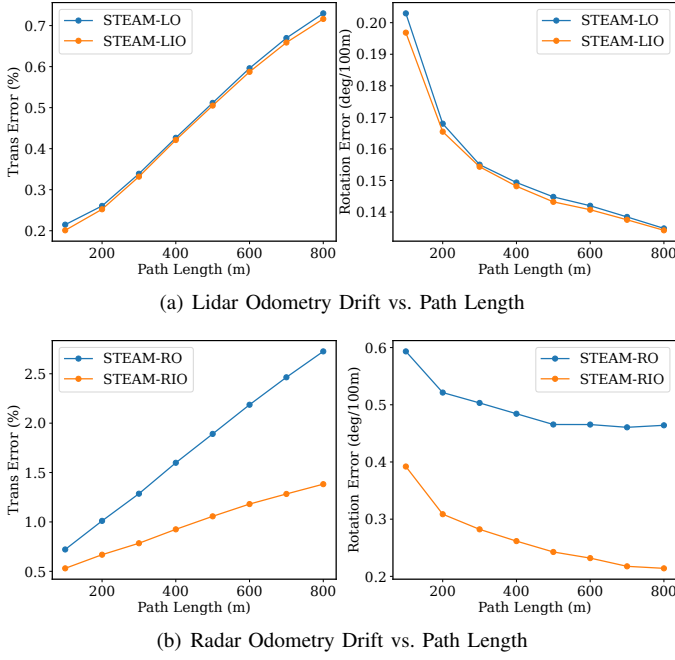(b) Radar Odometry Drift vs. Path Length

Fig. 16. 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.

estimates to obtain a relative odometry pose change,

$$\hat{\mathbf{T}}_{k,k-1} = \hat{\mathbf{T}}_k \hat{\mathbf{T}}_{k-1}^{-1}. \tag{34}$$

The error that we compute is

$$\boldsymbol{\xi}_{k,k-1} = \ln\left(\hat{\mathbf{T}}_{k,k-1}\mathbf{T}_{k,k-1}^{-1}\right)^{\vee}, \tag{35}$$

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 [70, §11.3.2] for details. The covariance of $\hat{\mathbf{T}}_{k,k-1}$, $\mathrm{cov}(\hat{\mathbf{T}}_{k,k-1}) = \boldsymbol{\Sigma}_{k,k-1}$, is obtained using [70]

$$\boldsymbol{\Sigma}_{k,k-1} \approx \hat{\mathbf{P}}_{k,k} + \boldsymbol{\mathcal{T}}_{k,k-1}\hat{\mathbf{P}}_{k-1,k-1}\boldsymbol{\mathcal{T}}_{k,k-1}^{T}, \tag{36}$$

where $\boldsymbol{\mathcal{T}}_{k,k-1} = \mathrm{Ad}(\hat{\mathbf{T}}_{k,k-1})$. 35 and 36 are then used to produce the plots in Figure 17. We compute the normalized estimation error squared (NEES) using

$$\mathrm{NEES} = \sum_{k=1}^{K} \frac{\boldsymbol{\xi}_{k,k-1}^{T}\boldsymbol{\Sigma}_{k,k-1}^{-1}\boldsymbol{\xi}_{k,k-1}}{\dim(\boldsymbol{\xi}_{k,k-1})K}. \tag{37}$$

For the lidar-inertial odometry shown in Figure 15 and Figure 17, 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 18, we compare the maps generated using our lidar-inertial odometry and our radar-inertial odometry. The lidar map is colored by height and is displayed using a top-down orthographic projection. Our lidar-inertial odometry has minimal drift and so the map that it generates aligns quite well with satellite imagery.
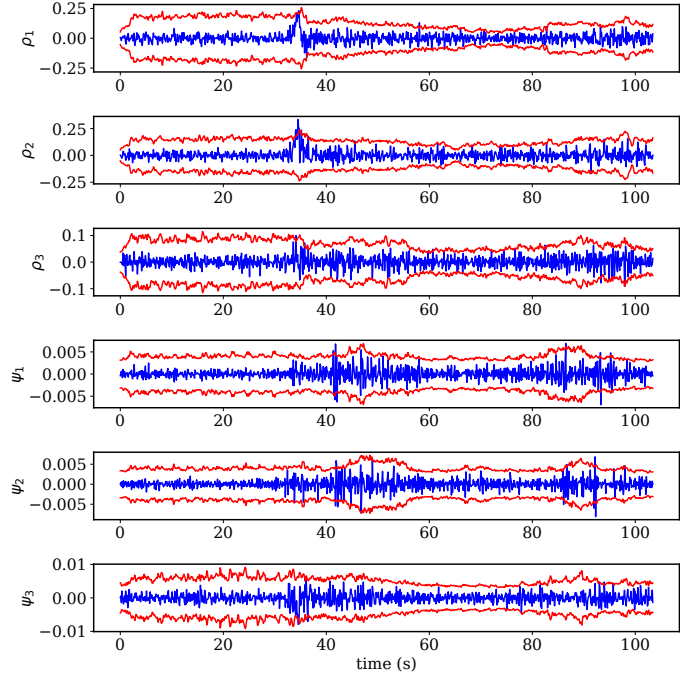

Fig. 17. This figure plots error vs. time for our lidar-inertial odometry when compared to ground truth on the first 100 seconds of sequence 2021-01-26-10-59. The red lines denote the estimated 3σ uncertainty bounds. Each row represents a dimension of the log map of the pose error where ρ is a translational dimension and ψ is a rotational dimension.

(a) Satellite image of UTIAS

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

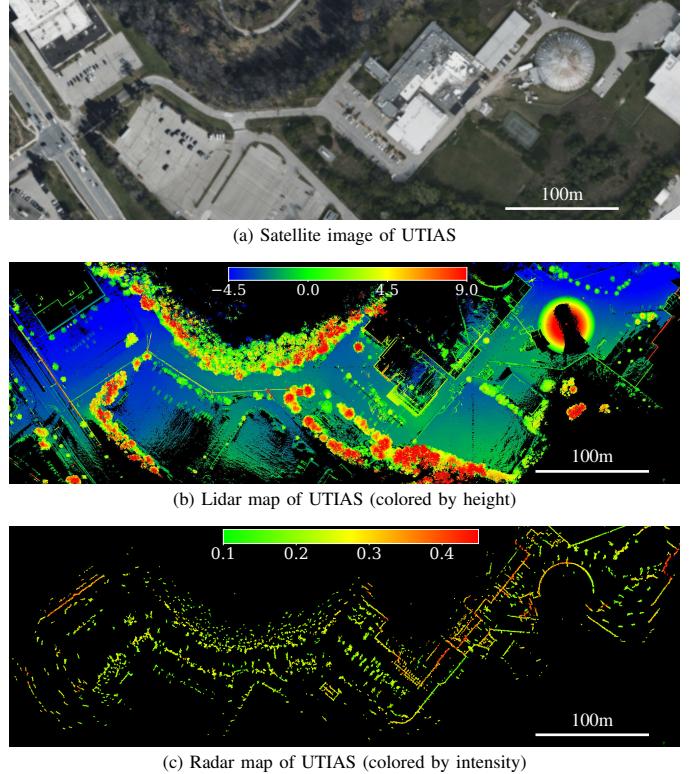
(c) Radar map of UTIAS (colored by intensity)

Fig. 18. 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.

It is clear that a high level of detail is being captured by the lidar map. In order 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 [1].

VI. Conclusions

In this work, we showed that our Gaussian process motion prior is often sufficient to compensate for motion-distorted lidar data when there is 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 the inclusion of gyroscope measurements whereas the addition of accelerometer measurements yielded only a minor additional 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.

Appendix

A. Interpolation Jacobians

In this work, 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

$$\mathbf{T}_k = \exp(\boldsymbol{\epsilon}_k^\wedge)\mathbf{T}_{\mathrm{op},k}, \tag{38a}$$

$$\boldsymbol{\varpi}_k = \boldsymbol{\varpi}_{\mathrm{op},k} + \boldsymbol{\eta}_k. \tag{38b}$$

In order to compute the interpolation Jacobians, we first need to linearize some expressions contained in (16). When

we evaluate the local Markovian variable at the endpoints of the local GP, we get the following results,

$$\hat{\boldsymbol{\gamma}}_k(t_k) = \begin{bmatrix} \mathbf{0} \\ \hat{\boldsymbol{\varpi}}_k \end{bmatrix}, \tag{39a}$$

$$\hat{\boldsymbol{\gamma}}_k(t_{k+1}) = \begin{bmatrix} \ln\left(\hat{\mathbf{T}}_{k+1}\hat{\mathbf{T}}_k^{-1}\right)^\vee \\ \mathcal{J}\left(\ln\left(\hat{\mathbf{T}}_{k+1}\hat{\mathbf{T}}_k^{-1}\right)^\vee\right)^{-1}\hat{\boldsymbol{\varpi}}_{k+1} \end{bmatrix}. \tag{39b}$$

Next, we linearize

$$\ln\left(\mathbf{T}_{k+1}\mathbf{T}_k^{-1}\right)^\vee \approx \ln\left(\mathbf{T}_{\mathrm{op},k+1}\mathbf{T}_{\mathrm{op},k}^{-1}\right)^\vee \tag{40}$$
$$+ \mathcal{J}_{\mathrm{op},k+1,k}^{-1}(\boldsymbol{\epsilon}_{k+1} - \mathcal{T}_{\mathrm{op},k+1,k}\boldsymbol{\epsilon}_k),$$

where we have assumed that $\boldsymbol{\epsilon}_{k+1} - \mathcal{T}_{\mathrm{op},k+1,k}\boldsymbol{\epsilon}_k$ is small and we have defined

$$\mathcal{T}_{\mathrm{op},k+1,k} = \mathcal{T}_{\mathrm{op},k+1}\mathcal{T}_{\mathrm{op},k}^{-1}, \tag{41a}$$

$$\mathcal{J}_{\mathrm{op},k+1,k} = \mathcal{J}\left(\ln(\mathcal{T}_{\mathrm{op},k+1}\mathcal{T}_{\mathrm{op},k}^{-1})^\vee\right), \tag{41b}$$

and $\mathcal{T}_{\mathrm{op},k} = \mathrm{Ad}(\mathbf{T}_{\mathrm{op},k})$. We also make the following linearization

$$\mathcal{J}\left(\ln\left(\hat{\mathbf{T}}_{k+1}\hat{\mathbf{T}}_k^{-1}\right)^\vee\right)^{-1} \approx \mathcal{J}_{\mathrm{op},k+1,k}^{-1} \tag{42}$$
$$- \frac{1}{2}\left(\mathcal{J}_{\mathrm{op},k+1,k}^{-1}(\boldsymbol{\epsilon}_{k+1} - \mathcal{T}_{\mathrm{op},k+1,k}\boldsymbol{\epsilon}_k)\right)^\curlywedge,$$

where we have again assumed that $\boldsymbol{\epsilon}_{k+1} - \mathcal{T}_{\mathrm{op},k+1,k}\boldsymbol{\epsilon}_k$ is small and we have approximated the inverse left Jacobian with $\mathcal{J}^{-1}(\mathbf{x}) \approx \mathbf{1} - \frac{1}{2}\mathbf{x}^\curlywedge$. The interpolated local variables between estimation times t_k, t_{k+1} are defined as

$$\boldsymbol{\xi}_k(\tau) = \boldsymbol{\Lambda}_1(\tau)\hat{\boldsymbol{\gamma}}_k(t_k) + \boldsymbol{\Psi}_1(\tau)\hat{\boldsymbol{\gamma}}_k(t_{k+1}), \tag{43a}$$

$$\dot{\boldsymbol{\xi}}_k(\tau) = \boldsymbol{\Lambda}_2(\tau)\hat{\boldsymbol{\gamma}}_k(t_k) + \boldsymbol{\Psi}_2(\tau)\hat{\boldsymbol{\gamma}}_k(t_{k+1}). \tag{43b}$$

The general formula for obtaining the interpolation Jacobians for perturbations to the pose and body-centric velocity is as follows:

$$\frac{\partial\delta\mathbf{T}(\tau)}{\partial\mathbf{x}} = \mathcal{J}_{\mathrm{op},\tau,k}\frac{\partial\boldsymbol{\xi}_k(\tau)}{\partial\mathbf{x}} + \mathcal{T}_{\mathrm{op},\tau,k}\frac{\partial\boldsymbol{\epsilon}_k}{\partial\mathbf{x}}, \tag{44a}$$

$$\frac{\partial\delta\boldsymbol{\varpi}(\tau)}{\partial\mathbf{x}} = \mathcal{J}_{\mathrm{op},\tau,k}\frac{\partial\dot{\boldsymbol{\xi}}_k(\tau)}{\partial\mathbf{x}} - \frac{1}{2}\dot{\boldsymbol{\xi}}_{\mathrm{op},\tau}^\curlywedge\frac{\partial\boldsymbol{\xi}_k(\tau)}{\partial\mathbf{x}}, \tag{44b}$$

where the Jacobians of the local variable $\boldsymbol{\xi}_k(\tau)$ with respect to state perturbations at the bracketing times are given by

$$\frac{\partial\boldsymbol{\xi}_k(\tau)}{\partial\boldsymbol{\epsilon}_{k+1}} = \boldsymbol{\Psi}_{11}\mathcal{J}_{\mathrm{op},k+1,k}^{-1} + \frac{1}{2}\boldsymbol{\Psi}_{12}\boldsymbol{\varpi}_{\mathrm{op},k+1}^\curlywedge\mathcal{J}_{\mathrm{op},k+1,k}^{-1}, \tag{45a}$$

$$\frac{\partial\boldsymbol{\xi}_k(\tau)}{\partial\boldsymbol{\epsilon}_k} = -\left(\frac{\partial\boldsymbol{\xi}_k(\tau)}{\partial\boldsymbol{\epsilon}_{k+1}}\right)\mathcal{T}_{\mathrm{op},k+1,k}, \tag{45b}$$

$$\frac{\partial\boldsymbol{\xi}_k(\tau)}{\partial\boldsymbol{\eta}_k} = \boldsymbol{\Lambda}_{12}, \tag{45c}$$

$$\frac{\partial\boldsymbol{\xi}_k(\tau)}{\partial\boldsymbol{\eta}_{k+1}} = \boldsymbol{\Psi}_{12}\mathcal{J}_{\mathrm{op},k+1,k}^{-1}. \tag{45d}$$

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.

B. 2024 Radar in Robotics Competition

At ICRA 2024 in Yokohama, Japan, we hosted a radar odometry competition as part of the Radar in Robotics workshop. In order to achieve our competition 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. 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 meter. Putting together all of these changes, our competition submission, STEAM-RIO++, achieved 0.62% translational drift, a 35% improvement over STEAM-RIO as shown in Table V. In the competition, we placed third, where the 2nd place submission was CFEAR with 0.61% drift [75]. Notably, CFEAR did not use an IMU, and their approach is computationally very efficient. In order to achieve their competition results, they greatly increased the length of their sliding window and employed a coarse-to-fine registration approach. The first-place entry was CFEAR++, achieving 0.51% drift [76]. CFEAR++ is based on CFEAR where the authors also used gyroscope angle estimates as a prior as well as semantic segmentation to retain only points that correspond to buildings.

TABLE V
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
Seq. Avg.	1.68 / 0.49	0.95 / 0.27	0.62 / 0.18
ΔT	115ms	139ms	153ms

REFERENCES

[1] K. Burnett, Y. Wu, D. J. Yoon, A. P. Schoellig, and T. D. Barfoot, "Are we ready for radar to replace lidar in all-weather mapping and localization?" *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 328–10 335, 2022.

[2] T. Lupton and S. Sukkarieh, "Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 61–76, 2012.

[3] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, 2017.

[4] M. Brossard, A. Barrau, P. Chauchat, and S. Bonnabel, "Associating uncertainty to extended poses for on lie group imu preintegration with rotating earth," *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 998–1015, 2022.

[5] I. Vizzo, T. Guadagnino, B. Mersch, L. Wiesmann, J. Behley, and C. 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*, vol. 8, no. 2, pp. 1029–1036, 2023.

[6] S. Anderson and T. 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)*, 2015, pp. 157–164.

[7] T. Y. Tang, D. J. Yoon, and T. D. Barfoot, "A white-noise-on-jerk motion prior for continuous-time trajectory estimation on se(3)," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 594–601, 2019.

[8] J. N. Wong, D. J. Yoon, A. P. Schoellig, and T. D. Barfoot, "Variational inference with parameter learning applied to vehicle trajectory estimation," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5291–5298, 2020.

[9] C. Le Gentil and T. Vidal-Calleja, "Continuous latent state preintegration for inertial-aided systems," *The International Journal of Robotics Research*, vol. 42, no. 10, pp. 874–900, 2023.

[10] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "Fast-lio2: Fast direct lidar-inertial odometry," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2053–2073, 2022.

[11] K. Chen, R. Nemiroff, and B. T. Lopez, "Direct lidar-inertial odometry: Lightweight lio with continuous-time motion correction," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 3983–3989.

[12] Y. Wu, D. J. Yoon, K. Burnett, S. Kammel, Y. Chen, H. Vhavle, and T. D. Barfoot, "Picking up speed: Continuous-time lidar-only odometry using doppler velocity measurements," *IEEE Robotics and Automation Letters*, vol. 8, no. 1, pp. 264–271, 2022.

[13] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[14] K. Burnett, D. J. Yoon, Y. Wu, A. Z. Li, H. Zhang, S. Lu, J. Qian, W.-K. Tseng, A. Lambert, K. Y. Leung, *et al.*, "Boreas: A multi-season autonomous driving dataset," *The International Journal of Robotics Research*, vol. 42, no. 1-2, pp. 33–42, 2023.

[15] M. Ramezani, Y. Wang, M. Camurri, D. Wisth, M. Mattamala, and M. 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)*. IEEE, 2020, pp. 4353–4360.

[16] D. Lee, M. Jung, W. Yang, and A. Kim, "Lidar odometry survey: Recent advancements and remaining challenges," *arXiv preprint arXiv:2312.17487*, 2023.

[17] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing icp variants on real-world data sets: Open-source library and experimental protocol," *Autonomous robots*, vol. 34, pp. 133–148, 2013.

[18] P. Dellenbach, J.-E. Deschaud, B. Jacquet, and F. Goulette, "Ct-icp: Real-time elastic lidar odometry with loop closure," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 5580–5586.

[19] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp." in *Robotics: science and systems*, vol. 2, no. 4. Seattle, WA, 2009, p. 435.

[20] J. Zhang and S. Singh, "Low-drift and real-time lidar odometry and mapping," *Autonomous Robots*, vol. 41, pp. 401–416, 2017.

[21] X. Chen, A. Milioto, E. Palazzolo, P. Gigure, J. Behley, and C. Stachniss, "Suma++: Efficient lidar-based semantic slam," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 4530–4537.

[22] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping," in *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2020, pp. 5135–5142.

[23] A. Tagliabue, J. Tordesillas, X. Cai, A. Santamaria-Navarro, J. P. How, L. Carlone, and A.-a. Agha-mohammadi, "Lion: Lidar-inertial observability-aware navigator for vision-denied environments," in *Experimental Robotics: The 17th International Symposium*. Springer, 2021, pp. 380–390.

[24] H. Ye, Y. Chen, and M. Liu, "Tightly coupled 3d lidar inertial odometry and mapping," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3144–3150.

[25] S. Clark and H. Durrant-Whyte, "Autonomous land vehicle navigation using millimeter wave radar," in *Proceedings. 1998 IEEE International Conference on Robotics and Automation (Cat. No. 98CH36146)*, vol. 4. IEEE, 1998, pp. 3697–3702.

[26] M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map build-

[27] E. Jose and M. D. 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)(IEEE Cat. No. 04CH37566)*, vol. 1. IEEE, 2004, pp. 425–430.

[28] P. Checchin, F. Gérossier, C. Blanc, R. Chapuis, and L. Trassoudaine, "Radar scan matching slam using the fourier-mellin transform," in *Field and Service Robotics: Results of the 7th International Conference*. Springer, 2010, pp. 151–161.

[29] D. Vivet, P. Checchin, and R. Chapuis, "Localization and mapping using only a rotating fmcw radar sensor," *Sensors*, vol. 13, no. 4, pp. 4527–4552, 2013.

[30] D. Kellner, M. Barjenbruch, J. Klappstein, J. Dickmann, and K. Dietmayer, "Instantaneous ego-motion estimation using doppler radar," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. IEEE, 2013, pp. 869–874.

[31] J. Callmer, D. Törnqvist, F. Gustafsson, H. Svensson, and P. Carlbom, "Radar slam using visual features," *EURASIP Journal on Advances in Signal Processing*, vol. 2011, no. 1, pp. 1–11, 2011.

[32] 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)*. IEEE, 2016, pp. 2559–2564.

[33] S. H. Cen and P. Newman, "Precise ego-motion estimation with millimeter-wave radar under diverse and challenging conditions," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6045–6052.

[34] Z. Hong, Y. Petillot, and S. Wang, "Radarslam: Radar based large-scale slam in all weathers," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5164–5170.

[35] D. Adolfsson, M. Magnusson, A. Alhashimi, A. J. Lilienthal, and H. 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*, vol. 39, no. 2, pp. 1476–1495, 2022.

[36] K. Harlow, H. Jang, T. D. Barfoot, A. Kim, and C. Heckman, "A new wave in robotics: Survey on recent mmwave radar applications in robotics," *arXiv preprint arXiv:2305.01135*, 2023.

[37] A. Kramer, C. Stahoviak, A. Santamaria-Navarro, A.-A. Agha-Mohammadi, and C. Heckman, "Radar-inertial ego-velocity estimation for visually degraded environments," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 5739–5746.

[38] Y. Almalioglu, M. Turan, C. X. Lu, N. Trigoni, and A. Markham, "Milli-rio: Ego-motion estimation with low-cost millimetre-wave radar," *IEEE Sensors Journal*, vol. 21, no. 3, pp. 3314–3323, 2020.

[39] Y. S. Park, Y.-S. Shin, J. Kim, and A. Kim, "3d ego-motion estimation using low-cost mmwave radars via radar velocity factor for pose-graph slam," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7691–7698, 2021.

[40] C. Doer and G. 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)*. IEEE, 2021, pp. 331–338.

[41] J. Michalczyk, R. Jung, and S. Weiss, "Tightly-coupled ekf-based radar-inertial odometry," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 12 336–12 343.

[42] J. Michalczyk, R. Jung, C. Brommer, and S. Weiss, "Multi-state tightly-coupled ekf-based radar-inertial odometry with persistent landmarks," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 4011–4017.

[43] H. Chen, Y. Liu, and Y. Cheng, "Drio: Robust radar-inertial odometry in dynamic environments," *IEEE Robotics and Automation Letters*, 2023.

[44] J.-T. Huang, R. Xu, A. Hinduja, and M. Kaess, "Multi-radar inertial odometry for 3d state estimation using mmwave imaging radar," *arXiv preprint arXiv:2311.08608*, 2023.

[45] Y. Z. Ng, B. Choi, R. Tan, and L. Heng, "Continuous-time radar-inertial odometry for automotive radars," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 323–330.

[46] Y. Zhuang, B. Wang, J. Huai, and M. Li, "4d iriom: 4d imaging radar inertial odometry and mapping," *IEEE Robotics and Automation Letters*, 2023.

[47] V. Kubelka, E. Fritz, and M. Magnusson, "Do we need scan-matching in radar odometry?" *arXiv preprint arXiv:2310.18117*, 2023.

[48] R. Rouveure, M. Monod, and P. Faure, "High resolution mapping of the environment with a ground-based radar imager," in *2009 International Radar Conference" Surveillance for a Safer World"(RADAR 2009)*. IEEE, 2009, pp. 1–6.

[49] J. Mullane, S. Keller, and M. Adams, "Random set versus vector based slam in the presence of high clutter," in *IEEE ICRA*, 2012.

[50] D. Vivet, F. Gérossier, P. Checchin, L. Trassoudaine, and R. Chapuis, "Mobile ground-based radar sensor for localization and mapping: An evaluation of two approaches," *International Journal of Advanced Robotic Systems*, vol. 10, no. 8, p. 307, 2013.

[51] M. Bosse, R. Zlot, and P. Flick, "Zebedee: Design of a spring-mounted 3-d range sensor with application to mobile mapping," *IEEE Transactions on Robotics*, vol. 28, no. 5, pp. 1104–1119, 2012.

[52] T.-M. Nguyen, D. Duberg, P. Jensfelt, S. Yuan, and L. Xie, "Slict: Multi-input multi-scale surfel-based lidar-inertial continuous-time odometry and mapping," *IEEE Robotics and Automation Letters*, vol. 8, no. 4, pp. 2102–2109, 2023.

[53] X. Zheng and J. Zhu, "Traj-lo: In defense of lidar-only odometry using an effective continuous-time trajectory," *arXiv preprint arXiv:2309.13842*, 2023.

[54] J.-E. Deschaud, "Imls-slam: Scan-to-model matching based on 3d data," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 2480–2485.

[55] P. Furgale, C. H. Tong, T. D. Barfoot, and G. Sibley, "Continuous-time batch trajectory estimation using temporal basis functions," *The International Journal of Robotics Research*, vol. 34, no. 14, pp. 1688–1710, 2015.

[56] A. Patron-Perez, S. Lovegrove, and G. Sibley, "A spline-based trajectory representation for sensor fusion and rolling shutter cameras," *International Journal of Computer Vision*, vol. 113, no. 3, pp. 208–219, 2015.

[57] D. Droeschel and S. Behnke, "Efficient continuous-time slam for 3d lidar-based online mapping," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 5000–5007.

[58] J. Quenzel and S. 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)*. IEEE, 2021, pp. 5499–5506.

[59] C. Park, P. Moghadam, J. L. Williams, S. Kim, S. Sridharan, and C. Fookes, "Elasticity meets continuous-time: Map-centric dense 3d lidar slam," *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 978–997, 2021.

[60] J. Lv, X. Lang, J. Xu, M. Wang, Y. Liu, and X. Zuo, "Continuous-time fixed-lag smoothing for lidar-inertial-camera slam," *IEEE/ASME Transactions on Mechatronics*, 2023.

[61] X. Lang, C. Chen, K. Tang, Y. Ma, J. Lv, Y. Liu, and X. Zuo, "Coco-lic: Continuous-time tightly-coupled lidar-inertial-camera odometry using non-uniform b-spline," *IEEE Robotics and Automation Letters*, 2023.

[62] S. Anderson, T. D. Barfoot, C. H. Tong, and S. Särkkä, "Batch nonlinear continuous-time trajectory estimation as exactly sparse gaussian process regression," *Autonomous Robots*, vol. 39, pp. 221–238, 2015.

[63] C. Le Gentil, T. Vidal-Calleja, and S. Huang, "In2laama: Inertial lidar localization autocalibration and mapping," *IEEE Transactions on Robotics*, vol. 37, no. 1, pp. 275–290, 2020.

[64] J. Johnson, J. Mangelson, T. Barfoot, and R. Beard, "Continuous-time trajectory estimation: A comparative study between gaussian process and spline-based approaches," *arXiv preprint arXiv:2402.00399*, 2024.

[65] W. Talbot, J. Nubert, T. Tuna, C. Cadena, F. Dmbgen, J. Tordesillas, T. D. Barfoot, and M. Hutter, "Continuous-time state estimation methods in robotics: A survey," *arXiv preprint arXiv:2411.03951*, 2024.

[66] R. Nemiroff, K. Chen, and B. T. Lopez, "Joint on-manifold gravity and accelerometer intrinsics estimation," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023.

[67] A. Alhashimi, D. Adolfsson, M. Magnusson, H. Andreasson, and A. J. Lilienthal, "Bfar-bounded false alarm rate detector for improved radar odometry estimation," *arXiv preprint arXiv:2109.09669*, 2021.

[68] H. Rohling, "Radar cfar thresholding in clutter and multiple target situations," *IEEE transactions on aerospace and electronic systems*, no. 4, pp. 608–621, 1983.

[69] K. Burnett, A. P. Schoellig, and T. D. Barfoot, "Do we need to compensate for motion distortion and doppler effects in spinning radar navigation?" *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 771–778, 2021.

[70] T. D. Barfoot, *State Estimation for Robotics, 2nd Edition*. Cambridge University Press, 2024.

[71] K. Chen, R. Nemiroff, and B. T. Lopez, "Direct lidar-inertial odometry and mapping: Perceptive and connective slam," *arXiv preprint arXiv:2305.01843*, 2023.

[72] T.-M. Nguyen, S. Yuan, M. Cao, Y. Lyu, T. H. Nguyen, and L. Xie, "Ntu viral: A visual-inertial-ranging-lidar dataset, from an aerial vehicle

viewpoint," *The International Journal of Robotics Research*, vol. 41, no. 3, pp. 270–280, 2022.

[73] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. 13, no. 04, pp. 376–380, 1991.

[74] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 573–580.

[75] D. Adolfsson and M. Hilger, "An evaluation of cfear radar odometry," *arXiv preprint arXiv:2404.01781*, 2024.

[76] S. Li, Z. Hong, Y. Chen, L. Hu, and J. Qin, "Get it for free: Radar segmentation without expert labels and its application in odometry and localization," *arXiv preprint arXiv:2409.18434*, 2024.

Keenan Burnett received the B.A.Sc. degree in engineering science from University of Toronto, Toronto, ON, Canada, in 2017 and the M.A.Sc. degree in aerospace engineering from the University of Toronto in 2020. He is a Ph.D. student with the University of Toronto Robotics Institute. His work to date has centered around autonomous vehicles. His current research focuses on mapping and localization in adverse weather conditions through the application of radar and lidar. He is interested in extending the operational capabilities of robots to new and challenging environments.

Angela P. Schoellig received the M.Sc. degree in engineering science and mechanics from the Georgia Institute of Technology, Atlanta, GA, USA, in 2007, the master's degree in engineering cybernetics from the University of Stuttgart, Stuttgart, Germany, in 2008, and the Ph.D. degree in mechanical and process engineering from ETH Zurich, Zurich Switzerland, in 2012. She is currently an Assistant Professor with the Institute for Aerospace Studies, University of Toronto, North York, ON, Canada, where she leads the Dynamic Systems Laboratory. Her general areas of interest lie at the interface of robotics, controls, and machine learning.

Timothy D. Barfoot received the B.A.Sc. degree in engineering science from University of Toronto, Toronto, ON, Canada, in 1997 and the Ph.D. degree in aerospace science and engineering from University of Toronto, in 2002. He is a Professor with the University of Toronto Robotics Institute, Toronto, ON, Canada. He works in the areas of guidance, navigation, and control of autonomous systems in a variety of applications. He is interested in developing methods to allow robotic systems to operate over long periods of time in large-scale, unstructured, three-dimensional environments, using rich onboard sensing (e.g., cameras, laser rangefinders, and radar) and computation.