Picking up Speed: Continuous-Time Lidar-Only Odometry Using Doppler Velocity Measurements

Yuchen Wu, David J. Yoon, Keenan Burnett, Graduate Student Member, IEEE, Soeren Kammel, Yi Chen, Heethesh Vhavle, Member, IEEE, and Timothy D. Barfoot, Fellow, IEEE

Abstract—Frequency-Modulated Continuous-Wave (FMCW) lidar is a recently emerging technology that additionally enables per-return instantaneous relative radial velocity measurements via the Doppler effect. In this letter, we present the first continuous-time lidar-only odometry algorithm using these Doppler velocity measurements from an FMCW lidar to aid odometry in geometrically degenerate environments. We apply an existing continuous-time framework that efficiently estimates the vehicle trajectory using Gaussian process regression to compensate for motion distortion due to the scanning-while-moving nature of any mechanically actuated lidar (FMCW and non-FMCW). We evaluate our proposed algorithm on several real-world datasets, including publicly available ones and datasets we collected. Our algorithm outperforms the only existing method that also uses Doppler velocity measurements, and we study difficult conditions where including this extra information greatly improves performance. We additionally demonstrate state-of-the-art performance of lidar-only odometry with and without using Doppler velocity measurements in nominal conditions.

Index Terms—SLAM, range sensing.

I. INTRODUCTION

MULTI-BEAM lidars have become a common addition to the sensor suite of an autonomous vehicle. Estimation algorithms to handle the long-range 3D measurements (i.e., point clouds) produced by these sensors have also matured, and are capable of producing highly accurate motion estimates, often at a sub-decimeter level of accuracy for localization [1].

Lidar motion estimation performs exceptionally well when there exists sufficient geometric structure in the surroundings to uniquely constrain all six degrees of freedom of the vehicle pose. In contrast, even the best estimators will struggle and even fail in geometrically degenerate environments. Long tunnels, highways with a barren landscape, and bridges are typical examples of extreme conditions where prior knowledge of the vehicle kinematics is insufficient to compensate for the lack of geometric information. A common solution in such situations is to rely on an additional sensor such as an Inertial Measurement Unit (IMU) [2], [3].

Frequency-Modulated Continuous Wave (FMCW) lidar is a recently emerging technology [4], [5] that provides a promising alternative solution for geometrically degenerate environments. While capable of producing dense point clouds comparable to a typical time-of-flight lidar, FMCW lidars also measure the relative velocity between each measured point along the radial direction via the Doppler effect, which we call a Doppler velocity measurement. Fig. 1 depicts an example lidar frame colored by the measured relative radial velocity (i.e., Doppler velocity). Vehicles driving in the same direction are colored in magenta as they have the same velocity as the ego-vehicle, while vehicles driving in the opposite direction are in red. They can be clearly distinguished from the underlying static scene. Our proposed algorithm uses the relative radial velocity with respect to the static scene to aid odometry. Moving objects are naturally rejected by our algorithm using robust estimation techniques.

Throughout this letter, we refer to the aggregation of points over one full lidar field-of-view as a lidar frame.

Manuscript received 25 August 2022; accepted 27 October 2022. Date of publication 1 December 2022; date of current version 8 December 2022. This letter was recommended for publication by Associate Editor T. Stoyanov and Editor J. Civera upon evaluation of the reviewers’ comments. (Corresponding author: Yuchen Wu.)

Yuchen Wu, David J. Yoon, Keenan Burnett, and Timothy D. Barfoot are with the Institute for Aerospace Studies (UTIAS), University of Toronto, Toronto, ON M3H 5T6, Canada (e-mail: yuchen.wu@robotics.utias.utoronto.ca; david.yoon@robotics.utias.utoronto.ca; keenan.burnett@robotics.utias.utoronto.ca; tim.barfoot@utoronto.ca).

Soeren Kammel, Yi Chen, and Heethesh Vhavle are with the Aeva Inc., Mountain View, CA 94043 USA (e-mail: soeren@aeva.ai; yichen.um@gmail.com; heethesh@gmail.com).

Code for this project can be found at: https://github.com/utiasASRL/steam_icp.

Digital Object Identifier 10.1109/LRA.2022.3226068
In this letter, we improve upon the existing work [6] by incorporating the Doppler velocity measurements in a continuous-time estimation framework. Continuous-time estimation allows for each measurement to be associated with its actual time of acquisition, avoiding the need for an IMU to correct the motion distortion of a lidar frame due to the scanning-while-moving nature of mechanically actuated lidars. Similar to [6], we present a Doppler velocity factor that can be applied in conjunction with the usual point-to-plane factor for frame-to-map alignment. However, our proposed factor differs from theirs in that it is applied to the vehicle’s body-centric velocity as opposed to pose since body-centric velocity is also part of our estimated state. We evaluate our lidar odometry algorithm on several real-world datasets, including publicly available ones collected using a non-FMCW lidar and datasets we collected using an FMCW lidar. Through our evaluation, we demonstrate overall state-of-the-art lidar-only odometry performance with and without using Doppler velocity measurements under both nominal and geometrically degenerate conditions.

II. RELATED WORK

Lidar motion estimation typically adopts a point cloud registration approach using a variant of ICP [7], [8]. Most lidar motion estimation pipelines can be divided into a data processing front-end and a state estimation back-end [9].

The front-end processes the raw lidar frames, which includes keypoint/feature extraction, global/local map building, and data association. LOAM [2] extracts edge and plane features and matches them via nearest-neighbor association. SuMa [10] matches raw frames to a surfel map using projective data association. Recently, Vizzo et al. [11] introduced a triangle mesh map representation and a ray-casting-based point-to-mesh association approach. Other alternatives have been proposed in [12], [13], [14], [15], [16], [17], [18], though we do not discuss them in detail in the interest of space.

The back-end leverages the processed data to estimate the vehicle state over time, seen as the vehicle trajectory. Existing approaches are different in how they formulate and estimate the trajectory. Discrete-time estimators formulate a trajectory where there is a temporal state (i.e., marginal) that corresponds to the acquisition time of every measurement. However, typical multi-beam lidars are mechanically actuated and produce thousands of measurements for every lidar frame. Consequently, each measurement may have a unique timestamp. Even after preprocessing into sparser keypoints, it is often not feasible to have a discrete state estimate at each measurement time. One option is to correct for the motion of each frame using an additional sensor such as an IMU [2], [3].

Alternatively, the vehicle trajectory can be estimated as a continuous function of time. The most straightforward approach is to apply linear interpolation between discrete states [2], [19], [20]. However, using linear interpolation between poses restricts the trajectory to have a piece-wise constant velocity, which cannot accurately represent trajectories undergoing high-frequency changes in velocity. Dellenbach et al. [21] address this limitation by allowing the vehicle poses to be discontinuous between lidar frames, trading trajectory smoothness for a closer representation. Another option for higher representational power without losing smoothness is to represent the trajectory using temporal basis functions [22], [23]. Referred to as a parametric methods, the trajectory is parameterized by the associated basis-function weights and has been demonstrated several times for lidar motion estimation using B-splines [24], [25], [26], [27].

Nonparametric methods instead estimate the continuous-time trajectory as a Gaussian process (GP) [28]. Barfoot et al. [29] demonstrated that through careful selection of the underlying GP prior and a Markovian state, we can benefit from the usual sparsity exploited in discrete-time estimation. Anderson and Barfoot [30] extended this idea for trajectory estimation in \( SE(3) \). A desirable feature of their approach is that the GP prior is made to be physically motivated (e.g., white noise on acceleration). Lidar motion estimation has been demonstrated several times using various GP priors [1], [31], [32], [33].

All works discussed thus far consider lidars that only produce point clouds (optionally with intensity information). Advances in FMCW technology have enabled a new type of lidar, i.e., the Aeva Aeries I FMCW Lidar [34], that can additionally measure the relative radial velocity (Doppler velocity) of each point. Hexsel et al. [6] presented an ICP-based algorithm that uses the Doppler velocity measurements and demonstrated improved performance in environments with insufficient geometric structure.

The Aeva Aeries I FMCW Lidar is mechanically actuated, similar to existing multi-beam lidars. Hexsel et al. [6] applied a discrete-time estimator, which requires the lidar frames to be corrected for motion beforehand using an IMU. We improve upon their work by formulating the back-end estimator using GP regression in \( SE(3) \) [30]. Although the GP regression approach has been applied to lidodometry before [1], [31], [32], [33], the benefits of including Doppler velocity measurements have not been demonstrated. Compared to the work of Hexsel et al. [6], we demonstrate state-of-the-art performance in lidar odometry without using additional sensors.

Furthermore, the FMCW technology can also be applied to radars, and there has been some work using an FMCW radar for motion estimation [35], [36]. Both Kellner et al. [35] and Vivet et al. [36] use the Doppler velocity measurements from an FMCW radar to estimate the velocity of the ego-vehicle, which is the same as what they are used for in this work.

III. METHODOLOGY

Our lidar odometry algorithm adopts the conventional ICP-based frame-to-map approach while incorporating Doppler velocity measurements in a sliding-window implementation.

A. Data Processing Front-End

We follow the approach of Dellenbach et al. [21]. Keypoints are extracted from each lidar frame via voxel grid downsampling. We use a grid size of 1.5 m and keep one random point in each voxel. Our local map is a point cloud accumulated from the most recent frames and cropped to be within 100 m of the latest estimate of the vehicle after each frame update. The local
map is stored in a sparse voxel grid structure with a 1 m grid size and a maximum of 20 points per voxel. We use point-to-plane ICP for frame-to-map matching. Each frame point is associated with a map point via nearest-neighbor association, and the corresponding plane normal is computed by applying Principle Component Analysis (PCA) to the 20 closest neighbors of the associated map point.

B. Trajectory Estimation Back-End

**Notation:** We denote \( F_{\text{i}} \) to be the inertial reference frame, \( F_{\text{v}} \) to be the vehicle reference frame, and \( F_{\text{l}} \) to be the lidar reference frame. Let \( T_{\text{i}l} \in SE(3) \) be the inertial-to-vehicle transformation (i.e., the vehicle pose) and \( \sigma_{\text{iv}} = [\nu_{\text{iv}}^T \ \omega_{\text{iv}}^T]^T \in \mathbb{R}^6 \) be the vehicle body-centric velocity where \( \nu_{\text{iv}} \) and \( \omega_{\text{iv}} \) are translational and rotational velocities, respectively.\(^2\) Furthermore, let \( T_{\ell v} \) be the fixed vehicle-to-lidar transformation assuming the lidar is rigidly mounted on the vehicle, \( q \) be a homogeneous point from a lidar frame (expressed in \( F_{\text{l}} \)), and \( p \) be a homogeneous point in the local map (expressed in \( F_{\text{i}} \)).

We follow previous work \([30]\) to represent a continuous-time trajectory as a Gaussian process (GP). Our trajectory is \( x(t) = \{ T_{\text{iv}}(t), \sigma_{\text{iv}}(t) \} \). We use the White-Noise-on-Acceleration (WNOA) motion prior

\[
\dot{T}_{\text{iv}}(t) = \sigma_{\text{iv}}(t) T_{\text{iv}}(t),
\]

\[
\dot{\sigma}_{\text{iv}}(t) = w(t), \quad w(t) \sim \mathcal{GP}(0, Q, \delta(t - \tau)),
\]

where \( w(t) \in \mathbb{R}^6 \) is a zero-mean, white-noise GP. The prior is applied in a piecewise fashion across an underlying discrete trajectory of pose-velocity state pairs, \( x_k \), each corresponding to the end time of the \( k \)th lidar frame.\(^3\) Note that the period of the \( k \)th lidar frame lies between \( x_{k-1} \) and \( x_k \). For a keypoint \( q_j \) with acquisition time \( t_j \) where \( t_{k-1} < t_j < t_k \), we associate it to an interpolated state \( x(t_j) \) that depends on \( x_{k-1} \) and \( x_k \) through GP interpolation. Interpolation is done efficiently through our choice of motion prior and Markovian state \([30]\).

We jointly align the most recent five frames to the local map in a sliding window by optimizing for the states \( x_{k-5} \) to \( x_k \). Fig. 2 illustrates the states and factors in our sliding-window estimation with an example window size of two. We query the trajectory states at the acquisition time of each keypoint in each lidar frame. The motion prior factor \( \phi_{\text{mp}} \) connects neighboring trajectory states. The point-to-plane factor \( \phi_{\text{p2p}} \) requires local map information while the Doppler velocity factor \( \phi_{\text{dv}} \) does not. For computational efficiency, we implement this sliding-window estimation by introducing one estimated state at the end of each lidar frame and using GP interpolation to obtain states at other measurement times. This way, motion prior factors between interpolated and estimated/interpolated states are merged into factors between estimated states. Point-to-plane and Doppler velocity factors applied to interpolated states depend on the adjacent estimated states. Past states falling outside the window are marginalized and no longer updated.

2) **Doppler Velocity Factor:** The Doppler velocity factor is

\[
\phi_{\text{dv}} = \rho \left( \sqrt{\frac{\beta e_{\text{dv}}^2}{\delta}} \right),
\]

\[
e_{\text{dv}} = \dot{r} - \frac{D^T D}{(q^T D^T D q)^{1/2}} D q \ T_{\ell v} \sigma_{\text{iv}}(t),
\]

where \( \sigma_{\text{iv}}(t) \) is the vehicle body-centric velocity queried at \( t \), \( D \) is the same projection as above, \( \dot{r} \) is the Doppler velocity measurement associated with \( q \) measured at time \( t \). \( T_{\ell v} = \text{Ad}(T_{\ell v}) \) is the adjoint matrix of \( T_{\ell v} \), and \( (\cdot)^\circ \) operator converts a homogeneous point to a \( 4 \times 6 \) matrix, as defined in \([37, p.246]\). \( \beta = 0.1 \) is a constant heuristic weight and \( \rho(\cdot) \) is again the Cauchy robust cost function with \( c = 0.05 \). We discard any measurement resulting in \( e_{\text{dv}} \geq 2 \text{ m/s} \).

To derive our error term \( (5) \), we first define \( q \) and \( q_i \) to be \( q \)’s velocity expressed in the lidar and inertial reference frame, respectively. Applying the transport theorem,

\[
q_i = T_{\ell v}(t) \ (q - \sigma_{\text{iv}}(t)^\circ q) .
\]

Assuming \( q \) is static in the inertial frame such that \( q_0 = 0 \), we can rearrange \( (6) \) to be (see \( SE(3) \) identities in \([37]\))

\[
q = \sigma_{\text{iv}}(t)^\circ q
= q^\circ \sigma_{\text{iv}}(t)
= q^\circ T_{\ell v} \sigma_{\text{iv}}(t),
\]

where \( (7) \) relates the point velocity \( q \) to the body-centric velocity component of our trajectory state \( \sigma_{\text{iv}} \). Next, we project \( q \) onto

---

\(^2\)The superscripts and subscripts follow the convention in \([37]\).

\(^3\)\(x_0\) is at the start of the first frame.
Aeva Aeries I FMCW Lidar has a horizontal field-of-view of 30°, a 300 m maximum operating range, a Doppler velocity measurement precision of 3 cm/s, and a sampling rate of 10 Hz.

Fig. 3. This diagram provides a graphical illustration of the Doppler velocity error term derivation. $\mathcal{F}_q$, $\mathcal{F}_v$, and $\mathcal{F}_\ell$ are the inertial, vehicle, and lidar reference frames, respectively. $\mathbf{q}$ and $\mathbf{q}$ are the measured point position and velocity expressed in $\mathcal{F}_q$, respectively. $\mathbf{d}$ is a unit vector along the direction of $\mathbf{q}$, and $\tilde{\mathbf{r}}$ is the relative radial velocity according to $\omega_t^\ell = \begin{bmatrix} \nu_t^\ell & \omega_t^\ell \end{bmatrix}^T$. See Section III-B for the expression of $\mathbf{d}$ and $\tilde{\mathbf{r}}$.

We treat sliding-window estimation as a factor-graph optimization problem and solve it iteratively using Gauss-Newton with Iteratively Reweighted Least Squares (IRLS) [41]. Rather than naively dropping states out of the sliding window, we explicitly marginalize out each old state. Each new state is initialized based on the GP extrapolation of our motion prior (i.e., constant velocity). The point-to-plane association of keypoints in the latest frame and the plane normals are re-computed after every five Gauss-Newton iterations.

Finally, taking the difference between the predicted and measured relative radial velocity gives the error function $e_{dv}$ in (5). This derivation is graphically illustrated in Fig. 3.

At this point, we highlight the characteristics of our Doppler velocity factor. Firstly, the factor does not require the local map. Secondly, the rotational velocity of the lidar frame $\omega_t^\ell$ is unfortunately not observable. Finally, by having the vehicle’s body-centric velocity as part of the state, the Doppler velocity measurement is incorporated directly into our estimator without further approximations or assumptions. While the first two characteristics have been observed by Hexsel et al. [6], the last one is unique to our formulation. Haxel et al. [6] derived a relation between Doppler velocity and vehicle pose using a constant-velocity approximation within the period of each frame, which is not needed in our formulation.

We treat sliding-window estimation as a factor-graph optimization problem and solve it iteratively using Gauss-Newton with Iteratively Reweighted Least Squares (IRLS) [41]. Rather than naively dropping states out of the sliding window, we explicitly marginalize out each old state. Each new state is initialized based on the GP extrapolation of our motion prior (i.e., constant velocity). The point-to-plane association of keypoints in the latest frame and the plane normals are re-computed after every five Gauss-Newton iterations.

IV. Experiments

A. Datasets

We demonstrate our method and make comparisons to existing work on several real-world data sequences, which we categorize into two datasets: KITTI-raw/360 and Aeva. All data sequences provide accurate pose estimates from a GNSS-INS system for evaluation.

1) KITTI-Raw/360 Dataset: contains 19 sequences of raw lidar frames from the publicly available KITTI dataset [42] and its successor KITTI-360 [43]. The lidar used in this dataset is a Velodyne HDL-64, which is not an FMCW lidar. The sequences in this dataset were collected from urban/suburban environments and are rich in geometric structure for lidar motion estimation.

We apply our estimator to the variation of the dataset made available by Dellenbach et al. [21].

2) Aeva Dataset: contains 8 sequences of lidar frames from an Aeva Aeries I FMCW Lidar that produces Doppler velocity measurements. Aeva Aeries I FMCW Lidar has a horizontal field-of-view of 120°, a vertical field-of-view of 30°, a 300 m maximum operating range, a Doppler velocity measurement precision of 3 cm/s, and a sampling rate of 10 Hz.

Sequences 00-03 are relatively short sequences purposely collected in environments with poor geometric structure (e.g., tunnels, freeways), which have been used in previous work [6], but required an additional IMU to correct for the scanning-while-moving nature of the sensor. Sequences 04-07 are longer sequences collected using our data collection platform, Boreas (Fig. 5), on Toronto, Ontario highways with moderate geometric structure. Table I has a summary of the dataset statistics and Fig. 4 shows some representative scenes.

B. Evaluation

We use the KITTI Relative Translation Error (KITTI RTE), which averages the translation error over path segments of lengths 100 m to 800 m in 100 m intervals. For the Aeva dataset, we additionally include evaluation using the Frame-to-Frame Relative Translation Error (Frame-to-Frame RTE) metric [44], which was used in [6]. In the interest of space, we omit showing the rotation error in the main text since the translation error

Note that $q^T D^T D q$ will have its last three elements being zero, which masks the rotational component of $\omega_t^\ell$. It implies that we cannot rely solely on Doppler velocity measurements to recover the full vehicle state (pose/velocity).

### Table I

| Seq. #   | Location                   | Length [m] | Num. of Frames | Geometric Structure |
|----------|----------------------------|------------|----------------|---------------------|
| 00       | Baker-Barry Tunnel (Empty) | 860        | 837            | Poor                |
| 01       | Baker-Barry Tunnel ( Vehicles) | 907        | 658            | Poor                |
| 02       | Robin Williams Tunnel      | 689        | 301            | Poor                |
| 03       | Brisbane Lagoon Freeway    | 4942       | 1762           | Poor                |
| 04       | Ontario Highway 7          | 8876       | 6343           | Moderate            |
| 05       | Ontario Highway 407        | 7836       | 4734           | Moderate            |
| 06       | Don Valley Parkway         | 10310      | 5083           | Moderate            |
| 07       | Ontario Highway 427        | 7238       | 4012           | Moderate            |

5For ground truth pose generation, we refer the readers to [42] for KITTI-raw/360, [6] for Aeva sequence 00-03, and [40] for Aeva sequence 04-07.

6We have published these sequences, which can be found at https://github.com/utiasASRL/steam_icp.
Fig. 4. Representative scenes in the Aeva dataset. The straight tunnels (a) and (b) have poor geometric structure to constrain the vehicle motion in the longitudinal direction. The Ontario highway sequences have moderate geometric structure from buildings and vegetation by the side of the highway.

Fig. 5. Our platform, Boreas, is previously used to collect the https://www.boreas.atias.utoronto.caBoreas dataset [40] and additionally equipped with an Aeva Aeries I FMCW Lidar [34] for use in this work.

is where our comparisons differ the most. Interested readers can find our full evaluation results in our supplementary material [45].

We name the two variants of our algorithm using and not using the Doppler velocity measurements STEAM-DICP and STEAM-ICP, respectively, after the continuous-time trajectory estimation framework, STEAM, presented in [30].

On the KITTI-raw/360 dataset, we compare STEAM-ICP with CT-ICP [21]. CT-ICP, to the best of our knowledge, is the current state-of-the-art continuous-time lidar-only odometry algorithm. It achieves similar performance on the KITTI odometry benchmark as other state-of-the-art methods (e.g., LOAM [2], IMLS-SLAM [38], and MULLS [18]) using lidar frames without motion distortion and demonstrates better performance using lidar frames with motion distortion. On the Aeva dataset, we compare STEAM-DICP with STEAM-ICP, CT-ICP, and Doppler-ICP [6]. Note that Doppler-ICP is the only existing algorithm that also uses Doppler velocity measurements to aid lidar odometry.

C. Results

Table II shows quantitative results on the KITTI-raw/360 dataset. Compared to CT-ICP, our odometry achieves comparable performance, demonstrating that we are performing at state of the art using non-FMCW lidars (i.e., no Doppler velocity measurements). Example plots of the estimated trajectory are shown in Fig. 6.

Quantitative results on the Aeva dataset are shown in Table III. On the difficult sequences with insufficient geometric structure (i.e., sequences 00-03), both CT-ICP and STEAM-ICP (i.e., methods that do not use Doppler velocity measurements) perform poorly. Fig. 7 highlights the problem in sequence 02 (Robin Williams Tunnel), where we see that the Frame-to-Frame RTE for both CT-ICP and STEAM-ICP increases dramatically during the tunnel portion of the sequence.

On sequences with moderate geometric structure (i.e., sequences 04-07), methods that do not use the velocity measurements perform well. Doppler-ICP performs worse overall due to using a frame-to-frame approach (i.e., no accumulated local maps) and not accounting for the motion in each lidar frame. STEAM-DICP performs similarly to STEAM-ICP and CT-ICP, showing that the velocity measurements are not as impactful on sequences with sufficient geometry. However, we still see minor improvements in Frame-to-Frame RTE, suggesting that the velocity measurements overall help improve performance.

Another way to highlight the benefit of the Doppler velocity measurements is to limit the range of the lidar artificially. The majority of the distinctive structure in the environment is off-road and further away from the sensor, therefore limiting the range increases the difficulty for estimation. We see in the Range-Limited section of Table III that methods using the Doppler velocity measurements, Doppler-ICP and STEAM-DICP, are the most robust to this artificial increase in difficulty, with ours (STEAM-DICP) performing the best.

D. Implementation

We implemented our back-end (see Section III-B) for continuous-time trajectory estimation using an open-source C++ library, STEAM7 [30]. The front-end that processes the lidar data was made to be the same as CT-ICP by using the C++ library

7STEAM library: https://github.com/utiasASRL/steam
that the authors have made publicly available.\textsuperscript{8} We use the same parameters as CT-ICP for keypoint extraction, neighborhood search, normal estimation, and map building to maintain a fair comparison. Parameters of the trajectory estimation back-end were determined empirically.

Our implementation is currently not quite real-time capable, running at approximately 5 Hz on the KITTI sequences and approximately 2 Hz on the Aeva sequences after incorporating the Doppler velocity measurements.\textsuperscript{9} The current bottleneck is the processing of each motion prior and measurement factor in STEAM, which was designed with more focus on generalization rather than computational speed. We strongly believe that a

\textsuperscript{8}CT-ICP library: https://github.com/jedeschaud/ct_icp

\textsuperscript{9}We run experiments using an Intel Xeon E5-2698 v4 2.2 GHz processor with 20 physical cores. We parallelize trajectory interpolation and Jacobian computation using 20 threads.
real-time capable implementation can be achieved by implementing an estimator specific to our problem, e.g., by hardcoding Jacobian computations instead of relying on STEAM’s built-in automatic differentiation.

V. CONCLUSION

In this letter, we presented a continuous-time lidar odometry algorithm leveraging the Doppler velocity measurements from an FMCW lidar to aid odometry. Our algorithm combines the lidar data processing front-end in [21] with our STEAM continuous-time estimator as the back-end, efficiently estimating the vehicle trajectory using GP regression. Through continuous-time estimation, our algorithm handles the motion distortion problem of mechanically actuated lidars. Incorporating the Doppler velocity information helps prevent odometry failures under geometrically degenerate conditions. By estimating vehicle pose and body-centric velocity, Doppler velocity measurements are incorporated directly into the estimator without further approximations or assumptions. Using both publicly available and our own datasets, we demonstrated state-of-the-art lidar odometry performance under nominal conditions with and without velocity information. On kilometer-scale highway sequences, we demonstrated superior performance over the only existing lidar odometry method that also uses Doppler velocity information under nominal and geometrically degenerate conditions.

There are multiple directions for future work to improve the proposed algorithm further. Currently, we apply robust cost functions to the point-to-plane error $e_{p2p}$ and the Doppler velocity error $e_{dv}$ individually for outlier rejection. Alternatively, one can apply a single robust cost function to $[e_{p2p} \quad e_{dv}]^T$ to reject outliers using both sources of information (in the same spirit as the Dynamic Point Outlier Rejection scheme in [6]). The velocity measurements can also be used by the data processing front-end to segment and remove points from moving objects before data association (e.g., by using the method of [46]). In addition, one can replace the WNOA GP prior used in this work with the WNOJ prior [32] or the Singer prior [33], which have shown to improve lidar odometry in nominal conditions.

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 Robot. Automat. Lett., vol. 7, no. 4, pp. 10328–10335, Oct. 2022.

[2] J. Zhang and S. Singh, “Low-drift and real-time LiDAR odometry and mapping,” Auton. Robots, vol. 41, no. 2, pp. 401–416, 2017.
