WING: Wheel-Inertial Neural Odometry with Ground Manifold Constraints

Kunyi Zhang[1,3,*], Chenxing Jiang[2,3,4,*], Sheng Yang[3], Shaojie Shen[2], Chao Xu[1,4], and Fei Gao[1,4]

***Abstract*—In this paper, we propose an interoceptive-only odometry system with neural network processing and globally continuous ground constraints specifically designed for ground robots. Exteroceptive sensors such as GPS, cameras, and LiDAR may encounter difficulties in scenarios with poor illumination and indoor environments. On the other hand, interoceptive sensors like IMU and wheel encoders may suffer from large drift due to noisy measurements. To overcome these challenges, the proposed system trains deep neural networks to estimate the biases of the IMU and the kinematics of the wheel encoders, while considering their uncertainty. Moreover, because the ground robot can only travel on the ground, we also model the ground surface as a globally continuous using dual cubic B-spline manifold to further improve the estimation accuracy by this constraint. A novel space-based sliding window filtering framework is proposed to fully exploit the C^1 continuity of ground manifold constraints and fuse all the information from raw measurements and neural networks in a yaw-independent attitude convention. Experiments in complex and large-scale urban environments demonstrate that our proposed approach can outperform state-of-the-art learning-based interoceptive-only odometry methods.**

***Index Terms*—Ground robot, State estimation, Deep learning.**

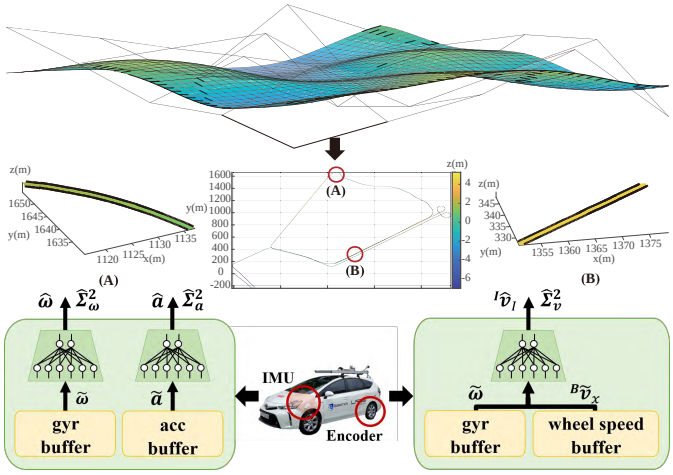

Fig. 1. The pipeline of our proposed odometry. Our proposed odometry estimates pose by processing the interoceptive sensor data through neural networks while simultaneously reconstructing a continuous cubic B-spline surface. The middle row of the figure shows the estimated position and terrain of scenario *urban17* in the KAIST Urban Dataset [15], where **A** and **B** represent the corner and straight routes, respectively.

I. Introduction

GROUND robots have been widely used in autonomous driving and space exploration. Localization is a critical component of autonomous robot navigation and has been extensively studied for decades. Most localization algorithms rely on exteroceptive sensors such as GPS [1], LiDAR [2]–[4], and camera [5]–[9]. Even though these methods can tackle most conditions, these exteroceptive sensors provide limited or interfering information in some degrading scenarios such as GPS-denied environments, poor illumination, severe weather, and repetitive surroundings. These scenarios can lead to inaccurate pose estimation, which can significantly affect the robot's performance.

In contrast, interoceptive sensors like IMU or wheel encoders are not influenced by these environmental impacts. IMU can provide six degrees of freedom (6DOF) relative motion constraints using the pre-integration technique [10] for localization [11]–[14]. However, raw IMU measurements often suffer from significant drift after several integrations without real-time calibration of intrinsic parameters. Besides, wheel encoders provide measurements of the distance traveled by two wheels. By simplifying the ground robot to a two-wheel [16], [17] or bicycle [18], [19] model, we can get the 2DOF relative motion of the ground robot. However, these models rely on assumptions such as no skidding or vertical bouncing [20], which may not hold in all scenarios. Consequently, relying solely on the raw measurements from interoceptive sensors like IMU and wheel encoders may not always provide accurate and robust localization for ground robots.

Therefore, to further improve the robustness and accuracy of localization under various failures or degradations of the exteroceptive sensors, researchers have proposed data-driven approaches to enhance the performance of inertial odometry [20]–[23] or wheel-inertial odometry [24]–[26]. However, these approaches do not leverage the ground manifold constraints, which is an important property of ground robots. As ground robots can only travel on a ground surface, leveraging this ground manifold constraint could help improve the accuracy of pose estimation [17], [27]–[29].

In this work, we propose a learning-based wheel-inertial odometry system that utilizes neural networks to learn the

This work was supported by the National Key Research and Development Program of China (Grant NO. 2020AAA0108104), Alibaba Innovative Research (AIR) Program, and the National Natural Science Foundation of China under Grant 62003299. (*Corresponding author: Fei Gao.*)

[1]State Key Laboratory of Industrial Control Technology, Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou 310027, China.

[2]Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology.

[3]Alibaba DAMO Academy, Hangzhou 311121, China.

[4]Huzhou Institute, Zhejiang University, Huzhou 313000, China.

E-mail: {kunyizhang, fgaoaa}@zju.edu.cn

[*]The authors have the same contribution.

robot's motion pattern and consider the ground manifold constraint. The neural networks enable the system to modify the IMU and wheel encoder measurements, implicitly perform the online calibration of these two sensors, and provide time-varying uncertainty estimation to achieve better sensor fusion. Additionally, to better portray the consistency and continuity of the ground robot during its motion, we leverage a globally continuous cubic B-spline manifold to model the ground manifold to constrain the position and tilt of the ground robot. The cubic B-spline manifold is different from the segmented planar surface [16] or the segmented curved surface [17] used in previous works, which can provide globally continuous constraints rather than locally discrete constraints. Furthermore, a sliding window filtering framework is proposed to fuse all the measurements and constraints mentioned above. The proposed work contributes as follows:

1) The proposed system uses neural networks to learn the robot's motion pattern from the raw measurements of IMU and wheel encoder and provide the time-varying uncertainty for better sensor fusion;
2) The proposed system parameterizes the ground manifold using a globally continuous cubic B-spline surface. This surface is used to enforce analytical constraints, which capture the continuity of the robot's motion and improve the accuracy of the pose estimation;
3) The proposed system utilizes a space-based sliding window filtering framework, which fuses all the measurements and constraints to estimate the robot's poses along with the parameters of the continuous ground manifold.

II. Related works

A. Exteroceptive navigation systems for ground robots

Over the past few decades, localization for the ground robot using exteroceptive sensors has been heavily investigated. Some works fuse the exteroceptive sensors with wheel encoders to get better performance. VIWO [30] utilize the wheel encoder based on the two-wheel vehicle model, where the robot's forward linear speed and angular speed are deduced from the two-wheel encoders. Other works, such as VINS-vehicle [18] and LIO-vehicle [19] use the bicycle model. These methods use the wheel encoder and steering angle sensors measurements to build a 2DOF vehicle dynamics model and then construct an additional pre-integration for the factor graph optimization.

Unlike the aerial or underwater robot which can move freely in 3D space, the ground robot can only travel on the ground. Therefore, some methods utilize the ground manifold as an additional constraint to improve the pose estimation accuracy [17], [27]–[29]. Based on the planar ground assumption, [27] solves the homography matrix to estimate the motion of the ground robot. Meanwhile, [28] propose a one-point RANSAC outlier rejection method that speeds up pose estimation. However, these approaches only focus on incorporating motion constraints in visual data association. Coupling the ground manifold constraints in the optimization problem, [16], [31] adds approximately planar manifold constraints to the system to reduce estimation error. Recently, [17] parameterizes the ground in the form of a quadratic surface and fuse with the camera, IMU, and wheel encoder. Furthermore, [29] estimate the 6D pose of robots and recover the ground manifold only utilizing wheel encoder and camera, without IMU measurements.

However, the above methods rely heavily on the accuracy of the estimations from exteroceptive sensors. When robots encounter degrading scenarios like repetitive or insufficient visual features, these methods are unable to provide robust pose estimation due to the direct integration of biased and noisy measurements from the IMU or wheel encoders.

B. Interoceptive-only navigation systems

With the tremendous improvement of deep learning, researchers nowadays begin to use data-driven approaches to improve the performance of interoceptive-only odometry.

To enhance the accuracy of inertial odometry, LWOI [24], AI-IMU [21], RoNIN [22], and TLIO [23] utilize neural networks to obtain complete pose estimation only using IMU measurements. Specifically, RINS-W [20] trains an LSTM network to detect specific motion profiles, which are then fused into an Iterative Extended Kalman Filtering (IEKF) framework to estimate robot states. AI-IMU [21] uses a convolutional neural network (CNN) to dynamically output observation noise parameters of the invariant extended Kalman filtering framework. RoNIN [22] uses several different neural networks to regress a moving subject's horizontal positions and direction. TLIO [23] learns pseudo-measurements of relative displacement and places them as observations in a tightly coupled statistical cloning extended Kalman filtering.

To better fuse the measurements from the wheel encoder, [25], [26] use deep learning methods to track a car's velocity during autonomous racing. [25] presents an end-to-end GRU network that takes inputs from IMU, torque, wheel encoder, and steering angle. [26] proposes a hybrid model-based KalmanNet that learns the Kalman gain by training a GRU network, which can fuse the interpretability of Kalman filtering and overcome model mismatch. As a complete pose estimation system, LWOI [24] takes measurements from a Fiber Optical Gyroscope (FOG), wheel encoder, and IMU as inputs to train a Gaussian process that corrects dynamical and observation models, which are then fused in an Extended Kalman Filtering (EKF) framework.

These learning-based methods demonstrate the potential of deep neural networks for interoceptive-only odometry. However, these methods do not leverage the ground manifold constraint, which could further improve the accuracy of pose estimation using only interoceptive sensors.

III. Preliminaries

A. Wheel odometer model

In this paper, we consider a two-wheel ground robot model, which means that the forward velocity ${}^{\mathcal{B}}v_x$ and rotational angular velocity ${}^{\mathcal{B}}\omega_x$ of the robot can be derived from speeds of two wheels:

$$ {}^{\mathcal{B}}v_x = \frac{\omega_l * r_l + \omega_r * r_r}{2}, \quad {}^{\mathcal{B}}\omega_x = \frac{\omega_r * r_r - \omega_l * r_l}{w_b}, \quad (1) $$

where ω_l and ω_r are the speeds of the left and right wheels, r_l and r_r are the radii of two wheels, w_b is the wheelbase between two wheels, $\mathcal{B}$ is the body frame centered on the midpoint of two wheels (with a triaxial sequence of Front-Left-Up).

B. IMU kinematic model

IMU measurements include the non-gravitational acceleration $\widetilde{\boldsymbol{a}}$ and gyroscope $\widetilde{\boldsymbol{\omega}}$, which are measured in the IMU frame $\mathcal{I}$ (centered on the IMU sensor with a triaxial sequence of Front-Left-Up) and given by:

$$\begin{aligned} {}^{\mathcal{I}}\widetilde{\boldsymbol{a}} &= {}^{\mathcal{G}}_{\mathcal{I}}\mathbf{R}^{\top}({}^{\mathcal{G}}\boldsymbol{a} + {}^{\mathcal{G}}\mathbf{g}) + {}^{\mathcal{I}}\boldsymbol{b}_{\boldsymbol{a}} + \boldsymbol{n}_{\boldsymbol{a}}, \\ {}^{\mathcal{I}}\widetilde{\boldsymbol{\omega}} &= {}^{\mathcal{I}}\boldsymbol{\omega} + {}^{\mathcal{I}}\boldsymbol{b}_{\boldsymbol{\omega}} + \boldsymbol{n}_{\boldsymbol{\omega}}, \end{aligned} \tag{2}$$

where ${}^{\mathcal{G}}\boldsymbol{a}$ is the true acceleration in the gravity-aligned frame $\mathcal{G}$ (with z-axis pointing up vertically), ${}^{\mathcal{I}}\boldsymbol{\omega}$ is the true angular velocity in $\mathcal{I}$ frame, ${}^{\mathcal{G}}\mathbf{g} = [0, 0, -9.8m/s^2]$ is the gravity vector in $\mathcal{G}$ frame, ${}^{\mathcal{G}}_{\mathcal{I}}\mathbf{R}$ is the rotation matrix from $\mathcal{I}$ frame to $\mathcal{G}$ frame, $\boldsymbol{n}_{\boldsymbol{a}}$ and $\boldsymbol{n}_{\boldsymbol{\omega}}$ are the additive Gaussian white noise in acceleration and gyroscope measurements, $\boldsymbol{b}_{\boldsymbol{a}}$ and $\boldsymbol{b}_{\boldsymbol{\omega}}$ are the biases of IMU modeled as random walks:

$$\begin{aligned} \boldsymbol{n}_{\boldsymbol{a}} &\sim \mathcal{N}(0, \boldsymbol{\Sigma}^2_{\boldsymbol{a}}), \quad \dot{\boldsymbol{b}}_{\boldsymbol{a}} = \boldsymbol{n}_{\boldsymbol{b}_{\boldsymbol{a}}} \sim \mathcal{N}(0, \boldsymbol{\Sigma}^2_{\boldsymbol{b}_{\boldsymbol{a}}}), \\ \boldsymbol{n}_{\boldsymbol{\omega}} &\sim \mathcal{N}(0, \boldsymbol{\Sigma}^2_{\boldsymbol{\omega}}), \quad \dot{\boldsymbol{b}}_{\boldsymbol{\omega}} = \boldsymbol{n}_{\boldsymbol{b}_{\boldsymbol{\omega}}} \sim \mathcal{N}(0, \boldsymbol{\Sigma}^2_{\boldsymbol{b}_{\boldsymbol{\omega}}}), \end{aligned} \tag{3}$$

where $\boldsymbol{\Sigma}^2_*$ is the covariance corresponding to each Gaussian distribution respectively.

C. Yaw independent attitude convention

Inpired by [32], we utilize an attitude convention, which decouples the attitude into yaw and tilt angles. Therefore, the estimations of the velocity and the tilt angle could avoid being affected by the drift of the yaw angle when updating by body velocity. To make better use of the wheel encoder as the measurement of body velocity, the attitude is represented as follows:

$$\begin{aligned} \mathbf{R} &= \mathbf{R}_{\psi}\mathbf{R}_{\phi}, \\ \mathbf{R}_{\psi} &= \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix}, \\ \mathbf{R}_{\phi} &= \begin{pmatrix} \frac{1-s_1^2+s_2^2}{1+s_1^2+s_2^2} & \frac{-2s_1s_2}{1+s_1^2+s_2^2} & \frac{2s_1}{1+s_1^2+s_2^2} \\ \frac{-2s_1s_2}{1+s_1^2+s_2^2} & \frac{1+s_1^2-s_2^2}{1+s_1^2+s_2^2} & \frac{2s_2}{1+s_1^2+s_2^2} \\ \frac{-2s_1}{1+s_1^2+s_2^2} & \frac{-2s_2}{1+s_1^2+s_2^2} & \frac{1-s_1^2-s_2^2}{1+s_1^2+s_2^2} \end{pmatrix}, \end{aligned} \tag{4}$$

where $\mathbf{R}$ is the rotation matrix of the body frame, ψ is the yaw angle, and $\mathbf{s} = (s_1, s_2)^{\top}$ is the tilt vector.

D. Uniform dual cubic B-spline surface

Considering the cubic B-spline manifold with the uniform knot vectors:

$$\begin{aligned} X &= \{x_1, x_2, \cdots, x_t \mid x_{i+1} - x_i = \text{const}, 1 \le i \le t, i \in \mathbb{Z}\}, \\ Y &= \{y_1, y_2, \cdots, y_s \mid y_{j+1} - y_j = \text{const}, 1 \le j \le s, j \in \mathbb{Z}\}. \end{aligned} \tag{5}$$

We represent the B-spline basis functions by the local parameter $u \in [0, 1]$ and $v \in [0, 1]$ instead of the global parameter $x \in [x_i, x_{i+1}]$ and $y \in [y_j, y_{j+1}]$ on each knot interval by performing the transformations as:

$$\begin{aligned} x &= x(u) = (1-u)x_i + ux_{i+1}, u \in [0, 1], \\ y &= y(v) = (1-v)y_j + vy_{j+1}, v \in [0, 1]. \end{aligned} \tag{6}$$

Then, the cubic B-spline manifold $\mathcal{S}(x, y)$ can also be represented in compact matrix form:

$$\begin{aligned} \mathcal{S}(x, y) &= \sum\nolimits_{i=0}^{n}\sum\nolimits_{j=0}^{m} B_{i,3}(x)B_{j,3}(y)C_{i,j}, \\ \Rightarrow \mathcal{S}(u, v) &= \boldsymbol{u}\mathbf{B}\mathbf{C}\mathbf{B}^{\top}\boldsymbol{v}^{\top}, \end{aligned} \tag{7}$$

where $\boldsymbol{u} = [u^3, u^2, u, 1]$, $\boldsymbol{v} = [v^3, v^2, v, 1]$, and

$$\mathbf{B} = \frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}, \mathbf{C} = \begin{bmatrix} c_{0,0} & c_{0,1} & c_{0,2} & c_{0,3} \\ c_{1,0} & c_{1,1} & c_{1,2} & c_{1,3} \\ c_{2,0} & c_{2,1} & c_{2,2} & c_{2,3} \\ c_{3,0} & c_{3,1} & c_{3,2} & c_{3,3} \end{bmatrix}. \tag{8}$$

IV. Kinematic Neural Network Design

A. IMU De-Bias Net

Accurately estimating IMU bias is crucial as multiple integrations from IMU measurements are required to obtain the velocity and position of the robot. Expanding on the promising outcomes of our earlier research [33], we utilize the *IMU De-Bias Net* in this work to learn the kinematic characteristic of IMU for estimating the time-varying bias and the corresponding covariance. Note that the estimation of covariance is an added feature of this work.

The *IMU De-Bias Net* comprises two components, one for the accelerometer and the other for the gyroscope. Both components share the same network architecture, which is a 1D version of ResNet [34] with one residual block and two fully connected layers for output. Respectively, input features are historical raw accelerometer measurements ${}^{\mathcal{I}}\widetilde{\boldsymbol{a}}$ and raw gyroscope measurements ${}^{\mathcal{I}}\widetilde{\boldsymbol{\omega}}$. The outputs are $\{{}^{\mathcal{I}}\widehat{\boldsymbol{b}}_{\boldsymbol{a}}, \boldsymbol{\zeta}_{\boldsymbol{v}_{i,i+n}}\}$ and $\{{}^{\mathcal{I}}\widehat{\boldsymbol{b}}_{\boldsymbol{\omega}}, \boldsymbol{\zeta}_{\boldsymbol{q}_{i,i+n}}\}$, which describe the bias and the covariance. To optimize the *IMU De-Bias Net*, the Negative Log Likelihood (NLL) loss functions of relative velocity and relative rotation (in quaternion) are defined on the following integrated increments:

$$\begin{aligned} \mathcal{L}_{\text{NLL},d\boldsymbol{v}} &= \frac{1}{2N}\sum\nolimits_{i=1}^{N}\log\det(\widehat{\boldsymbol{\Sigma}}^2_{\boldsymbol{v}_{i,i+n}}) \\ &\quad + \frac{1}{2N}\sum\nolimits_{i=1}^{N}\|\boldsymbol{v}_{i,i+n} - \widehat{\boldsymbol{v}}_{i,i+n}\|^2_{\widehat{\boldsymbol{\Sigma}}^2_{\boldsymbol{v}_{i,i+n}}}, \\ \mathcal{L}_{\text{NLL},d\boldsymbol{q}} &= \frac{1}{2N}\sum\nolimits_{i=1}^{N}\log\det(\widehat{\boldsymbol{\Sigma}}^2_{\boldsymbol{q}_{i,i+n}}) \\ &\quad + \frac{1}{2N}\sum\nolimits_{i=1}^{N}\|\text{Log}\big((\widehat{\boldsymbol{q}}_{i,i+n})^* \otimes \boldsymbol{q}_{i,i+n}\big)\|^2_{\widehat{\boldsymbol{\Sigma}}^2_{\boldsymbol{q}_{i,i+n}}}, \end{aligned} \tag{9}$$

where

$$\widehat{\boldsymbol{v}}_{i,i+n} = \int_i^{i+n} {}^{\mathcal{G}}_{\mathcal{I}_t}\mathbf{R}({}^{\mathcal{I}_t}\widetilde{\boldsymbol{a}} - {}^{\mathcal{I}_t}\widehat{\boldsymbol{b}}_{\boldsymbol{a}})dt, \tag{10a}$$

$$\boldsymbol{v}_{i,i+n} = {}^{\mathcal{G}}\boldsymbol{v}_{\mathcal{I}_{i+n}} - {}^{\mathcal{G}}\boldsymbol{v}_{\mathcal{I}_i}, \tag{10b}$$

$$\widehat{\boldsymbol{q}}_{i,i+n} = \int_i^{i+n} \frac{1}{2}{}^{\mathcal{I}_i}_{\mathcal{I}_t}\boldsymbol{q} \otimes ({}^{\mathcal{I}_t}\widetilde{\boldsymbol{\omega}} - {}^{\mathcal{I}_t}\widehat{\boldsymbol{b}}_{\boldsymbol{\omega}})dt, \tag{10c}$$

$$\boldsymbol{q}_{i,i+n} = ({}^{\mathcal{G}}_{\mathcal{I}_i}\boldsymbol{q})^* \otimes {}^{\mathcal{G}}_{\mathcal{I}_{i+n}}\boldsymbol{q}. \tag{10d}$$

Note that, the subscript i and $i+n$ is the abbreviation for time t_i and t_{i+n}, $\widehat{\boldsymbol{v}}_{i,i+n}$ and $\widehat{\boldsymbol{q}}_{i,i+n}$ are the intermediate variables calculated from the estimated bias values ${}^{\mathcal{I}}\widehat{\boldsymbol{b}}_{\boldsymbol{a}}$ and ${}^{\mathcal{I}}\widehat{\boldsymbol{b}}_{\boldsymbol{\omega}}$ of the *IMU De-Bias Net* by the forward Euler method, $\boldsymbol{v}_{i,i+n}$ and $\boldsymbol{q}_{i,i+n}$ are the ground truth velocity increment and relative rotation of each input sequence, n is the window size, $\widehat{\boldsymbol{\Sigma}}^2_{\boldsymbol{v}_{i,i+n}}$ and $\widehat{\boldsymbol{\Sigma}}^2_{\boldsymbol{q}_{i,i+n}}$ are the estimated covariance matrices. We assume that the covariance matrices $\widehat{\boldsymbol{\Sigma}}^2_{\boldsymbol{v}_{i,i+n}}$ and $\widehat{\boldsymbol{\Sigma}}^2_{\boldsymbol{q}_{i,i+n}}$ are respectively parameterized by the output vectors $\boldsymbol{\zeta}_{\boldsymbol{v}_{i,i+n}}$ and $\boldsymbol{\zeta}_{\boldsymbol{q}_{i,i+n}}$ using the following function: $\widehat{\boldsymbol{\Sigma}}^2 = \text{diag}(e^{2\widehat{\zeta}_x}, e^{2\widehat{\zeta}_y}, e^{2\widehat{\zeta}_z})$. N is the batch size, $*$ and $\otimes$ are the conjugate and multiplication of quaternion. In training, ${}^{\mathcal{G}}_{\mathcal{I}}\mathbf{R}$ is the ground truth rotation. The quaternion conjugate, quaternion multiplication, and quaternion logarithm Log operations are defined in [35].

B. Wheel Encoder Net

Commonly, it is assumed that there is no speed in the non-forward direction:

$$ {}^{\mathcal{B}}\widetilde{\boldsymbol{v}}_{\mathcal{B}} = \begin{bmatrix} {}^{\mathcal{B}}\widetilde{v}_{\mathcal{B}_x} & 0 & 0 \end{bmatrix}^\top, \tag{11} $$

where ${}^{\mathcal{B}}\widetilde{v}_{\mathcal{B}_x}$ is the measurement value from wheel encoder. However, the above nonholonomic constraints are not always strictly satisfied due to skidding and vertical bouncing. In addition, the extrinsic parameters $\{{}^{\mathcal{I}}_{\mathcal{B}}\mathbf{R}, {}^{\mathcal{I}}\boldsymbol{t}_{\mathcal{B}}\}$ between the IMU frame ($\mathcal{I}$) and the wheel frame ($\mathcal{B}$) require real-time calibration due to the effect of the vibrations and suspension system in motion. We aim to incorporate both IMU data and wheel encoder measurements into a neural network to handle wheel speed outliers and time-varying extrinsic parameters. The architecture of the *Wheel Encoder Net* is the same as that of the *IMU De-bias Net*. Considering the following velocity transfer equation:

$$ {}^{\mathcal{I}}\boldsymbol{v}_{\mathcal{I}} = {}^{\mathcal{I}}_{\mathcal{B}}\mathbf{R}\ {}^{\mathcal{B}}\boldsymbol{v}_{\mathcal{B}} - {}^{\mathcal{I}}\boldsymbol{\omega} \times {}^{\mathcal{I}}\boldsymbol{t}_{\mathcal{B}} - {}^{\mathcal{I}}\dot{\boldsymbol{t}}_{\mathcal{B}}, \tag{12} $$

We feed historical raw wheel speed ${}^{\mathcal{B}}\widetilde{v}_{\mathcal{B}_x}$ and raw gyroscope measurements ${}^{\mathcal{I}}\widetilde{\boldsymbol{\omega}}$ into the *Wheel Encoder Net* and output the compensation of the ${}^{\mathcal{B}}\widetilde{\boldsymbol{v}}_{\mathcal{B}}$ at the last time stamp in each input sequence. To train the *Wheel Encoder Net*, we define the negative log-likelihood (NLL) loss function of the body velocity as follows:

$$ \begin{aligned} \mathcal{L}_{\text{NLL},\boldsymbol{v}} = &\frac{1}{2N}\sum\nolimits_{i=1}^{N} \log\det(\widehat{\boldsymbol{\Sigma}}^2_{\boldsymbol{v}_i}) \\ &+ \frac{1}{2N}\sum\nolimits_{i=1}^{N} \|{}^{\mathcal{I}}\boldsymbol{v}_{\mathcal{I}i} - {}^{\mathcal{I}}\widehat{\boldsymbol{v}}_{\mathcal{I}i}\|^2_{\widehat{\boldsymbol{\Sigma}}^2_{\boldsymbol{v}_i}}, \end{aligned} \tag{13} $$

where

$$ \begin{aligned} {}^{\mathcal{I}}\widehat{\boldsymbol{v}}_{\mathcal{I}i} &= {}^{\mathcal{I}}_{\mathcal{B}}\mathbf{R}\ {}^{\mathcal{B}}\widetilde{\boldsymbol{v}}_{\mathcal{B}i} + {}^{\mathcal{I}}\delta\widehat{\boldsymbol{v}}_{\mathcal{I}i}, \\ \widehat{\boldsymbol{\Sigma}}^2_{\boldsymbol{v}_i} &= \text{diag}(e^{2\widehat{\boldsymbol{\zeta}}_{v_{i,x}}}, e^{2\widehat{\boldsymbol{\zeta}}_{v_{i,y}}}, e^{2\widehat{\boldsymbol{\zeta}}_{v_{i,z}}}) \end{aligned} \tag{14} $$

and $\{{}^{\mathcal{I}}\delta\widehat{\boldsymbol{v}}_{\mathcal{I}i}, \widehat{\boldsymbol{\zeta}}_{v_i}\}$ are the outputs of *Wheel Encoder Net*, ${}^{\mathcal{I}}\boldsymbol{v}_{\mathcal{I}}$ is the ground truth velocity in the $\mathcal{I}$ frame.

V. Ground Manifold Constraint

A. Dual cubic B-spline manifold

Assuming the existence of a ground manifold that can be represented as a dual cubic B-spline surface, any three-dimensional (3D) point $\boldsymbol{p} = (x, y, z)$ on the manifold satisfies the following constraint equation:

$$ z = \sum\nolimits_{i=0}^{n}\sum\nolimits_{j=0}^{m} B_{i,3}(x)B_{j,3}(y)c_{i,j}. \tag{15} $$

Denoting the manifold as $\mathcal{M}$, we can express the constraint equation as:

$$ \mathcal{M}(\boldsymbol{p}) = \sum\nolimits_{i=0}^{n}\sum\nolimits_{j=0}^{m} B_{i,3}(x)B_{j,3}(y)c_{i,j} - z = 0. \tag{16} $$

By using the local representation in Eq. (7), we can express Eq. (16) in terms of the local coordinates (u, v), where each piece of the manifold is represented as follows:

$$ \begin{aligned} \mathcal{M}(\boldsymbol{p}) &= \sum\nolimits_{i=0}^{3}\sum\nolimits_{j=0}^{3} B_{i,3}(u)B_{j,3}(v)c_{i,j} - z = 0 \\ &= \boldsymbol{u}\mathbf{B}\mathbf{C}\mathbf{B}^\top\boldsymbol{v}^\top - z = 0, \end{aligned} \tag{17} $$

where

$$ \begin{aligned} u &= \frac{x - x_i}{x_{i+1} - x_i}, x \in [x_i, x_{i+1}], \\ v &= \frac{y - y_j}{y_{j+1} - y_j}, y \in [y_j, y_{j+1}]. \end{aligned} \tag{18} $$

Assuming that the uniform knot vector of the dual cubic B-spline manifold has a constant interval of $d = x_{i+1} - x_i = y_{j+1} - y_j$, we can express the transformation as follows:

$$ \begin{aligned} u &= k_x x + b_x \in [0,1],\ k_x = 1/d,\ b_x = -g_x/d, \\ v &= k_y y + b_y \in [0,1],\ k_y = 1/d,\ b_y = -g_y/d, \end{aligned} \tag{19} $$

where $(g_x, g_y) = (x_i, y_j)$ is the global coordinate of the ground manifold. Therefore, the ground manifold in Eq. (17) can be abbreviated as follows:

$$ \begin{aligned} \mathcal{M}(\boldsymbol{p}) &= \boldsymbol{u}\mathbf{B}\mathbf{C}\mathbf{B}^\top\boldsymbol{v}^\top - z = 0, \\ &= \boldsymbol{x}\mathbf{K}_x\mathbf{B}\mathbf{C}\mathbf{B}^\top\mathbf{K}_y^\top\boldsymbol{y}^\top - z = 0, \end{aligned} \tag{20} $$

where

$$ \begin{aligned} \boldsymbol{x} = \begin{bmatrix} x^3, x^2, x, 1 \end{bmatrix}, \mathbf{K}_x = \begin{bmatrix} k_x^3 & 0 & 0 & 0 \\ 3k_x^2 b_x & k_x^2 & 0 & 0 \\ 3k_x b_x^2 & 2k_x b_x & k_x & 0 \\ b_x^3 & b_x^2 & b_x & 1 \end{bmatrix}, \\ \boldsymbol{y} = \begin{bmatrix} y^3, y^2, y, 1 \end{bmatrix}, \mathbf{K}_y = \begin{bmatrix} k_y^3 & 0 & 0 & 0 \\ 3k_y^2 b_y & k_y^2 & 0 & 0 \\ 3k_y b_y^2 & 2k_y b_y & k_y & 0 \\ b_y^3 & b_y^2 & b_y & 1 \end{bmatrix}. \end{aligned} \tag{21} $$

B. Ground manifold constraint

Furthermore, given that the robot travels on the ground that is parameterized as a dual cubic B-spline manifold as Eq. (20), which indicates that the poses of the tangent points between the wheels and the ground satisfy two kinds of constraints, one is that the position where the wheel attached to the ground ${}^{\mathcal{G}}\boldsymbol{p}_w$ is on the ground manifold, and the other is that the orientation ${}^{\mathcal{G}}_{w}\mathbf{R}$ is parallel to the normal of the surface manifold:

$$ \begin{aligned} &\mathcal{M}\left({}^{\mathcal{G}}\boldsymbol{p}_w\right) = 0, \\ &{}^{\mathcal{G}}_{w}\mathbf{R} \cdot \boldsymbol{e}_3 \times \nabla\mathcal{M}\left({}^{\mathcal{G}}\boldsymbol{p}_w\right) = \mathbf{0}, \end{aligned} \tag{22} $$

where $\boldsymbol{e}_3 = [0, 0, 1]^\top$ is the third axis, $\nabla = \frac{\partial}{\partial x}\vec{i} + \frac{\partial}{\partial y}\vec{j} + \frac{\partial}{\partial z}\vec{k}$ is a vector differential operator, and $\mathbf{0}$ is the zero vector. The

pose of any wheel tangent point $\{{}^{\mathcal{G}}_{w}\mathbf{R}, {}^{\mathcal{G}}\boldsymbol{p}_w\}$ can be converted by extrinsic transformation $\{{}^{w}_{\mathcal{I}}\mathbf{R}, {}^{\mathcal{I}}\boldsymbol{t}_w\}$ from the IMU pose $\{{}^{\mathcal{G}}_{\mathcal{I}}\mathbf{R}, {}^{\mathcal{G}}\boldsymbol{p}_{\mathcal{I}}\}$:

$$\begin{aligned} {}^{\mathcal{G}}\boldsymbol{p}_w &= {}^{\mathcal{G}}\boldsymbol{p}_{\mathcal{I}} + {}^{\mathcal{G}}_{\mathcal{I}}\mathbf{R}\ {}^{\mathcal{I}}\boldsymbol{t}_w, \\ {}^{\mathcal{G}}_{w}\mathbf{R} &= {}^{\mathcal{G}}_{\mathcal{I}}\mathbf{R}\ {}^{\mathcal{I}}_{w}\mathbf{R}. \end{aligned} \tag{23}$$

Assuming that the $\mathcal{I}$ frame and the w frame have the same rotation representation, we can substitute Eq. (4), Eq. (20) and Eq. (23) into equation Eq. (22). After simplification, we obtain:

$$\begin{aligned} &\boldsymbol{x}_w\mathbf{K}_x\mathbf{B}\mathbf{C}\mathbf{B}^\top\mathbf{K}_y^\top\boldsymbol{y} - z_w = 0, \\ &\boldsymbol{x}_w\mathbf{K}_x\mathbf{B}\mathbf{C}\mathbf{B}^\top\mathbf{K}_y^\top\boldsymbol{\partial y}_w + 2\frac{s_1\sin(\psi)+s_2\cos(\psi)}{1-s_1^2-s_2^2} = 0, \\ &\boldsymbol{\partial x}_w\mathbf{K}_x\mathbf{B}\mathbf{C}\mathbf{B}^\top\mathbf{K}_y^\top\boldsymbol{y}_w + 2\frac{s_1\cos(\psi)-s_2\sin(\psi)}{1-s_1^2-s_2^2} = 0, \end{aligned} \tag{24}$$

where

$$\begin{aligned} \boldsymbol{x}_w &= \left[x_w^3, x_w^2, x_w, 1\right], \boldsymbol{\partial x}_w = \left[3x_w^2, 2x_w, 1, 0\right], \\ \boldsymbol{y_w} &= \left[y_w^3, y_w^2, y_w, 1\right], \ \boldsymbol{\partial y_w} = \left[3y_w^2, 2y_w, 1, 0\right], \end{aligned} \tag{25}$$

${}^{\mathcal{G}}\boldsymbol{p}_w = [x_w, y_w, z_w]^\top$ is calculated by Eq. (23), $\mathbf{K}_x$, $\mathbf{K}_y$, $\mathbf{B}$ and $\mathbf{C}$ have the same definition as Eq. (21), $\{s_1, s_2, \psi\}$ are the abbreviations of IMU rotation components $\{{}^{\mathcal{G}}_{\mathcal{I}}s_1, {}^{\mathcal{G}}_{\mathcal{I}}s_2, {}^{\mathcal{G}}_{\mathcal{I}}\psi\}$. Furthermore, we can transfer Eq. (24) to a vector form w.r.t. the matrix $\mathbf{C}$:

$$\begin{aligned} &(\boldsymbol{y}_w\mathbf{K}_y\mathbf{B}\boxtimes\boldsymbol{x}_w\mathbf{K}_x\mathbf{B})Vec(\mathbf{C}) - z = 0, \\ &(\boldsymbol{\partial y}_w\mathbf{K}_y\mathbf{B}\boxtimes\boldsymbol{x}_w\mathbf{K}_x\mathbf{B})Vec(\mathbf{C}) + 2\frac{s_1\sin(\psi)+s_2\cos(\psi)}{1-s_1^2-s_2^2} = 0, \\ &(\boldsymbol{y}_w\mathbf{K}_y\mathbf{B}\boxtimes\boldsymbol{\partial x}_w\mathbf{K}_x\mathbf{B})Vec(\mathbf{C}) + 2\frac{s_1\cos(\psi)-s_2\sin(\psi)}{1-s_1^2-s_2^2} = 0, \end{aligned} \tag{26}$$

where $\boxtimes$ is the Kronecker product, and $Vec(\mathbf{C})$ is the column straightening function that transforms a matrix into a vector.

VI. Space-based sliding window fusion

To better estimate the poses of a ground robot, we apply a space-based sliding window framework that combines manifold constraints with neural-processed IMU and wheel odometer data. This approach is specifically designed to not only smooth poses within a sliding window but also recover the parameters of the dual cubic B-spline manifold.

A. States

The state vector $\mathcal{X}$ includes IMU state $\mathcal{X}_I$, space-based sliding window state $\mathcal{X}_S$ and space-based sliding control mesh $\boldsymbol{c}$, which is represented as follows:

$$\begin{aligned} \mathcal{X} &= \begin{bmatrix}\mathcal{X}_I{}^\top & \mathcal{X}_S{}^\top & \boldsymbol{c}^\top\end{bmatrix}^\top, \\ \mathcal{X}_I &= \begin{bmatrix}{}^{\mathcal{G}}\boldsymbol{p}_{\mathcal{I}}{}^\top & {}^{\mathcal{I}}\boldsymbol{v}_{\mathcal{I}}{}^\top & {}^{\mathcal{G}}_{\mathcal{I}}\psi & {}^{\mathcal{G}}_{\mathcal{I}}\boldsymbol{s}^\top\end{bmatrix}^\top, \\ \mathcal{X}_S &= \begin{bmatrix}\boldsymbol{\xi}_1 & \boldsymbol{\xi}_2 & \cdots & \boldsymbol{\xi}_n\end{bmatrix}^\top, \\ \boldsymbol{\xi}_j &= \begin{bmatrix}{}^{\mathcal{G}}\boldsymbol{p}_{\mathcal{I}_j}^\top & {}^{\mathcal{G}}_{\mathcal{I}_j}\psi & {}^{\mathcal{G}}_{\mathcal{I}_j}\boldsymbol{s}^\top\end{bmatrix}^\top, \\ \boldsymbol{c} &= Vec(\mathbf{C}), \end{aligned} \tag{27}$$

where ${}^{\mathcal{G}}\boldsymbol{p}_{\mathcal{I}}$, ${}^{\mathcal{G}}_{\mathcal{I}}\psi$ and ${}^{\mathcal{G}}_{\mathcal{I}}\boldsymbol{s}$ are the position, yaw angle, and tilt vector of the ground robot in the $\mathcal{G}$ frame, ${}^{\mathcal{I}}\boldsymbol{v}_{\mathcal{I}}$ is the velocity of the robot in $\mathcal{I}$ frame, $\boldsymbol{\xi}_j$ is the jth state of the space-based sliding window, and $\boldsymbol{c}$ is the control mesh vector obtained by column straightening of the 4×4 control mesh matrix $\mathbf{C}$.

This work utilizes a uniform dual cubic B-spline manifold to characterize the ground on which the vehicle travels. Therefore, a control mesh with a fixed interval (parameter d) horizontally is shown as all circles in Fig. 2. In practice, we only optimize the states associated with the mesh where the vehicle is currently traveling. The trajectories ($\mathcal{X}_S$, yellow lines) and control mesh points ($\mathbf{C}$, 4×4 yellow circles) in Fig. 2 represent the currently optimized states.

B. Process model

In this work, we leverage the neural-processed IMU measurements to drive the state estimation system, whose full process model is given as follows:

$$\begin{aligned} {}^{\mathcal{G}}\dot{\boldsymbol{p}}_{\mathcal{I}} &= {}^{\mathcal{G}}_{\mathcal{I}}\mathbf{R}\,{}^{\mathcal{I}}\boldsymbol{v}_{\mathcal{I}}, \\ {}^{\mathcal{I}}\dot{\boldsymbol{v}}_{\mathcal{I}} &= {}^{\mathcal{I}}\boldsymbol{a} - {}^{\mathcal{G}}_{\mathcal{I}}\mathbf{R}^\top{}^{\mathcal{G}}\mathbf{g} - {}^{\mathcal{I}}\boldsymbol{\omega}\times{}^{\mathcal{I}}\boldsymbol{v}_{\mathcal{I}}, \\ {}^{\mathcal{G}}_{\mathcal{I}}\dot{\psi} &= \begin{bmatrix}-s_1 & -s_2 & 1\end{bmatrix}{}^{\mathcal{I}}\boldsymbol{\omega}, \\ {}^{\mathcal{G}}_{\mathcal{I}}\dot{\boldsymbol{s}} &= \frac{1}{2}\begin{bmatrix}-2s_1s_2 & s_1^2-s_2^2+1 & 2s_2 \\ s_1^2-s_2^2-1 & 2s_1s_2 & -2s_1\end{bmatrix}{}^{\mathcal{I}}\boldsymbol{\omega}, \end{aligned} \tag{28}$$

where the rotation ${}^{\mathcal{G}}_{\mathcal{I}}\mathbf{R} = \mathbf{R}_\psi\mathbf{R}_\phi$ is expressed as a yaw-independent form described in Sec. III-C, and

$$\begin{aligned} {}^{\mathcal{I}}\boldsymbol{\omega} &= {}^{\mathcal{I}}\widetilde{\boldsymbol{\omega}} - {}^{\mathcal{I}}\widehat{\boldsymbol{b}}_{\boldsymbol{\omega}} - \boldsymbol{n}_{\boldsymbol{\omega}}, \quad \boldsymbol{n}_{\boldsymbol{\omega}} \sim \mathcal{N}(0, \widehat{\boldsymbol{\Sigma}}^2_{\boldsymbol{\omega}}), \\ {}^{\mathcal{I}}\boldsymbol{a} &= {}^{\mathcal{I}}\widetilde{\boldsymbol{a}} - {}^{\mathcal{I}}\widehat{\boldsymbol{b}}_{\boldsymbol{a}} - \boldsymbol{n}_{\boldsymbol{a}}, \quad \boldsymbol{n}_{\boldsymbol{a}} \sim \mathcal{N}(0, \widehat{\boldsymbol{\Sigma}}^2_{\boldsymbol{a}}). \end{aligned} \tag{29}$$

In order to improve the efficiency of tuning the covariance of IMU measurements ${}^{\mathcal{I}}\widehat{\boldsymbol{b}}_{\boldsymbol{\omega}}$ and ${}^{\mathcal{I}}\widehat{\boldsymbol{b}}_{\boldsymbol{a}}$ during sensor fusion and to exploit the consistency of the IMU bias and covariance of the network output, we can use the outputs of *IMU De-Bias Net* to derive the IMU covariance as follows:

$$\widehat{\boldsymbol{\Sigma}}^2_{\boldsymbol{a}} = \frac{1}{n\Delta t^2}\widehat{\boldsymbol{\Sigma}}^2_{\boldsymbol{v}_{i,i+n}}, \quad \widehat{\boldsymbol{\Sigma}}^2_{\boldsymbol{\omega}} = \frac{1}{n\Delta t^2}\widehat{\boldsymbol{\Sigma}}^2_{\boldsymbol{q}_{i,i+n}}. \tag{30}$$

The details of formula derivation can be found in Appendix. A. Therefore, the differential equations can be discretized as follows:

$$\begin{aligned} \mathcal{X}_{\mathcal{I}_{k+1}} &= \mathbf{F}(\mathcal{X}_{\mathcal{I}_k}, \boldsymbol{u}_k) \\ &= \mathbf{F}_{\mathcal{I}}\mathcal{X}_{\mathcal{I}_k} + \mathbf{F}_{\boldsymbol{n}}\boldsymbol{n}_k, \end{aligned} \tag{31}$$

where

$$\mathcal{X}_{\mathcal{I}_k} = \begin{bmatrix}{}^{\mathcal{G}}\boldsymbol{p}_{\mathcal{I}_k} & {}^{\mathcal{I}_k}\boldsymbol{v} & {}^{\mathcal{G}}_{\mathcal{I}_k}\psi & {}^{\mathcal{G}}_{\mathcal{I}_k}\boldsymbol{s}\end{bmatrix}^\top, \tag{32}$$

$$\boldsymbol{u}_k = \begin{bmatrix}{}^{\mathcal{I}_k}\widehat{\boldsymbol{\omega}} - \boldsymbol{n}_{\boldsymbol{\omega}} \\ {}^{\mathcal{I}_k}\widehat{\boldsymbol{a}} - \boldsymbol{n}_{\boldsymbol{a}}\end{bmatrix} = \begin{bmatrix}{}^{\mathcal{I}}\widetilde{\boldsymbol{\omega}} - {}^{\mathcal{I}}\widehat{\boldsymbol{b}}_{\boldsymbol{\omega}} - \boldsymbol{n}_{\boldsymbol{\omega}} \\ {}^{\mathcal{I}}\widetilde{\boldsymbol{a}} - {}^{\mathcal{I}}\widehat{\boldsymbol{b}}_{\boldsymbol{a}} - \boldsymbol{n}_{\boldsymbol{a}}\end{bmatrix}. \tag{33}$$

The details about $\mathbf{F}_{\mathcal{I}}$ and $\mathbf{F}_{\boldsymbol{n}}$ can be found in Appendix. B.

C. Measurement model

There are two kinds of measurements in the proposed system, one for the neural velocity measurement and the other for the manifold constraint.

1) Neural velocity measurement: We could obtain tri-axial pseudo-measurements of the IMU's velocity through the *Wheel Encoder Net* in Sec. IV-B:

$$\boldsymbol{h}_{\boldsymbol{v}}(\mathcal{X}) = {}^{\mathcal{I}}\boldsymbol{v}_{\mathcal{I}} = {}^{\mathcal{I}}\widehat{\boldsymbol{v}}_{\mathcal{I}} + \boldsymbol{n}_{\boldsymbol{v}}, \tag{34}$$

where $\boldsymbol{n_v} \sim \mathcal{N}(0, \widehat{\boldsymbol{\Sigma}}_{\boldsymbol{v}}^2)$, ${}^{\mathcal{I}}\widehat{\boldsymbol{v}}_{\mathcal{I}}$ and $\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{v}}^2$ can be obtained from the *Wheel Encoder Net*.

2) Manifold constraint: The IMU motion system is often affected by noise, which can result in inaccurate state estimation. To address this issue, we use the dual cubic B-spline manifold constraints, as stated in Eq. (26), to improve performance. As shown in the left subfigure of Fig. 2, we divide the globally continuous B-spline ground surface into 7×7 manifolds centered around the ground robot. The 7×7 manifolds have two parts.

The first part represents the current manifold, which corresponds to the most intermediate ground (active ground manifold). In the active ground manifold, we can get the constraint equation:

$$\begin{aligned}
&\boldsymbol{h}_{g_x,g_y}(\mathcal{X}) = \begin{bmatrix} \boldsymbol{h}_{g_x,g_y,\boldsymbol{\xi}_1}(\mathcal{X}) \\ \vdots \\ \boldsymbol{h}_{g_x,g_y,\boldsymbol{\xi}_{n_{g_x,g_y}}}(\mathcal{X}) \end{bmatrix}, \forall\, \boldsymbol{\xi}_j \in \mathcal{X}_S \in \mathcal{M}_{g_x,g_y}, \\
&\boldsymbol{h}_{g_x,g_y,\boldsymbol{\xi}}(\mathcal{X}) = \mathbf{0}_{3\times 1} = \boldsymbol{n}_{\boldsymbol{\mathcal{M}}} + \\
&\begin{bmatrix} (\boldsymbol{y}_w\mathbf{K}_y\mathbf{B} \boxtimes \boldsymbol{x}_w\mathbf{K}_x\mathbf{B})Vec(\mathbf{C}_{g_x,g_y}) - z_w \\ (\boldsymbol{\partial y}_w\mathbf{K}_y\mathbf{B} \boxtimes \boldsymbol{x}_w\mathbf{K}_x\mathbf{B})Vec(\mathbf{C}_{g_x,g_y}) + 2\frac{s_1\sin(\psi)+s_2\cos(\psi)}{1-s_1^2-s_2^2} \\ (\boldsymbol{y}_w\mathbf{K}_y\mathbf{B} \boxtimes \boldsymbol{\partial x}_w\mathbf{K}_x\mathbf{B})Vec(\mathbf{C}_{g_x,g_y}) + 2\frac{s_1\cos(\psi)-s_2\sin(\psi)}{1-s_1^2-s_2^2} \end{bmatrix},
\end{aligned} \tag{35}$$

where $\boldsymbol{n}_{\boldsymbol{\mathcal{M}}} \sim \mathcal{N}(0, \boldsymbol{\Sigma}_{\boldsymbol{\mathcal{M}}}^2)$ and $\mathbf{C}_{g_x,g_y}$ is the control mesh contains active B-Spline control points.

The other part represents the other 48 manifolds (static ground manifolds) except the most intermediate manifold (active ground manifold). In these manifolds, we can obtain:

$$\begin{aligned}
&\boldsymbol{h}_{u,v}(\mathcal{X}) = \begin{bmatrix} \boldsymbol{h}_{u,v,\boldsymbol{\xi}_1}(\mathcal{X}) \\ \vdots \\ \boldsymbol{h}_{u,v,\boldsymbol{\xi}_{n_{u,v}}}(\mathcal{X}) \end{bmatrix}, \forall\, \boldsymbol{\xi}_j \in \mathcal{M}_{u\neq g_x, v\neq g_y}, \\
&\boldsymbol{h}_{u,v,\boldsymbol{\xi}}(\mathcal{X}) = \mathbf{0}_{3\times 1} = \boldsymbol{n}_{\boldsymbol{\mathcal{M}}} + \\
&\begin{bmatrix} (\boldsymbol{y}_w\mathbf{K}_y\mathbf{B} \boxtimes \boldsymbol{x}_w\mathbf{K}_x\mathbf{B})Vec(\mathbf{C}_{u,v}) - z_w \\ (\boldsymbol{\partial y}_w\mathbf{K}_y\mathbf{B} \boxtimes \boldsymbol{x}_w\mathbf{K}_x\mathbf{B})Vec(\mathbf{C}_{u,v}) + 2\frac{s_1\sin(\psi)+s_2\cos(\psi)}{1-s_1^2-s_2^2} \\ (\boldsymbol{y}_w\mathbf{K}_y\mathbf{B} \boxtimes \boldsymbol{\partial x}_w\mathbf{K}_x\mathbf{B})Vec(\mathbf{C}_{u,v}) + 2\frac{s_1\cos(\psi)-s_2\sin(\psi)}{1-s_1^2-s_2^2} \end{bmatrix}, \\
&Vec(\mathbf{C}_{u,v}) = \\
&\mathbf{M}_{S1}(u,v)Vec(\mathbf{C}_{g_x,g_y}) + \mathbf{M}_{S2}(u,v)Vec(\mathbf{C}_{u\neq g_x,v\neq g_y}),
\end{aligned} \tag{36}$$

where $Vec(\mathbf{C}_{u,v})$ is a vector that can be separated into the control points that will be optimized ($Vec(\mathbf{C}_{g_x,g_y})$) and those that will not be optimized ($Vec(\mathbf{C}_{u\neq g_x,v\neq g_y})$). This separation is achieved through two selection matrices, $\mathbf{M}_{S1}(u,v)$ and $\mathbf{M}_{S2}(u,v)$.

Note that all vectors and matrices in Eq. (35) and Eq. (36) are defined in the same way as in Sec. V. Among the 7×7 manifolds shown in the left subfigure of Fig. 2, only the current sliding window poses (the active state $\mathcal{X}_S = \{\boldsymbol{\xi} | \boldsymbol{\xi} \in \mathcal{M}g_x, g_y\}$, which are located in the active ground manifold $\mathcal{M}_{g_x,g_y}$) and the current control mesh (the active control points in $\mathbf{C} = \mathbf{C}_{g_x,g_y}$, which shape the active ground manifold $\mathcal{M}_{g_x,g_y}$) are estimated in one update process. The poses on the other 48 manifolds (the static ground manifolds $\{\mathcal{M}_{u,v} \mid u \neq g_x, v \neq g_y\}$) and the other 64 control points (the static control points) are not optimized but utilized to construct complete constraints of all active states for better optimization.

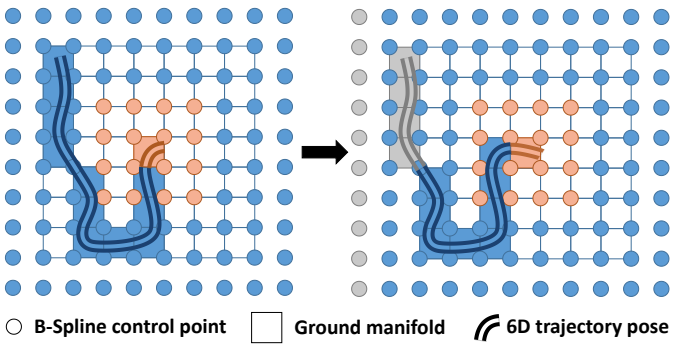

Fig. 2. The demonstration of space-based sliding window update strategy. Yellow, blue and gray represent the active, the static and the marginalized states or information in the proposed system, respectively.

By combining the constraints in each manifold, we can obtain the observation equation with Gaussian white noise:

$$\begin{aligned}
&\boldsymbol{h}_{\boldsymbol{\mathcal{M}}}(\mathcal{X}) = \\
&Vec\begin{pmatrix} \boldsymbol{h}_{g_x-3,g_y-3}(\mathcal{X}) \cdots \boldsymbol{h}_{g_x-3,g_y}(\mathcal{X}) \cdots \boldsymbol{h}_{g_x-3,g_y+3}(\mathcal{X}) \\ \vdots \quad \ddots \quad \vdots \quad \ddots \quad \vdots \\ \boldsymbol{h}_{g_x,g_y-3}(\mathcal{X}) \quad \cdots \quad \boldsymbol{h}_{g_x,g_y}(\mathcal{X}) \quad \cdots \quad \boldsymbol{h}_{g_x,g_y+3}(\mathcal{X}) \\ \vdots \quad \ddots \quad \vdots \quad \ddots \quad \vdots \\ \boldsymbol{h}_{g_x+3,g_y-3}(\mathcal{X}) \cdots \boldsymbol{h}_{g_x+3,g_y}(\mathcal{X}) \cdots \boldsymbol{h}_{g_x+3,g_y+3}(\mathcal{X}) \end{pmatrix},
\end{aligned} \tag{37}$$

where each $\boldsymbol{h}_{u,v}(\mathcal{X})$ represents $3 \times n_{u,v}$ equations in one manifold, where $u \in \{g_x, g_x \pm 1, g_x \pm 2, g_x \pm 3\}$, $v \in \{g_y, g_y \pm 1, g_y \pm 2, g_y \pm 3\}$, and $n_{u,v}$ is the number of states $\boldsymbol{\xi}_j$ in each manifold. The corresponding Jacobian matrix of Eq. (37) can be found in Appendix. C.

TABLE I
STATES INITIALIZATION

	${}^{\mathcal{G}}\boldsymbol{p}_{\mathcal{I}}$	${}^{\mathcal{I}}\boldsymbol{v}_{\mathcal{I}}$	${}^{\mathcal{G}}_{\mathcal{I}}\psi$	${}^{\mathcal{G}}_{\mathcal{I}}\boldsymbol{s}$	$\boldsymbol{c}$	d	$\boldsymbol{n}_{\boldsymbol{\mathcal{M}}}$
value	${}^{\mathcal{G}}\boldsymbol{p}_{\mathcal{I}_0}$	${}^{\mathcal{I}_0}\boldsymbol{v}_{\mathcal{I}_0}$	${}^{\mathcal{G}}_{\mathcal{I}_0}\psi$	${}^{\mathcal{G}}_{\mathcal{I}_0}\boldsymbol{s}$	LSM	5	\
diag(cov)	1e−8	1e−8	1e−8	1e−8	1e−2	\	1e1

D. Extended Kalman filtering process

1) Initialization: At the time of initialization, there is no prior information about the ground manifold, so the space-based sliding window state $\mathcal{X}_S$ and the space-based sliding control vector $\boldsymbol{c}$ do not exist in the initial state vector. The proposed filter framework initializes the IMU state $\mathcal{X}_I$ as the ground truth, and the covariance of this state as well as other parameters are initialized as shown in TABLE I. When the robot enters the second ground manifold (before manifold constraint update), the control vector $\boldsymbol{c}$ is solved by the least squares method as follows:

$$\begin{aligned}
&\boldsymbol{c} = (\mathbf{A}^\top\mathbf{A})^{-1}\mathbf{A}^\top\boldsymbol{b}, \quad \forall\, \boldsymbol{\xi}_j \in \mathcal{X}_S \in \mathcal{M}_{g_x,g_y}, \\
&\mathbf{A} = \left[\mathbf{A}_{\xi_1}, \mathbf{A}_{\xi_2}, \ldots, \mathbf{A}_{\xi_n}\right]^\top, \quad \boldsymbol{b} = \left[\boldsymbol{b}_{\boldsymbol{\xi}_1}, \boldsymbol{b}_{\boldsymbol{\xi}_2}, \ldots \boldsymbol{b}_{\boldsymbol{\xi}_n}\right]^\top, \\
&\mathbf{A}_\xi = \begin{bmatrix} \boldsymbol{y}_w\mathbf{K}_y\mathbf{B} \otimes \boldsymbol{x}_w\mathbf{K}_x\mathbf{B} \\ \boldsymbol{\partial y}_w\mathbf{K}_y\mathbf{B} \otimes \boldsymbol{x}_w\mathbf{K}_x\mathbf{B} \\ \boldsymbol{y}_w\mathbf{K}_y\mathbf{B} \otimes \boldsymbol{\partial x}_w\mathbf{K}_x\mathbf{B} \end{bmatrix}, \boldsymbol{b}_{\boldsymbol{\xi}} = \begin{bmatrix} z_w \\ -2\frac{s_1\sin(\psi)+s_2\cos(\psi)}{1-s_1^2-s_2^2} \\ 2\frac{s_1\cos(\psi)-s_2\sin(\psi)}{1-s_1^2-s_2^2} \end{bmatrix}
\end{aligned} \tag{38}$$

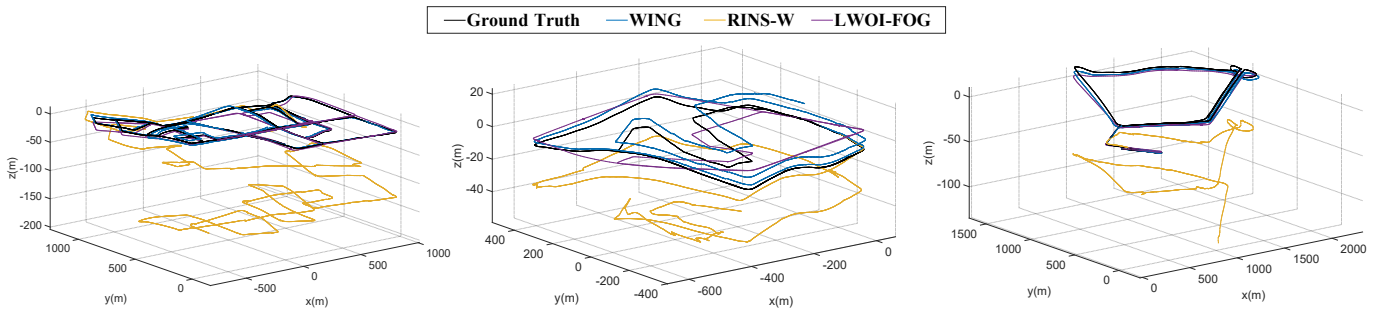

Fig. 3. Demonstration of different algorithms for pose estimation. From left to right, there are 3D estimated trajectories of *urban09*, *urban15*, and *urban17* of the KAIST Urban Dataset [15].

2) State propagation: We can obtain the discrete state propagation equations from the derivations in the Sec. VI-B:

$$\begin{aligned}\mathcal{X}_{\mathcal{I}_{k+1}} &= \mathbf{F}(\mathcal{X}_{\mathcal{I}_k}, \boldsymbol{u}_k),\\ \mathbf{P}_{k+1} &= \mathbf{A}_k\mathbf{P}_k\mathbf{A}_k^\top + \mathbf{B}_k\mathbf{W}\mathbf{B}_k^\top,\\ \mathbf{A}_k = \begin{bmatrix}\mathbf{F}_\mathcal{I} & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{bmatrix},\quad \mathbf{B}_k &= \begin{bmatrix}\mathbf{F}_{\boldsymbol{n}}\\ \mathbf{0}\end{bmatrix},\quad \mathbf{W} = \begin{bmatrix}\boldsymbol{\Sigma}_{\boldsymbol{a}}^2 & \mathbf{0}\\ \mathbf{0} & \boldsymbol{\Sigma}_{\boldsymbol{\omega}}^2\end{bmatrix},\end{aligned} \tag{39}$$

3) State augmentation: In the proposed system, the dimension of the sliding window state $\mathcal{X}_S$ grows as the ground robot runs longer than a fixed distance. During an augmentation step, the state dimension is incremented with cloning. After that, the dimension of the sliding window state $\mathcal{X}_S$ increases by 6.

$$\begin{aligned}\mathbf{P}_{k+1} &= \bar{\mathbf{A}}_k\mathbf{P}_k\bar{\mathbf{A}}_k^\top + \bar{\mathbf{B}}_k\mathbf{W}\bar{\mathbf{B}}_k^\top,\\ \bar{\mathbf{A}}_k = \begin{bmatrix}\mathbf{F}_\mathcal{I} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{I} & \mathbf{0}\\ \mathbf{F}_\mathcal{I} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{0} & \mathbf{I}\end{bmatrix},\quad \bar{\mathbf{B}}_k &= \begin{bmatrix}\mathbf{F}_{\boldsymbol{n}}\\ \mathbf{0}\\ \mathbf{F}_{\boldsymbol{n}}\\ \mathbf{0}\end{bmatrix},\end{aligned} \tag{40}$$

4) Measurement update: There are two measurement updates in the proposed EKF. One is a network-corrected velocity in the IMU frame ($\mathcal{I}$) with a frequency of $20Hz$. And the other is a pseudo-measurement of the ground manifold, which is used to update the state when the ground robot runs beyond the boundary of one ground manifold. We use the conventional extended Kalman filtering process for measurement updates, and the two measurement models in this framework are written uniformly as follows:

$$\begin{aligned}\mathbf{K} &= \mathbf{P}\mathbf{H}^\top\left(\mathbf{H}\mathbf{P}\mathbf{H}^\top + \mathbf{R}\right)^{-1},\\ \mathcal{X} &\leftarrow \mathcal{X} + \mathbf{K}\big(\boldsymbol{z} - \boldsymbol{h}(\mathcal{X})\big),\\ \mathbf{P} &\leftarrow (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}(\mathbf{I} - \mathbf{K}\mathbf{H})^\top + \mathbf{K}\mathbf{R}\mathbf{K}^\top,\end{aligned} \tag{41}$$

where $\mathbf{H}$, $\mathbf{R}$, $\boldsymbol{h}(\mathcal{X})$ and $\boldsymbol{z}$ denote the Jacobian matrix, measurement covariance, measurement model and measurement value provided by Sec. VI-C, respectively.

5) Sliding and marginalization: In Fig. 2, when the ground robot travels to the next ground manifold, the trajectory poses that locate in the previous ground and the control points that no longer shape the previous ground manifold are both removed from the active state and added to the static state. As mentioned earlier, the poses and control points in the static state are used to optimize the manifold that the robot currently stays on, which is shaped by the active control mesh. Only when the static state cannot affect the current 7×7 manifold that is centered around the robot, will it be marginalized, that is, dropped into the marginalized state as shown in the right subfigure of Fig. 2.

VII. Experiments

To better evaluate the proposed method, we compare estimation accuracy with other approaches and perform ablation studies to demonstrate the performance of each module in the KAIST Urban Dataset [15]. We randomly choose *urban07*, *urban09*, *urban11*, *urban13*, *urban15* and *urban17* for testing, the others for training and validation.

A. Pose comparison

We select the advanced open-source interoceptive-only odometry methods for comparison: RINS-W [20], TLIO [23], LWOI [24] with FOG measurements (LWOI-FOG), and LWOI [24] with IMU's gyroscope measurements (LWOI-IMU). In order to compare the estimation accuracy, we use EVO [36] to evaluate the RMSE of absolute translation error (**ATE**) and rotation error (**ARE**):

- **ATE** (m):= $\sqrt{\frac{1}{M}\sum_{i=1}^{M}\|{}^{\mathcal{G}}\boldsymbol{p}_i - {}^{\mathcal{G}}\widehat{\boldsymbol{p}}_i\|_2^2}$,
- **ARE** (deg):= $\sqrt{\frac{1}{M}\sum_{i=1}^{M}\|\mathrm{Log}(\widehat{\boldsymbol{q}}^* \otimes \boldsymbol{q})\|_2^2}$,

where M is the number of the estimated poses.

The third to seventh rows of TABLE II show the translation and rotation RMSE results of all competing algorithms. These results demonstrate that our proposed method, WING, can achieve more accurate pose estimation than RINS-W [20], TLIO [23], and LWOI-IMU [24]. Additionally, even though LWOI-FOG [24] utilizes FOG measurements, our method can still compete with it in a part of test set. The 3D trajectories are drawn in Fig. 3, where TLIO [23] and LWOI-IMU [24] are not plotted as their divergent estimations. These results demonstrate the benefits of using the ground manifold, as well as the bias and velocity estimation from neural networks.

B. Evaluation of each module

1) IMU De-Bias Net: Since the performance of bias estimation has already been demonstrated in our previous work [33], this study focuses solely on evaluating the performance of the new covariance branch. To assess the consistency of the learned covariance, we gather samples from the complete test set. Fig. 4 presents the results of this analysis, which

TABLE II
TRANSLATION AND ROTATION EVALUATION IN KAIST URBAN DATASET [15]. THE **BEST** AND **SECOND BEST** RESULTS ARE SEPARATELY MARKED IN RED BOLD AND BLACK BOLD, RESPECTIVELY.

		RINS-W	LWOI-IMU	LWOI-FOG	TLIO	WING	WING w.o. M.
urban07	ATE	18.41	409.41	**5.17**	96.74	**8.10**	8.86
(2.55km)	ARE	2.34	84.56	5.08	44.96	**1.47**	**2.10**
urban09	ATE	214.88	2402.82	**19.31**	846.22	**22.25**	24.33
(15.7km)	ARE	10.85	100.54	4.23	114.24	**2.11**	**2.17**
urban11	ATE	113.90	156.29	70.79	4024.93	**40.25**	**44.81**
(17.33km)	ARE	4.58	3.77	2.58	45.53	**1.50**	**1.52**
urban13	ATE	449.29	624.22	**15.59**	425.82	**14.58**	20.22
(2.36km)	ARE	28.37	86.17	**2.51**	59.723	**1.94**	2.93
urban15	ATE	69.86	206.62	**8.08**	333.15	**7.75**	17.79
(5.43km)	ARE	8.55	32.35	3.78	65.18	**1.11**	**2.66**
urban17	ATE	86.10	1384.02	**11.52**	955.16	**12.64**	34.61
(10.32km)	ARE	4.49	57.33	3.12	69.87	**0.57**	**2.56**

demonstrate that more than 99.5% of the error points fall within the 3σ region. These observations confirm that the outputs generated by the *IMU De-Bias Net* exhibit high consistency. Moreover, the results reveal that the covariance of dq_y is significantly higher than that of dq_x and dq_z. This can be attributed to the fact that when a car is moving forward, its pitch angle is more susceptible to disturbances, leading to increased uncertainty in the estimation of dq_y.

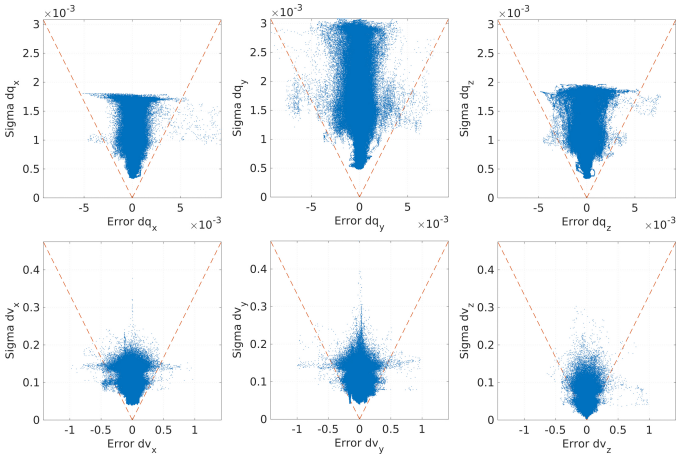

Fig. 4. Standard deviation σ against the errors in $\widehat{\boldsymbol{v}}_{i,i+n}$ (first row) and $\boldsymbol{q}_{i,i+n}$ (second row) along the x, y, and z axes. Error dv represents $\boldsymbol{v}_{i,i+n} - \widehat{\boldsymbol{v}}_{i,i+n}$, and error dq denotes $\mathrm{Log}\big((\widehat{\boldsymbol{q}}_{i,i+n})^* \otimes \boldsymbol{q}_{i,i+n}\big)$. The dashed red line represents the 3σ region. The results indicate that only 0.05% of the error points for the x-axis, 0.10% for the y-axis, and 0.38% for the z-axis were observed outside the 3σ bounds.

2) Wheel Encoder Net: We evaluate the performance of *Wheel Encoder Net* on the test set of KAIST Urban Dataset [15] and present the results in TABLE III. These results demonstrate that *Wheel Encoder Net* can significantly improve the accuracy of velocity estimation on the x-axis and y-axis. However, the enhancement on the z-axis is not as significant. This is because vehicles are more likely to skid during turning, whereas vertical bouncing is relatively rare on smooth urban roads.

Fig. 6 illustrates an example of the improved position estimation achieved by utilizing the *Wheel Encoder Net*. We integrate the velocity obtained from both the *Wheel Encoder Net* ($^{\mathcal{I}}\widehat{\boldsymbol{v}}_{\mathcal{I}}$) and raw measurements($^{\mathcal{I}}\widetilde{\boldsymbol{v}}_{\mathcal{I}} = {}^{\mathcal{I}}_{\mathcal{B}}\mathbf{R}\ {}^{\mathcal{B}}\widetilde{\boldsymbol{v}}_{\mathcal{B}} - {}^{\mathcal{I}}\widetilde{\boldsymbol{\omega}} \times {}^{\mathcal{I}}\boldsymbol{t}_{\mathcal{B}} - {}^{\mathcal{I}}\dot{\boldsymbol{t}}_{\mathcal{B}}$) using the ground truth attitude. The ${}^{\mathcal{I}}_{\mathcal{B}}\mathbf{R}$ and ${}^{\mathcal{I}}\boldsymbol{t}_{\mathcal{B}}$ are obtained from the data sheet of KAIST Urban Dataset [15]. As depicted in Fig. 6, utilizing the *Wheel Encoder Net* leads to more accurate position estimation.

TABLE III
$^{\mathcal{I}}\boldsymbol{v}_{\mathcal{I}}$ ESTIMATION ERROR. THE **BEST** RESULTS ARE MARKED IN RED BOLD.

	Raw measurements			***Wheel Encoder Net***		
	$^{\mathcal{I}}\boldsymbol{v}_{\mathcal{I}_x}$	$^{\mathcal{I}}\boldsymbol{v}_{\mathcal{I}_y}$	$^{\mathcal{I}}\boldsymbol{v}_{\mathcal{I}_z}$	$^{\mathcal{I}}\boldsymbol{v}_{\mathcal{I}_x}$	$^{\mathcal{I}}\boldsymbol{v}_{\mathcal{I}_y}$	$^{\mathcal{I}}\boldsymbol{v}_{\mathcal{I}_z}$
urban07	2.7e-2	3.6e-2	3.8e-3	**7.7e-3**	**3.7e-3**	**3.5e-3**
urban09	2.4e-2	2.7e-2	3.2e-3	**7.5e-3**	**3.4e-3**	**3.1e-3**
urban11	1.2e-1	3.8e-2	**3.7e-3**	**8.2e-2**	**2.2e-3**	3.9e-3
urban13	1.1e-2	1.6e-2	7.9e-4	**2.1e-3**	**7.2e-4**	**6.7e-4**
urban15	3.2e-2	4.1e-2	1.8e-3	**3.9e-3**	**1.4e-3**	**1.6e-3**
urban17	4.0e-2	3.5e-2	2.9e-3	**2.0e-2**	**4.6e-3**	**2.8e-3**

Similar to the consistency analysis performed on the *IMU De-Bias Net*, we have conducted a similar assessment on the *Wheel Encoder Net*. As illustrated in Fig. 5, over 99% of the points are situated within the 3σ region, indicating a high level of consistency in the outputs generated by the *Wheel Encoder Net*. Due to the skidding, the majority of errors in the x-axis of $^{\mathcal{I}}\widehat{\boldsymbol{v}}_{\mathcal{I}_i}$ are positive. Furthermore, as the vehicle is moving forward, the error and uncertainty along the x-axis are greater than those in the y-axis and z-axis.

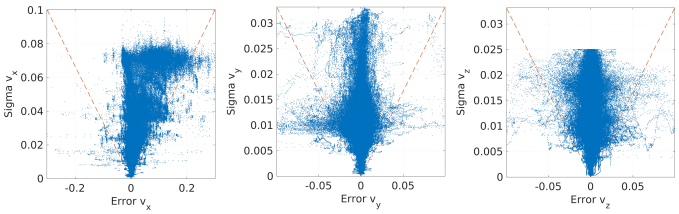

Fig. 5. Standard deviation σ against the errors in $^{\mathcal{I}}\widehat{\boldsymbol{v}}_{\mathcal{I}_i}$ along the x, y, and z axes. Error v represents $^{\mathcal{I}}\boldsymbol{v}_{\mathcal{I}_i} - {}^{\mathcal{I}}\widehat{\boldsymbol{v}}_{\mathcal{I}_i}$. The results reveal that only 0.10% of the error points for the x-axis, 0.03% for the y-axis, and 0.06% for the z-axis were observed outside the 3σ bounds.

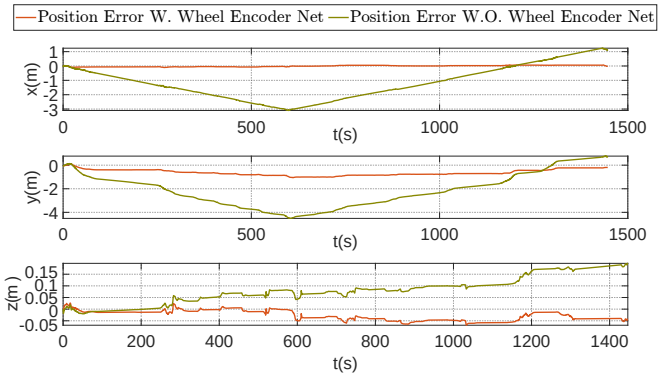

Fig. 6. The error in position estimation resulting from integrating velocity when using the *Wheel Encoder Net* compared to not using it on the *urban13* of KAIST Urban Dataset [15].

3) Manifold constraint: To further illustrate the improvement achieved by utilizing the manifold constraint within the EKF framework, we compare the complete WING system with WING without manifold constraint (WING w.o. M) in terms of their performance. The resulting RMSEs are listed in the seventh and eighth columns of TABLE II. The results confirm that the utilization of manifold constraints leads to better state estimation performance. This is due to the fact that the manifold constraint helps to smooth the poses over a history window and provides additional position and attitude constraints to the system.

VIII. Conclusion and Future Work

In this work, we propose a novel wheel-inertial odometry system that leverages deep neural networks and ground manifold constraints within a space-based sliding window filtering framework. Our system is specifically designed for ground robots, and we utilize a dual cubic B-spline surface to provide globally continuous manifold constraints. To fully exploit the benefits of both the IMU and wheel encoders, we have designed deep neural networks to reduce IMU biases and compensate for wheel encoder measurement errors caused by skidding and vertical bouncing. Additionally, the network implicitly calibrate the IMU and wheel encoder skidding, as well as estimate their uncertainties for better sensor fusion.

Experiments show that our proposed algorithm outperforms other learning-based methods. In addition, the effectiveness of neural velocity measurements and continuous surface constraints for pose estimation is shown in the ablation studies.

A. Future work

In the proposed system, there is no direct measurement of the ground but only a surface assumption. Therefore, if we can leverage some prior ground information, such as point clouds from LiDAR or depth camera, the ground manifold constraints may be more accurate and thus obtain better estimation performance. Besides, in this work, the covariance of added control points and the size of each knot vector in the B-spline manifold are invariant. In the future, we could use a learning-based method to output the dynamic covariance and time-varying knot vector, making the system more adaptive to match variant scenarios.

Appendix A
Proof of the Eq. (30)

Considering the following kinematic models as utilized in the training process of *De-Bias Net* :

$$\begin{aligned}\boldsymbol{v}_{i,\tau} &= \int_i^\tau {}^{\mathcal{G}}_{\mathcal{I}_t}\mathbf{R}({}^{\mathcal{I}_t}\widetilde{\boldsymbol{a}} - {}^{\mathcal{I}_t}\widehat{\boldsymbol{b}}_{\boldsymbol{a}} - \boldsymbol{n}_{\boldsymbol{a}})dt,\\ \boldsymbol{q}_{i,\tau} &= \int_i^\tau \frac{1}{2}{}^{\mathcal{I}_i}_{\mathcal{I}_t}\boldsymbol{q}\ ({}^{\mathcal{I}_t}\widetilde{\boldsymbol{\omega}} - {}^{\mathcal{I}_t}\widehat{\boldsymbol{b}}_{\boldsymbol{\omega}} - \boldsymbol{n}_{\boldsymbol{\omega}})dt,\end{aligned} \tag{42}$$

where τ is the time stamp between time i and $i+n$, ${}^{\mathcal{G}}_{\mathcal{I}_t}\mathbf{R}$ is the ground truth rotation, ${}^{\mathcal{I}_t}\widehat{\boldsymbol{b}}_{\boldsymbol{a}}$ and ${}^{\mathcal{I}_t}\widehat{\boldsymbol{b}}_{\boldsymbol{\omega}}$ are estimated values of the *De-Bias Net*, $\boldsymbol{n}_{\boldsymbol{a}}$ and $\boldsymbol{n}_{\boldsymbol{\omega}}$ are the input noise of the integral system in Eq. (42). We discretize the above equations to obtain the nominal state equations:

$$\begin{aligned}\boldsymbol{v}_{i,k+1} &= \boldsymbol{v}_{i,k} + {}^{\mathcal{G}}_{\mathcal{I}_k}\mathbf{R}({}^{\mathcal{I}_k}\widetilde{\boldsymbol{a}} - {}^{\mathcal{I}_k}\widehat{\boldsymbol{b}}_{\boldsymbol{a}})\Delta t_k,\\ \boldsymbol{q}_{i,k+1} &= \boldsymbol{q}_{i,k}\ \boldsymbol{q}\{({}^{\mathcal{I}_t}\widetilde{\boldsymbol{\omega}} - {}^{\mathcal{I}_t}\widehat{\boldsymbol{b}}_{\boldsymbol{\omega}})\Delta t_k\},\end{aligned} \tag{43}$$

with the following discrete error state models:

$$\begin{aligned}&\delta\boldsymbol{v}_{i,k+1} = \delta\boldsymbol{v}_{i,k} + {}^{\mathcal{G}}_{\mathcal{I}_k}\mathbf{R}\mathbf{v}_{\mathbf{i}} = \delta\boldsymbol{v}_{i,k} + \mathbf{v}_{\mathbf{i}},\\ &\delta\boldsymbol{\theta}_{i,k+1} = \mathbf{R}^\top\{({}^{\mathcal{I}_t}\widetilde{\boldsymbol{\omega}} - {}^{\mathcal{I}_t}\widehat{\boldsymbol{b}}_{\boldsymbol{\omega}})\Delta t_k\}\delta\boldsymbol{\theta}_{i,k} + \boldsymbol{\theta}_{\mathbf{i}},\\ &\mathbf{i} = \begin{bmatrix}\mathbf{v}_{\mathbf{i}}\\ \boldsymbol{\theta}_{\mathbf{i}}\end{bmatrix} \sim \mathcal{N}(0, \mathbf{Q}_{\mathbf{i}}), \quad \mathbf{Q}_{\mathbf{i}} = \Delta t^2\begin{bmatrix}\boldsymbol{\Sigma}_{\boldsymbol{a}}^2 & \mathbf{0}_3\\ \mathbf{0}_3 & \boldsymbol{\Sigma}_{\boldsymbol{\omega}}^2\end{bmatrix},\end{aligned} \tag{44}$$

where k and the $k+1$ are the IMU time stamps between time i and $i+n$, $\mathbf{i}$ is the perturbation impulse vector as in [35].To simplify the notation, we assume that each time interval Δt_k is equal to Δt. Note that, there is no error propagation about the rotation since the ground truth ${}^{\mathcal{G}}_{\mathcal{I}}\mathbf{R}$ is provided. We combine the two states in the Eq. (44) to yield the following error state equation:

$$\begin{aligned}\delta\boldsymbol{x}_{i,k+1} = \begin{bmatrix}\delta\boldsymbol{v}_{i,k+1}\\ \delta\boldsymbol{\theta}_{i,k+1}\end{bmatrix} &= \boldsymbol{f}(\boldsymbol{x}_{i,k}, \delta\boldsymbol{x}_{i,k}, \boldsymbol{u}_k, \mathbf{i})\\ &= \mathbf{F}_{\mathbf{x},k}(\boldsymbol{x}_{i,k}, \boldsymbol{u}_k)\delta\boldsymbol{x}_{i,k} + \mathbf{F}_{\mathbf{i},k}\ \mathbf{i},\end{aligned} \tag{45}$$

whose error state prediction equations are written:

$$\begin{aligned}\delta\widehat{\boldsymbol{x}}_{i,k+1} &= \mathbf{F}_{\mathbf{x},k}(\boldsymbol{x}_{i,k}, \boldsymbol{u}_k)\delta\widehat{\boldsymbol{x}}_{i,k},\\ \mathbf{P}_{i,k+1} &= \mathbf{F}_{\mathbf{x},k}\mathbf{P}_{i,k}\mathbf{F}_{\mathbf{x},k}^\top + \mathbf{F}_{\mathbf{i},k}\mathbf{Q}_{\mathbf{i}}\mathbf{F}_{\mathbf{i},k}^\top,\end{aligned} \tag{46}$$

where the Jacobi matrices are:

$$\mathbf{F}_{\mathbf{x},k} = \begin{bmatrix}\mathbf{I}_3 & \mathbf{0}_3\\ \mathbf{0}_3 & \mathbf{R}^\top\{({}^{\mathcal{I}_k}\widetilde{\boldsymbol{\omega}} - {}^{\mathcal{I}_k}\widehat{\boldsymbol{b}}_{\boldsymbol{\omega}})\Delta t\}\end{bmatrix}, \quad \mathbf{F}_{\mathbf{i}} = \begin{bmatrix}\mathbf{I}_3 & \mathbf{0}_3\\ \mathbf{0}_3 & \mathbf{I}_3\end{bmatrix}, \tag{47}$$

so we can obtain the propagation of the covariance as follows:

$$\begin{aligned}\mathbf{P}_{i,i+1} &= \mathbf{F}_{\mathbf{x},i+1}\mathbf{P}_{i,i}\mathbf{F}_{\mathbf{x},i+1}^\top + \mathbf{Q}_{\mathbf{i}},\\ \mathbf{P}_{i,i+2} &= \mathbf{F}_{\mathbf{x},i+2}\mathbf{F}_{\mathbf{x},i+1}\mathbf{P}_{i,i}\mathbf{F}_{\mathbf{x},i+1}^\top\mathbf{F}_{\mathbf{x},i+2}^\top\\ &\quad + \mathbf{F}_{\mathbf{x},i+2}\mathbf{P}_{i,i}\mathbf{F}_{\mathbf{x},i+2}^\top + \mathbf{Q}_{\mathbf{i}},\\ &\ \ \vdots\\ \mathbf{P}_{i,i+n} &= \mathbf{F}_{\mathbf{x},i+n}\cdots\mathbf{F}_{\mathbf{x},i+1}\mathbf{P}_{i,i}\mathbf{F}_{\mathbf{x},i+1}^\top\cdots\mathbf{F}_{\mathbf{x},i+n}^\top\\ &\quad + \mathbf{F}_{\mathbf{x},i+n}\cdots\mathbf{F}_{\mathbf{x},i+2}\mathbf{Q}_{\mathbf{i}}\mathbf{F}_{\mathbf{x},i+2}^\top\cdots\mathbf{F}_{\mathbf{x},i+n}^\top\\ &\quad + \cdots\\ &\quad + \mathbf{F}_{\mathbf{x},i+n}\mathbf{Q}_{\mathbf{i}}\mathbf{F}_{\mathbf{x},i+n}^\top + \mathbf{Q}_{\mathbf{i}},\end{aligned} \tag{48}$$

Assuming that the initial covariance matrix is:

$$\mathbf{P}_{i,i} = \begin{bmatrix}\mu\mathbf{I}_3 & \mathbf{0}_3\\ \mathbf{0}_3 & \nu\mathbf{I}_3\end{bmatrix},\ \mu{>}0,\ \nu{>}0, \tag{49}$$

then we can derive:

$$\mathbf{P}_{i,i+n} = \begin{bmatrix} \mu\mathbf{I}_3 + n\Delta t^2 \mathbf{\Sigma}_{\boldsymbol{a}}^2 & \mathbf{0}_3 \\ \mathbf{0}_3 & \nu\mathbf{I}_3 + n\Delta t^2 \mathbf{\Sigma}_{\boldsymbol{\omega}}^2 \end{bmatrix}. \tag{50}$$

Furthermore, since the initial values of each input window are ground truth in training the *De-Bias Net* and the initial covariance matrix $\mathbf{P}_{i,i}$ is very small (but cannot be set to 0), then we approximate that

$$\mathbf{P}_{i,i+n} = \begin{bmatrix} \mathbf{\Sigma}_{\boldsymbol{v}_{i,i+n}}^2 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{\Sigma}_{\boldsymbol{q}_{i,i+n}}^2 \end{bmatrix} \approx n\Delta t^2 \begin{bmatrix} \mathbf{\Sigma}_{\boldsymbol{a}}^2 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{\Sigma}_{\boldsymbol{\omega}}^2 \end{bmatrix}. \tag{51}$$

As a result, the estimated covariance matrices in Eq. (9) can be used to approximate the theoretical covariance matrices of the IMU data processed by the *De-Bias Net*:

$$\frac{1}{n\Delta t^2} \begin{bmatrix} \widehat{\mathbf{\Sigma}}_{\boldsymbol{v}_{i,i+n}}^2 \\ \widehat{\mathbf{\Sigma}}_{\boldsymbol{q}_{i,i+n}}^2 \end{bmatrix} \Longrightarrow \begin{bmatrix} \widehat{\mathbf{\Sigma}}_{\boldsymbol{a}}^2 \\ \widehat{\mathbf{\Sigma}}_{\boldsymbol{\omega}}^2 \end{bmatrix}. \tag{52}$$

Appendix B
Jacobian of Process model

The jacobian of Eq. (31) can be described as:

$$\mathbf{F}_{\mathcal{I}} = \begin{bmatrix} \mathbf{I}_3 & {}^{\mathcal{G}}_{\mathcal{I}_k}\mathbf{R}\Delta t_k & \frac{\partial {}^{\mathcal{G}}\boldsymbol{p}_{\mathcal{I}_{k+1}}}{\partial {}^{\mathcal{G}}_{\mathcal{I}_k}\psi} & \frac{\partial {}^{\mathcal{G}}\boldsymbol{p}_{\mathcal{I}_{k+1}}}{\partial {}^{\mathcal{G}}_{\mathcal{I}_k}\boldsymbol{s}} \\ \mathbf{0}_3 & \mathbf{I}_3 - \lfloor \boldsymbol{\omega}_k \rfloor_\times \Delta t_k & \mathbf{0}_{3\times 1} & \frac{\partial {}^{\mathcal{I}_{k+1}}\boldsymbol{v}}{\partial {}^{\mathcal{G}}_{\mathcal{I}_k}\boldsymbol{s}} \\ \mathbf{0}_{1\times 3} & \mathbf{0}_{1\times 3} & 1 & \frac{\partial {}^{\mathcal{G}}_{\mathcal{I}_{k+1}}\psi}{\partial {}^{\mathcal{G}}_{\mathcal{I}_k}\boldsymbol{s}} \\ \mathbf{0}_{2\times 3} & \mathbf{0}_{2\times 3} & \mathbf{0}_{2\times 1} & \frac{\partial {}^{\mathcal{G}}_{\mathcal{I}_{k+1}}\boldsymbol{s}}{\partial {}^{\mathcal{G}}_{\mathcal{I}_k}\boldsymbol{s}} \end{bmatrix}, \tag{53a}$$

$$\mathbf{F}_{\boldsymbol{n}} = \begin{bmatrix} \mathbf{0}_3 & \mathbf{0}_3 \\ -\mathbf{I}_3\Delta t_k & -\lfloor {}^{\mathcal{I}}\boldsymbol{v} \rfloor_\times \Delta t_k \\ \mathbf{0}_{1\times 3} & \frac{\partial {}^{\mathcal{G}}_{\mathcal{I}_{k+1}}\psi}{\partial \boldsymbol{n}_{\boldsymbol{\omega}}} \\ \mathbf{0}_{2\times 3} & \frac{\partial {}^{\mathcal{G}}_{\mathcal{I}_{k+1}}\boldsymbol{s}}{\partial \boldsymbol{n}_{\boldsymbol{\omega}}} \end{bmatrix}. \tag{53b}$$

Specifically,

$$\frac{\partial {}^{\mathcal{G}}\boldsymbol{p}_{\mathcal{I}_{k+1}}}{\partial {}^{\mathcal{G}}_{\mathcal{I}_k}\psi} = \begin{bmatrix} -\sin\psi & -\cos\psi & 0 \\ \cos\psi & -\sin\psi & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{R}_\phi {}^{\mathcal{I}}\boldsymbol{v}\Delta t_k, \tag{54a}$$

$$\frac{\partial {}^{\mathcal{G}}\boldsymbol{p}_{\mathcal{I}_{k+1}}}{\partial {}^{\mathcal{G}}_{\mathcal{I}_k}s_1} = \mathbf{R}_\psi \begin{bmatrix} \frac{-4s_1(1+s_2^2)}{1+s_1^2+s_2^2} & \frac{-2s_2(1-s_1^2+s_2^2)}{1+s_1^2+s_2^2} & \frac{2(1-s_1^2+s_2^2)}{1+s_1^2+s_2^2} \\ \frac{-2s_2(1-s_1^2+s_2^2)}{1+s_1^2+s_2^2} & \frac{4s_1s_2^2}{1+s_1^2+s_2^2} & \frac{-4s_1s_2}{1+s_1^2+s_2^2} \\ \frac{-2(1-s_1^2+s_2^2)}{1+s_1^2+s_2^2} & \frac{4s_1s_2}{1+s_1^2+s_2^2} & \frac{-4s_1}{1+s_1^2+s_2^2} \end{bmatrix} {}^{\mathcal{I}}\boldsymbol{v}\Delta t_k, \tag{54b}$$

$$\frac{\partial {}^{\mathcal{G}}\boldsymbol{p}_{\mathcal{I}_{k+1}}}{\partial {}^{\mathcal{G}}_{\mathcal{I}_k}s_2} = \mathbf{R}_\psi \begin{bmatrix} \frac{4s_1^2s_2}{1+s_1^2+s_2^2} & \frac{-2s_2(1+s_1^2-s_2^2)}{1+s_1^2+s_2^2} & \frac{-4s_1s_2}{1+s_1^2+s_2^2} \\ \frac{-2s_2(1+s_1^2-s_2^2)}{1+s_1^2+s_2^2} & \frac{-4s_2(1+s_1^2)}{1+s_1^2+s_2^2} & \frac{2(1+s_1^2-s_2^2)}{1+s_1^2+s_2^2} \\ \frac{4s_1s_2}{1+s_1^2+s_2^2} & \frac{-2(1+s_1^2-s_2^2)}{1+s_1^2+s_2^2} & \frac{-4s_2}{1+s_1^2+s_2^2} \end{bmatrix} {}^{\mathcal{I}}\boldsymbol{v}\Delta t_k, \tag{54c}$$

$$\frac{\partial {}^{\mathcal{I}_{k+1}}\boldsymbol{v}}{\partial {}^{\mathcal{G}}_{\mathcal{I}_k}\boldsymbol{s}} = -g \begin{bmatrix} \frac{-2(1-s_1^2+s_2^2)}{1+s_1^2+s_2^2} & \frac{4s_1s_2}{1+s_1^2+s_2^2} \\ \frac{4s_1s_2}{1+s_1^2+s_2^2} & \frac{-2(1+s_1^2-s_2^2)}{1+s_1^2+s_2^2} \\ \frac{-4s_1}{1+s_1^2+s_2^2} & \frac{-4s_2}{1+s_1^2+s_2^2} \end{bmatrix} \Delta t_k, \tag{54d}$$

$$\frac{\partial {}^{\mathcal{G}}_{\mathcal{I}_{k+1}}\psi}{\partial {}^{\mathcal{G}}_{\mathcal{I}_k}\boldsymbol{s}} = -\begin{bmatrix} \omega_1 & \omega_2 \end{bmatrix} \Delta t_k, \tag{54e}$$

$$\frac{\partial {}^{\mathcal{G}}_{\mathcal{I}_{k+1}}\boldsymbol{s}}{\partial {}^{\mathcal{G}}_{\mathcal{I}_k}\boldsymbol{s}} = \mathbf{I}_2 + \begin{bmatrix} \omega_2 s_1 - \omega_1 s_2 & -\omega_1 s1 - \omega_2 s_2 + \omega_3 \\ \omega_1 s1 + \omega_2 s_2 - \omega_3 & -\omega_2 s_s - \omega_1 s_2 \end{bmatrix} \Delta t_k, \tag{54f}$$

$$\frac{\partial {}^{\mathcal{G}}_{\mathcal{I}_{k+1}}\psi}{\partial \boldsymbol{n}_{\boldsymbol{\omega}}} = \begin{bmatrix} s_1 & s_2 & -1 \end{bmatrix} \Delta t_k, \tag{54g}$$

$$\frac{\partial {}^{\mathcal{G}}_{\mathcal{I}_{k+1}}\boldsymbol{s}}{\partial \boldsymbol{n}_{\boldsymbol{\omega}}} = \begin{bmatrix} s_1s_2 & \frac{1}{2}(-1-s_1^2+s_2^2) & -s_2 \\ \frac{1}{2}(1-s_1^2+s_2^2) & -s_1s_2 & s_1 \end{bmatrix} \Delta t_k, \tag{54h}$$

where $\{{}^{\mathcal{I}}\boldsymbol{v}, \psi, \phi, s_1, s_2, \omega_1, \omega_2, \omega_3\}$ are the abbreviations of $\{{}^{\mathcal{I}_k}\boldsymbol{v}, {}^{\mathcal{G}}_{\mathcal{I}_k}\psi, {}^{\mathcal{G}}_{\mathcal{I}_k}\phi, {}^{\mathcal{G}}_{\mathcal{I}_k}s_1, {}^{\mathcal{G}}_{\mathcal{I}_k}s_2, {}^{\mathcal{I}_k}\omega_1, {}^{\mathcal{I}_k}\omega_2, {}^{\mathcal{I}_k}\omega_3\}$ and $\lfloor \boldsymbol{a} \rfloor_\times$ is the skew symmetric matrix of any vector $\boldsymbol{a}$.

Appendix C
Jacobian of Observation model

The jacobian of Eq. (34) can be described as:

$$\frac{\partial \boldsymbol{h}_{\boldsymbol{v}}(\mathcal{X})}{\partial \mathcal{X}} = \begin{bmatrix} \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_{3\times(3+6n+16)} \end{bmatrix}, \tag{55}$$

The jacobian of the Eq. (36) with respect to the state $\mathcal{X}$ can be described as:

$$\frac{\partial \boldsymbol{h}_{u,v}(\mathcal{X})}{\partial \mathcal{X}} = \begin{bmatrix} \mathbf{0}_{3n_{u,v}\times 9} & \mathbf{0}_{3n_{u,v}\times 6n_{u,v}} & \frac{\partial \boldsymbol{h}_{u,v}(\mathcal{X})}{\partial \boldsymbol{c}} \end{bmatrix}. \tag{56a}$$

Specifically,

$$\frac{\partial \boldsymbol{h}_{u,v}(\mathcal{X})}{\partial \boldsymbol{c}} = \begin{bmatrix} \frac{\partial \boldsymbol{h}_{u,v,\boldsymbol{\xi}_1}(\mathcal{X})}{\partial \boldsymbol{c}} \\ \vdots \\ \frac{\partial \boldsymbol{h}_{u,v,\boldsymbol{\xi}_{n_{u,v}}}(\mathcal{X})}{\partial \boldsymbol{c}} \end{bmatrix}_{3n_{u,v}\times 16}, \tag{57a}$$

$$\frac{\partial \boldsymbol{h}_{u,v,\boldsymbol{\xi}}(\mathcal{X})}{\partial \boldsymbol{c}} = \begin{bmatrix} \boldsymbol{y}_w\mathbf{K}_y\mathbf{B} \otimes \boldsymbol{x}_w\mathbf{K}_x\mathbf{B}\mathbf{M}_S(u,v) \\ \boldsymbol{\partial y}_w\mathbf{K}_y\mathbf{B} \otimes \boldsymbol{x}_w\mathbf{K}_x\mathbf{B}\mathbf{M}_S(u,v) \\ \boldsymbol{y}_w\mathbf{K}_y\mathbf{B} \otimes \boldsymbol{\partial x}_w\mathbf{K}_x\mathbf{B}\mathbf{M}_S(u,v) \end{bmatrix}_{3\times 16}. \tag{57b}$$

The jacobian of the Eq. (35) with respect to the state $\mathcal{X}$ can be described as:

$$\frac{\partial \boldsymbol{h}_{g_x,g_y}(\mathcal{X})}{\partial \mathcal{X}} = \begin{bmatrix} \mathbf{0}_{3n_{g_x,g_y}\times 9} & \frac{\partial \boldsymbol{h}_{g_x,g_y}(\mathcal{X})}{\partial \mathcal{X}_S} & \frac{\partial \boldsymbol{h}_{g_x,g_y}(\mathcal{X})}{\partial \boldsymbol{c}} \end{bmatrix}. \tag{58a}$$

Specifically,

$$\frac{\partial \boldsymbol{h}_{g_x,g_y}(\mathcal{X})}{\partial \mathcal{X}_S} = \begin{bmatrix} \frac{\partial \boldsymbol{h}_{g_x,g_y,\boldsymbol{\xi}_1}(\mathcal{X})}{\partial \boldsymbol{\xi}_1} & \mathbf{0}_{3\times 6} & \cdots & \mathbf{0}_{3\times 6} \\ \mathbf{0}_{3\times 6} & \frac{\partial \boldsymbol{h}_{g_x,g_y,\boldsymbol{\xi}_2}(\mathcal{X})}{\partial \boldsymbol{\xi}_2} & \cdots & \mathbf{0}_{3\times 6} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{3\times 6} & \mathbf{0}_{3\times 6} & \cdots & \frac{\partial \boldsymbol{h}_{g_x,g_y,\boldsymbol{\xi}_{n_{g_x,g_y}}}(\mathcal{X})}{\partial \boldsymbol{\xi}_{n_{g_x,g_y}}} \end{bmatrix}, \tag{59a}$$

$$\frac{\partial \boldsymbol{h}_{g_x,g_y}(\mathcal{X})}{\partial \boldsymbol{c}} = \begin{bmatrix} \frac{\partial \boldsymbol{h}_{g_x,g_y,\boldsymbol{\xi}_1}(\mathcal{X})}{\partial \boldsymbol{c}} \\ \vdots \\ \frac{\partial \boldsymbol{h}_{g_x,g_y,\boldsymbol{\xi}_{n_{u,v}}}(\mathcal{X})}{\partial \boldsymbol{c}} \end{bmatrix}_{3n_{g_x,g_y}\times 16}, \tag{59b}$$

$$\frac{\partial \boldsymbol{h}_{g_x,g_y,\boldsymbol{\xi}}(\mathcal{X})}{\partial \boldsymbol{c}} = \begin{bmatrix} \boldsymbol{y}_w\mathbf{K}_y\mathbf{B} \otimes \boldsymbol{x}_w\mathbf{K}_x\mathbf{B} \\ \boldsymbol{\partial y}_w\mathbf{K}_y\mathbf{B} \otimes \boldsymbol{x}_w\mathbf{K}_x\mathbf{B} \\ \boldsymbol{y}_w\mathbf{K}_y\mathbf{B} \otimes \boldsymbol{\partial x}_w\mathbf{K}_x\mathbf{B} \end{bmatrix}_{3\times 16}, \tag{59c}$$

$$\frac{\partial \boldsymbol{h}_{g_x,g_y,\boldsymbol{\xi}}(\mathcal{X})}{\partial \boldsymbol{\xi}} = \begin{bmatrix} \frac{\partial \boldsymbol{h}_{g_x,g_y,\boldsymbol{\xi}}(\mathcal{X})}{\partial \boldsymbol{p}} & \frac{\partial \boldsymbol{h}_{g_x,g_y,\boldsymbol{\xi}}(\mathcal{X})}{\partial \psi} & \frac{\partial \boldsymbol{h}_{g_x,g_y,\boldsymbol{\xi}}(\mathcal{X})}{\partial \boldsymbol{s}} \end{bmatrix}_{3\times 6}, \tag{59d}$$

$$\frac{\partial \boldsymbol{h}_{g_x,g_y,\boldsymbol{\xi}}(\mathcal{X})}{\partial \boldsymbol{p}} = \begin{bmatrix} \boldsymbol{\partial x}_w\mathbf{G}\boldsymbol{y}_w^\top & \boldsymbol{x}_w\mathbf{G}\boldsymbol{\partial y}_w^\top & -1 \\ \boldsymbol{\partial x}_w\mathbf{G}\boldsymbol{\partial y}_w^\top & \boldsymbol{x}_w\mathbf{G}\boldsymbol{\partial^2 y}_w^\top & 0 \\ \boldsymbol{\partial^2 x}_w\mathbf{G}\boldsymbol{y}_w^\top & \boldsymbol{\partial x}_w\mathbf{G}\boldsymbol{\partial y}_w^\top & 0 \end{bmatrix}, \tag{59e}$$

$$\frac{\partial \boldsymbol{h}_{g_x,g_y,\boldsymbol{\xi}}(\mathcal{X})}{\partial \psi} = \begin{bmatrix} \boldsymbol{\partial x}_w\mathbf{G}\boldsymbol{y}_w^\top & \boldsymbol{x}_w\mathbf{G}\boldsymbol{\partial y}_w^\top & -1 \\ \boldsymbol{\partial x}_w\mathbf{G}\boldsymbol{\partial y}_w^\top & \boldsymbol{x}_w\mathbf{G}\boldsymbol{\partial^2 y}_w^\top & 0 \\ \boldsymbol{\partial^2 x}_w\mathbf{G}\boldsymbol{y}_w^\top & \boldsymbol{\partial x}_w\mathbf{G}\boldsymbol{\partial y}_w^\top & 0 \end{bmatrix} \frac{\partial \mathbf{R}\boldsymbol{t}}{\partial \psi}$$

$$+2\begin{bmatrix}0\\ \frac{s_1\cos\psi-s_2\sin\psi}{1-s_1^2-s_2^2}\\ \frac{-s_1\sin\psi-s_2\cos\psi}{1-s_1^2-s_2^2}\end{bmatrix}, \tag{59f}$$

$$\frac{\partial \boldsymbol{h}_{g_x,g_y,\boldsymbol{\xi}}(\mathcal{X})}{\partial \boldsymbol{s}} = \begin{bmatrix}\boldsymbol{\partial x}_w\mathbf{G}\boldsymbol{y}_w^\top & \boldsymbol{x}_w\mathbf{G}\boldsymbol{\partial y}_w^\top & -1\\ \boldsymbol{\partial x}_w\mathbf{G}\boldsymbol{\partial y}_w^\top & \boldsymbol{x}_w\mathbf{G}\boldsymbol{\partial^2 y}_w^\top & 0\\ \boldsymbol{\partial^2 x}_w\mathbf{G}\boldsymbol{y}_w^\top & \boldsymbol{\partial x}_w\mathbf{G}\boldsymbol{\partial y}_w^\top & 0\end{bmatrix}\frac{\partial \mathbf{R}\boldsymbol{t}}{\partial \boldsymbol{s}}$$
$$+2\begin{bmatrix}0 & 0\\ \frac{(1+s_1^2-s_2^2)\sin\psi+2s_1s_2\cos\psi}{(1-s_1^2-s_2^2)^2} & \frac{(1-s_1^2+s_2^2)\cos\psi+2s_1s_2\sin\psi}{(1-s_1^2-s_2^2)^2}\\ \frac{(1+s_1^2-s_2^2)\sin\psi-2s_1s_2\cos\psi}{(1-s_1^2-s_2^2)^2} & \frac{(-1+s_1^2-s_2^2)\cos\psi+2s_1s_2\sin\psi}{(1-s_1^2-s_2^2)^2}\end{bmatrix}, \tag{59g}$$

$$\frac{\partial \mathbf{R}\boldsymbol{t}}{\partial \psi} = \begin{bmatrix}-\sin\psi & \cos\psi & 0\\ \cos\psi & -\sin\psi & 0\\ 0 & 0 & 0\end{bmatrix}\mathbf{R}_\phi\boldsymbol{t}, \tag{59h}$$

$$\frac{\partial \mathbf{R}\boldsymbol{t}}{\partial s_1} = \mathbf{R}_\psi\begin{bmatrix}\frac{-4s_1(1+s_2^2}{1+s_1^2+s_2^2} & \frac{-2s_2(1-s_1^2+s_2^2)}{1+s_1^2+s_2^2} & \frac{2(1-s_1^2+s_2^2)}{1+s_1^2+s_2^2}\\ \frac{-2s_2(1-s_1^2+s_2^2)}{1+s_1^2+s_2^2} & \frac{4s_1s_2^2}{1+s_1^2+s_2^2} & \frac{-4s_1s_2}{1+s_1^2+s_2^2}\\ \frac{-2(1-s_1^2+s_2^2)}{1+s_1^2+s_2^2} & \frac{4s_1s_2}{1+s_1^2+s_2^2} & \frac{-4s_1}{1+s_1^2+s_2^2}\end{bmatrix}\boldsymbol{t}, \tag{59i}$$

$$\frac{\partial \mathbf{R}\boldsymbol{t}}{\partial s_2} = \mathbf{R}_\psi\begin{bmatrix}\frac{4s_1^2s_2}{1+s_1^2+s_2^2} & \frac{-2s_2(1+s_1^2-s_2^2)}{1+s_1^2+s_2^2} & \frac{-4s_1s_2}{1+s_1^2+s_2^2}\\ \frac{-2s_2(1+s_1^2-s_2^2)}{1+s_1^2+s_2^2} & \frac{-4s_2(1+s_1^2)}{1+s_1^2+s_2^2} & \frac{2(1+s_1^2-s_2^2)}{1+s_1^2+s_2^2}\\ \frac{4s_1s_2}{1+s_1^2+s_2^2} & \frac{-2(1+s_1^2-s_2^2)}{1+s_1^2+s_2^2} & \frac{-4s_2}{1+s_1^2+s_2^2}\end{bmatrix}\boldsymbol{t}, \tag{59j}$$

where $\{\mathbf{R}, \mathbf{R}_\psi, \mathbf{R}_\phi, \boldsymbol{t}\}$ are the abbreviations of $\{{}^{\mathcal{G}}_{\mathcal{I}}\mathbf{R}, {}^{\mathcal{G}}_{\mathcal{I}}\mathbf{R}_\psi, {}^{\mathcal{G}}_{\mathcal{I}}\mathbf{R}_\phi, {}^{\mathcal{I}}\boldsymbol{t}_w\}$, $\mathbf{G} = \mathbf{K}_x\mathbf{B}\mathbf{C}\mathbf{B}^\top\mathbf{K}_y^\top$.

REFERENCES

[1] B.-F. Wu, T.-T. Lee, H.-H. Chang, J.-J. Jiang, C.-N. Lien, T.-Y. Liao, and J.-W. Perng, "Gps navigation based autonomous driving system design for intelligent vehicles," in *2007 IEEE International Conference on Systems, Man and Cybernetics*. IEEE, 2007, pp. 3294–3299.

[2] Y. Li and J. Ibanez-Guzman, "Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems," *IEEE Signal Processing Magazine*, vol. 37, no. 4, pp. 50–61, 2020.

[3] J. Zhang and S. Singh, "LOAM: Lidar odometry and mapping in real-time," *Proceedings of Robotics: Science and Systems Conference*, July 2014.

[4] J. Lin and F. Zhang, "Loam livox: A fast, robust, high-precision lidar odometry and mapping package for lidars of small fov," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 3126–3131.

[5] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 3354–3361.

[6] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[7] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *2007 6th IEEE and ACM international symposium on mixed and augmented reality*. IEEE, 2007, pp. 225–234.

[8] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[9] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 15–22.

[10] 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, 2016.

[11] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *ICRA*, 2007.

[12] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[13] 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.

[14] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "Fast-lio2: Fast direct lidar-inertial odometry," *IEEE Transactions on Robotics*, pp. 1–21, 2022.

[15] J. Jeong, Y. Cho, Y.-S. Shin, H. Roh, and A. Kim, "Complex urban dataset with multi-level sensors from highly diverse urban environments," *The International Journal of Robotics Research*, vol. 38, no. 6, pp. 642–657, 2019.

[16] K. J. Wu, C. X. Guo, G. Georgiou, and S. I. Roumeliotis, "Vins on wheels," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 5155–5162.

[17] M. Zhang, Y. Chen, and M. Li, "Vision-aided localization for ground robots," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 2455–2461.

[18] R. Kang, L. Xiong, M. Xu, J. Zhao, and P. Zhang, "Vins-vehicle: A tightly-coupled vehicle dynamics extension to visual-inertial state estimator," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 3593–3600.

[19] H. Xiao, Y. Han, J. Zhao, J. Cui, L. Xiong, and Z. Yu, "Lio-vehicle: A tightly-coupled vehicle dynamics extension of lidar inertial odometry," *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 446–453, 2021.

[20] M. Brossard, A. Barrau, and S. Bonnabel, "Rins-w: Robust inertial navigation system on wheels," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 2068–2075.

[21] ——, "Ai-imu dead-reckoning," *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 4, pp. 585–595, 2020.

[22] S. Herath, H. Yan, and Y. Furukawa, "RoNIN: Robust neural inertial navigation in the wild: Benchmark, evaluations, new methods," *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3146–3152, 2020.

[23] W. Liu, D. Caruso, E. Ilg, J. Dong, A. Mourikis, K. Daniilidis, V. Kumar, J. Engel, A. Valada, and T. Asfour, "TLIO: Tight learned inertial odometry," *IEEE Robotics and Automation Letters*, p. 1–1, 2020. [Online]. Available: http://dx.doi.org/10.1109/LRA.2020.3007421

[24] M. Brossard and S. Bonnabel, "Learning wheel odometry and imu errors for localization," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 291–297.

[25] S. Srinivasan, I. Sa, A. Zyner, V. Reijgwart, M. I. Valls, and R. Siegwart, "End-to-end velocity estimation for autonomous racing," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6869–6875, 2020.

[26] A. L. Escoriza, G. Revach, N. Shlezinger, and R. J. Van Sloun, "Data-driven kalman-based velocity estimation for autonomous racing," in *2021 IEEE International Conference on Autonomous Systems (ICAS)*. IEEE, 2021, pp. 1–5.

[27] B. Liang and N. Pears, "Visual navigation using planar homographies," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 1. IEEE, 2002, pp. 205–210.

[28] D. Scaramuzza, "1-point-ransac structure from motion for vehicle-mounted cameras by exploiting non-holonomic constraints," *International journal of computer vision*, vol. 95, no. 1, pp. 74–85, 2011.

[29] M. Zhang, X. Zuo, Y. Chen, Y. Liu, and M. Li, "Pose estimation for ground robots: On manifold representation, integration, reparameterization, and optimization," *IEEE Transactions on Robotics*, vol. 37, no. 4, pp. 1081–1099, 2021.

[30] W. Lee, K. Eckenhoff, Y. Yang, P. Geneva, and G. Huang, "Visual-inertial-wheel odometry with online calibration," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 4559–4566.

[31] T. Yap, M. Li, A. I. Mourikis, and C. R. Shelton, "A particle filter for monocular vision-aided odometry," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 5663–5669.

[32] J. Svacha, G. Loianno, and V. Kumar, "Inertial yaw-independent velocity and attitude estimation for high-speed quadrotor flight," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1109–1116, 2019.

[33] K. Zhang, C. Jiang, J. Li, S. Yang, T. Ma, C. Xu, and F. Gao, "Dido: Deep inertial quadrotor dynamical odometry," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 9083–9090, 2022.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.

[35] J. Sola, "Quaternion kinematics for the error-state kalman filter," *arXiv preprint arXiv:1711.02508*, 2017.

[36] M. Grupp, "evo: Python package for the evaluation of odometry and slam." https://github.com/MichaelGrupp/evo, 2017.