WING: Wheel-Inertial Neural Odometry with Ground Manifold Constraints

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

***Abstract*— We present learning-based wheel-inertial odometry that explicitly models the ground surface on which our ground robot is driving as a dual cubic B-spline manifold. Furthermore, to take full advantage of interceptive information in the degraded scenarios of sensors such as the Global Positioning System (GPS), camera, and Light Detection And Ranging (LiDAR), the proposed system trains deep neural networks to obtain the Inertial Measurement Unit (IMU) biases and the wheel encoder kinematics, respectively. Following such a continuous ground manifold assumption, extra analytical constraints are imposed on the state estimation to improve the 6 degrees of freedom (DOF) positioning accuracy. To maintain the C^1 continuity of the consistently reconstructed manifolds and the independence with yaw angle drift, we leverage a novel space-based sliding window filtering framework to fuse the above multi-source information in a yaw-independent attitude convention. Experiments in complex and large-scale urban environments have demonstrated that our proposed work outperforms state-of-the-art learning-based interoceptive-only odometry methods.**

I. Introduction

Ground robots are extensively employed in autonomous driving and environmental exploration scenarios. In terms of robot localization, common robot positioning algorithms fuse exteroceptive sensors such as GPS [1], LiDAR [2], and vision [3] to perform an inside-out localization strategy under most conditions. When robots face degenerated cases with less or interfering information from these exteroceptive sensors (e.g., GPS-denied, lousy illumination, severe weather, repetitive surroundings, etc.), interoceptive sensors, as a complement, including IMUs and wheel encoders, become critical to support relative positioning.

A common IMU consisting of a triaxial gyroscope and a triaxial accelerometer typically provides 6DOF relative motion constraints using the pre-integration technique [4] for odometry [5, 6] or a Simultaneous Localization And Mapping system (SLAM) [7, 8]. However, in the absence of real-time calibration of intrinsic parameters, raw IMU measurements often drift significantly after several integrations. While, wheel encoders provide only two-dimensional observations by simplifying the ground robot to a two-wheel [9, 10] or bicycle [11, 12] model. Besides, some assumptions such as no lateral slipping and vertical bouncing [13] are formulated to constrain the degrees of freedom of the robot, but which are far-fetched for all scenarios.

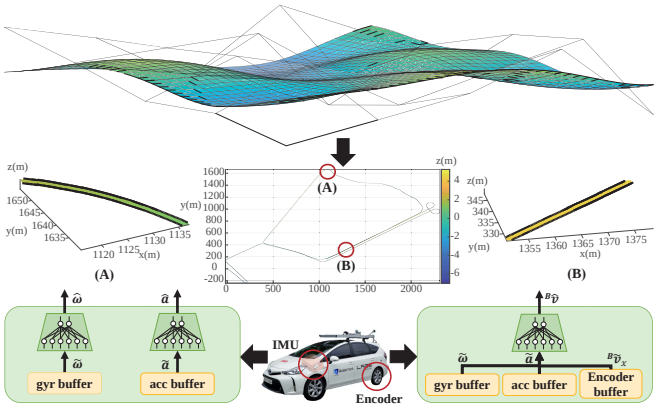

Fig. 1: The pipeline of our proposed odometry. We estimate poses by processing the interceptive sensor data through neural networks with simultaneously reconstructing a continuous cubic B-spline surface. The middle row shows the estimated position and terrain of scenario *urban17* in KAIST Urban Dataset [15], **A** and **B** represent the corner and straight routes respectively.

To further improve the robustness and accuracy of localization under various failures or degradations of the exteroceptive sensors, some data-driven approaches like RINS-W [13] and LWOI [14] could use only the measurements of the IMU or wheel encoder to infer the robot's pose, but which do not take advantage of the platform's motion constraints. These works inspire us to learn the robot's motion patterns through neural network processing and replace the tedious and complicated calibration procedure of intrinsic and extrinsic parameters. Moreover, to better portray the consistency and continuity of the ground robot during its motion, we leverage a continuous cubic B-spline surface to model the ground manifold to constrain the position and tilt of the ground robot, instead of using a discrete planar surface [9] or a discrete curved surface [10]. The proposed work contributes as follows:

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 Autonomous Driving Lab, Hangzhou 311121, China.

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

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

[*]The authors have the same contribution.

1) Parameterizing the ground manifold by a continuous cubic B-spline surface to impose analytical constraints;
2) Utilizing a space-based sliding window filtering to estimate the robot's poses along with a continuous ground manifold's parameters;
3) Exploiting interceptive information by deep neural networks to learn the kinematic properties of IMU and wheel encoder.

II. Related works

A. Interoceptive-only navigation systems

With the tremendous improvement of deep learning in computer vision and natural language processing, researchers embark on learning the characteristics of motion platforms through a data-driven approach.

[16] and [17] could track car's velocity when autonomous racing. [16] presents an end-to-end GRU network that takes IMU, torque, wheel speed, and steering angle as inputs. [17] proposes a hybrid model-based KalmanNet that learns the Kalman gain by training a GRU network, which can overcome the model mismatch and fuse the interpretability of Kalman filtering. However, neither of these estimated velocities is used to improve pose estimation accuracy.

Novelly, Wheel-INS [18] proposes a dead reckoning system using wheel-mounted IMU, where the authors replace the wheel odometry with the gyroscope output to reduce the constant bias during wheel rotation.

Some works [13, 14, 19]–[21] utilize only the on-board IMU measurements to obtain complete pose estimation. RINS-W [13] trains an LSTM network to build a specific motion profile detector including zero velocity, zero angular velocity, zero lateral velocity, and zero vertical velocity, which is fused into an Iterative Extended Kalman Filtering (IEKF) framework to get the robot states. LWOI [14] takes Fiber Optical Gyroscope (FOG), wheel encoder, and IMU measurements as input to train a Gaussian process to correct dynamical and observation models, which are fused in an Extended Kalman Filtering (EKF) framework. AI-IMU [19] utilizes a convolutional neural network (CNN) to dynamically output the observation noise parameters of the invariant extended Kalman filtering framework with two directions of robot motion constraints considered. RoNIN [20] uses several different neural networks to regress a moving subject's horizontal positions and direction. TLIO [21] learns pseudo-measurements of the relative displacement and places it as an observation in a tightly coupled statistical cloning extended Kalman filtering. However, these approaches are not explicitly optimized for ground-based robots, such as imposing ground constraints.

B. Exteroceptive navigation systems with ground constraints

Early, [22] estimates the motion of the ground-constrained robot by solving the homography matrix under the assumption of planar ground. Subsequently, [23] proposes a one-point RANSAC outlier rejection, which exploits the non-holonomic constraints of wheeled robots to accelerate pose estimation. Unfortunately, the above approaches only focus on joining motion constraints in the visual data association rather than considering coupling constraints in the optimization problem.

Some works [9, 10, 24, 25] are based on the two-wheel vehicle model, where the robot's forward linear speed and angular speed are deduced from the two-wheel encoders. For example, [9] extends Visual-Inertial-Navigation-System (VINS) to incorporate wheel-encoder measurements to ensure the scale is observable anytime. Besides, this work adds approximately planar manifold constraints into the system to reduce the estimation error. VIWO [24] proposes a Multi-State Constraint Kalman Filter (MSCKF) based visual-inertial-wheel odometry system to estimate the robot's motion state, wheel encoders' intrinsic and extrinsic parameters, as well as wheel-IMU time offset. Without IMU measurements, [10, 25] estimate the 6D pose of the robots and recover the ground manifold only by wheel encoder and visual measurements. In detail, to recover the 3-axis angular velocity, the ground manifold is parameterized explicitly in the form of a quadratic surface. Based on the bicycle model, both VINS-vehicle [11] and LIO-vehicle [12] use the wheel encoder and steering angle sensors measurements to build a two-degree-of-freedom vehicle dynamics model, and then construct a pre-integration factor for the factor graph optimization. The above exteroceptive sensor fusion schemes explicitly designed for ground robots could improve the robustness of pose estimation in most scenarios. Hence, interoceptive-based odometry considering the ground manifolds has the potential to further reduce uncertain DOF in case of exteroceptive sensor failure.

III. Preliminaries

A. Wheel odometer model

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

$$\begin{aligned} {}^{b}v_x &= \frac{\omega_l * r_l + \omega_r * r_r}{2} \\ {}^{b}\omega_x &= \frac{\omega_r * r_r - \omega_l * r_l}{w_b} \end{aligned} \tag{1}$$

where ω_l and ω_r are speeds of the left and right wheels, r_l and r_r are the radii of two wheels, and w_b is the wheelbase between two wheels.

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:

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

where ${}^{\mathcal{I}}\boldsymbol{a}$ and ${}^{\mathcal{I}}\boldsymbol{\omega}$ are the true angular velocity and acceleration, ${}^{\mathcal{G}}\mathbf{g} = [0, 0, 9.8m/s^2]$ is the gravity vector in the gravity-aligned frame $\mathcal{G}$ (with z-axis pointing down vertically), ${}^{\mathcal{G}}_{\mathcal{I}}\mathbf{R}$

is the rotation matrix from frame $\mathcal{I}$ to frame $\mathcal{G}$, $\boldsymbol{n_a}$ and $\boldsymbol{n_\omega}$ are the additive Gaussian white noise in gyroscope and acceleration measurements, $\boldsymbol{b_a}$ and $\boldsymbol{b_\omega}$ are the bias of IMU modeled as a random walk:

$$\begin{aligned} \boldsymbol{n_a} &\sim \mathcal{N}(0, \boldsymbol{\Sigma}_{\boldsymbol{a}}^2), \quad \dot{\boldsymbol{b}}_{\boldsymbol{a}} \sim \mathcal{N}(0, \boldsymbol{\Sigma}_{\boldsymbol{b_a}}^2), \\ \boldsymbol{n_\omega} &\sim \mathcal{N}(0, \boldsymbol{\Sigma}_{\boldsymbol{\omega}}^2), \quad \dot{\boldsymbol{b}}_{\boldsymbol{\omega}} \sim \mathcal{N}(0, \boldsymbol{\Sigma}_{\boldsymbol{b_\omega}}^2). \end{aligned} \tag{3}$$

C. Yaw independent attitude convention

In [26], an attitude convention is proposed, which decouples the attitude into yaw and tilt angles. Thus, the estimations of velocity and tilt could avoid being affected by the drift of yaw angle when updating by 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. Ground manifold representation

A ground robot always travels on the ground, which means that its three-dimensional (3D) position satisfies the ground's manifold constraints, and the robot's normal direction coincides with the manifold's normal direction. As the ground is typically continuous and smooth, we use a dual cubic B-spline surface to parameterize the ground manifold as:

$$\begin{aligned} \mathcal{M}(\boldsymbol{p}) &= \sum_{i=0}^{3}\sum_{j=0}^{3} B_{i,3}(u) B_{j,3}(v) c_{i,j} - z = 0 \\ &= \boldsymbol{u}\mathbf{B}\mathbf{C}\mathbf{B}^\top \boldsymbol{v}^\top - z = 0, \\ u &= k_x x + b_x \in [0,1), k_x = 1/d, b_x = -g_x/d, \\ v &= k_y y + b_y \in [0,1), k_y = 1/d, b_y = -g_y/d, \end{aligned} \tag{5}$$

where $\boldsymbol{p} = (x, y, z)^\top$ is a 3D point on the manifold, $B_{i,k}$ and $B_{j,l}$ are a basis function of the cubic B-spline curve, $c_{i,j}$ is the control point of the B-spline, d is the distance between adjacent surfaces, (g_x, g_y) is the coordinate of the ground grid. In this work, a uniform cubic B-spline is employed. Thus, Eq. 5 can be abbreviated as follow:

$$\begin{aligned} \mathcal{M}(\boldsymbol{p}) &= \boldsymbol{u}\mathbf{B}\mathbf{C}\mathbf{B}^\top \boldsymbol{v}^\top - z = 0, \\ &= \boldsymbol{x}\mathbf{K}_x\mathbf{B}\mathbf{C}\mathbf{B}^\top \mathbf{K}_y^\top \boldsymbol{y}^\top - z = 0, \end{aligned} \tag{6}$$

where,

$$\begin{aligned} &\boldsymbol{t} = [t^3, t^2, t, 1], t \in \{u, v, x, y\}, \\ &\mathbf{B} = \frac{1}{6}\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}, \mathbf{C} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \\ c_{41} & c_{42} & c_{43} & c_{44} \end{bmatrix}, \end{aligned} \tag{7}$$

$$\mathbf{K}_l = \begin{bmatrix} k_l^3 & 0 & 0 & 0 \\ 3k_l^2 b_l & k_l^2 & 0 & 0 \\ 3k_l b_l^2 & 2k_l b_l & k_l & 0 \\ b_l^3 & b_l^2 & b_l & 1 \end{bmatrix}, l \in \{x, y\}. \tag{8}$$

IV. Space-based sliding window fusion

In order to constrain the ground robot's poses using the ground manifold, we use a space-based sliding window filtering. This specialized filtering is designed not only to smooth the poses within a window but also to recover the parameters of the dual cubic B-sample surface.

A. State

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

$$\begin{aligned} \mathcal{X} &= \begin{bmatrix} \mathcal{X}_I{}^\top & \mathcal{X}_S{}^\top & \boldsymbol{c}^\top \end{bmatrix}^\top, \\ \mathcal{X}_I &= \begin{bmatrix} {}^{\mathcal{G}}\boldsymbol{p}_{\mathcal{I}}{}^\top & {}^{\mathcal{I}}\boldsymbol{v}_{\mathcal{I}}{}^\top & {}^{\mathcal{G}}_{\mathcal{I}}\psi & {}^{\mathcal{G}}_{\mathcal{I}}\boldsymbol{s}^\top \end{bmatrix}^\top, \\ \mathcal{X}_S &= \begin{bmatrix} \boldsymbol{\xi}_1 & \boldsymbol{\xi}_2 & \cdots & \boldsymbol{\xi}_n \end{bmatrix}^\top, \quad \boldsymbol{\xi}_j = \begin{bmatrix} {}^{\mathcal{G}}\boldsymbol{p}_{\mathcal{I}_j}{}^\top & {}^{\mathcal{G}}_{\mathcal{I}_j}\psi \end{bmatrix}^\top, \\ \boldsymbol{c} &= Vec(\mathbf{C}), \end{aligned} \tag{9}$$

where ${}^{\mathcal{G}}\boldsymbol{p}_{\mathcal{I}}$ is the position of the ground robot in $\mathcal{G}$ frame, ${}^{\mathcal{I}}\boldsymbol{v}_{\mathcal{I}}$ is the velocity of the robot in $\mathcal{I}$ frame, ${}^{\mathcal{G}}_{\mathcal{I}}\psi$ is the yaw angle in $\mathcal{G}$ frame, ${}^{\mathcal{G}}_{\mathcal{I}}\boldsymbol{s}$ is the tilt vector in $\mathcal{G}$ 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 control mesh matrix $\mathbf{C}$.

B. Process model

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

$$\begin{aligned} {}^{\mathcal{G}}\dot{\boldsymbol{p}}_{\mathcal{I}} &= {}^{\mathcal{G}}_{\mathcal{I}}\mathbf{R}\,{}^{\mathcal{I}}\boldsymbol{v}_{\mathcal{I}}, \\ {}^{\mathcal{I}}\dot{\boldsymbol{v}}_{\mathcal{I}} &= {}^{\mathcal{I}}\boldsymbol{a} - {}^{\mathcal{G}}_{\mathcal{I}}\mathbf{R}^\top {}^{\mathcal{G}}\mathbf{g} - {}^{\mathcal{I}}\boldsymbol{\omega} \times {}^{\mathcal{I}}\boldsymbol{v}_{\mathcal{I}}, \\ {}^{\mathcal{G}}_{\mathcal{I}}\dot{\psi} &= \begin{bmatrix} -s_1 & -s_2 & 1 \end{bmatrix} {}^{\mathcal{I}}\boldsymbol{\omega}, \\ {}^{\mathcal{G}}_{\mathcal{I}}\dot{\boldsymbol{s}} &= \frac{1}{2}\begin{bmatrix} -2s_1s_2 & s_1^2 - s_2^2 + 1 & 2s_2 \\ s_1^2 - s_2^2 - 1 & 2s_1s_2 & -2s_1 \end{bmatrix} {}^{\mathcal{I}}\boldsymbol{\omega}, \end{aligned} \tag{10}$$

where ${}^{\mathcal{G}}_{\mathcal{I}}\mathbf{R} = \mathbf{R}_\psi \mathbf{R}_\phi$, ${}^{\mathcal{G}}_{\mathcal{I}}\boldsymbol{s} = [s_1, s_2]^\top$,

$$\begin{aligned} {}^{\mathcal{I}}\widehat{\boldsymbol{\omega}} &= {}^{\mathcal{I}}\widetilde{\boldsymbol{\omega}} - {}^{\mathcal{I}}\widehat{\boldsymbol{b}}_{\boldsymbol{\omega}} + \boldsymbol{n_a}, \quad \boldsymbol{n_\omega} \sim \mathcal{N}(0, \boldsymbol{\Sigma}_{\boldsymbol{\omega}}^2), \\ {}^{\mathcal{I}}\widehat{\boldsymbol{a}} &= {}^{\mathcal{I}}\widetilde{\boldsymbol{a}} - {}^{\mathcal{I}}\widehat{\boldsymbol{b}}_{\boldsymbol{a}} + \boldsymbol{n_\omega}, \quad \boldsymbol{n_a} \sim \mathcal{N}(0, \boldsymbol{\Sigma}_{\boldsymbol{a}}^2). \end{aligned} \tag{11}$$

Due to the favorable results of our previous work [27], we inherit the *De-Bias Net* to output the estimated biases ${}^{\mathcal{I}}\widehat{\boldsymbol{b}}_{\boldsymbol{\omega}}$ and ${}^{\mathcal{I}}\widehat{\boldsymbol{b}}_{\boldsymbol{a}}$ of IMU in ground robots.

C. Observation model

There are three kinds of measurements in the proposed system.

1) Neural velocity measurement: According to Eq. (1), we obtain velocity measurements in the forward direction. Commonly, it is assumed that there is no speed in the non-forward direction, namely:

$$^{\mathcal{B}}\boldsymbol{v}_{\mathcal{B}} = \begin{bmatrix} ^{\mathcal{B}}v_{\mathcal{B}_x} & 0 & 0 \end{bmatrix}^\top. \tag{12}$$

In fact, the above constraints are not strictly satisfied, and its extrinsic parameters are not always fixed when a ground robot with an independent suspension system is in motion.

To avoid imposing incorrect constraints or estimating the time-varying parameters in the odometry system, we use the same neural network architecture as the *De-Bias Net* in our previous work [27] to compensate for the triaxial residuals between wheel encoder measurements and the ground truth velocity in the IMU frame, which takes raw IMU and wheel encoder measurements as input. In addition, the uncertainty of the network compensation is also output, which is trained by the negative log likelihood (NLL) loss as in [28]. In detail, the Mean Square Error (MSE) loss is replaced by the NLL loss after the MSE loss stabilizes and converges:

$$\begin{aligned} \mathcal{L}_{\text{MSE},\boldsymbol{v}} &= \frac{1}{N}\sum_{i=1}^{N} \|^{\mathcal{I}}\boldsymbol{v}_{\mathcal{I}i} - ^{\mathcal{I}}\widehat{\boldsymbol{v}}_{\mathcal{I}i}\|_2^2, \\ \mathcal{L}_{\text{NLL},\boldsymbol{v}} &= \frac{1}{2N}\sum_{i=1}^{N} \log\det(\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{v}_i}) + \frac{1}{N}\sum_{i=1}^{N} \|^{\mathcal{I}}\boldsymbol{v}_{\mathcal{I}i} - ^{\mathcal{I}}\widehat{\boldsymbol{v}}_{\mathcal{I}i}\|^2_{\widehat{\boldsymbol{\Sigma}}_{\boldsymbol{v}_i}}, \end{aligned} \tag{13}$$

where

$$^{\mathcal{I}}\widehat{\boldsymbol{v}}_{\mathcal{I}} = {}^{\mathcal{I}}_{\mathcal{B}}\mathbf{R}\,^{\mathcal{B}}\boldsymbol{v}_{\mathcal{B}} + {}^{\mathcal{I}}\delta\widehat{\boldsymbol{v}}_{\mathcal{I}}, \tag{14}$$

$$^{\mathcal{I}}\boldsymbol{v}_{\mathcal{I}} = \begin{bmatrix} ^{\mathcal{I}}\boldsymbol{v}_{\mathcal{I}_x} & ^{\mathcal{I}}\boldsymbol{v}_{\mathcal{I}_y} & ^{\mathcal{I}}\boldsymbol{v}_{\mathcal{I}_z} \end{bmatrix}^\top, \tag{15}$$

${}^{\mathcal{I}}_{\mathcal{B}}\mathbf{R}$ is the rotation between the $\mathcal{I}$ frame and the $\mathcal{B}$ frame, ${}^{\mathcal{I}}\delta\widehat{\boldsymbol{v}}_{\mathcal{I}}$ is the output of the network, $^{\mathcal{I}}\boldsymbol{v}_{\mathcal{I}}$ is the ground truth velocity in the $\mathcal{I}$ frame.

According to Eq. 14, we obtain triaxial pseudo-measurements of the velocity through a convolutional neural network as follows:

$$\begin{aligned} ^{\mathcal{I}}\widehat{\boldsymbol{v}}_{\mathcal{I}} &= \begin{bmatrix} ^{\mathcal{I}}\widehat{\boldsymbol{v}}_{\mathcal{I}_x} & ^{\mathcal{I}}\widehat{\boldsymbol{v}}_{\mathcal{I}_y} & ^{\mathcal{I}}\widehat{\boldsymbol{v}}_{\mathcal{I}_z} \end{bmatrix}^\top, \\ &= {}^{\mathcal{I}}\boldsymbol{v}_{\mathcal{I}} + \boldsymbol{n}_{\boldsymbol{v}}, \end{aligned} \tag{16}$$

where $\boldsymbol{n}_{\boldsymbol{v}} \sim \mathcal{N}(0, \widehat{\boldsymbol{\Sigma}}^2_{\boldsymbol{v}})$ is the output noise of the network.

2) Manifold constraint: In consideration of the fact that ground robots usually travel on smooth terrain, there are two observation equations, one indicating that the robot is on the ground manifold and the other indicating that the normal of the robot is parallel to the manifold normal:

$$\begin{aligned} 0 &= \mathcal{M}\left({}^{\mathcal{G}}\boldsymbol{p}_{\mathcal{I}}\right), \\ \mathbf{0} &= {}^{\mathcal{G}}_{\mathcal{B}}\mathbf{R}\cdot\mathbf{e}_3 \times \nabla\mathcal{M}\left({}^{\mathcal{G}}\boldsymbol{p}_{\mathcal{I}}\right), \end{aligned} \tag{17}$$

where $\mathcal{M}$ is the ground manifold represented by the dual cubic B-spline surface, $\mathbf{e}_3 = [0,0,1]^\top$ is the z-axis in any frame. In order to fuse these manifold constraints in an EKF framework, we rewrite Eq. 17 as follows:

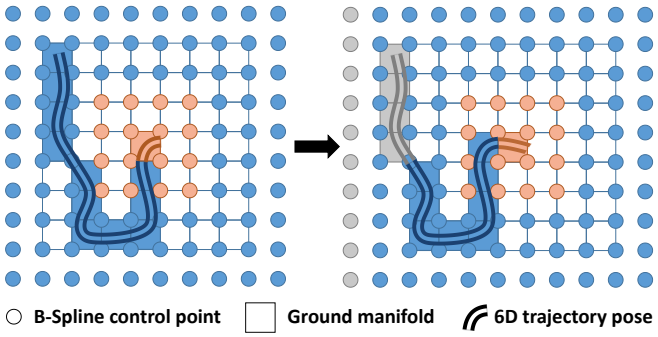

Fig. 2: The demonstration of space-based sliding window update strategy. Yellow, blue and gray represent the active, the static and the marginalized states or information in the proposed system, respectively.

$$\begin{aligned} 0 &= \boldsymbol{x}\mathbf{K}_x\mathbf{B}\mathbf{C}\mathbf{B}^\top\mathbf{K}_y^\top\boldsymbol{y} - z, \\ 0 &= \boldsymbol{x}\mathbf{K}_x\mathbf{B}\mathbf{C}\mathbf{B}^\top\mathbf{K}_y^\top\boldsymbol{\partial y} + 2\frac{s_1\sin(\psi) + s_2\cos(\psi)}{1 - s_1^2 - s_2^2}, \\ 0 &= \boldsymbol{\partial x}\mathbf{K}_x\mathbf{B}\mathbf{C}\mathbf{B}^\top\mathbf{K}_y^\top\boldsymbol{y} + 2\frac{s_1\cos(\psi) - s_2\sin(\psi)}{1 - s_1^2 - s_2^2}, \end{aligned} \tag{18}$$

where $\boldsymbol{\partial x} = [3x^2, 2x, 1, 0]$ and $\boldsymbol{\partial y} = [3y^2, 2y, 1, 0]$.

If there are n trajectory poses in the current ground grid in the updating step, we assume that a total of $2n$ trajectory poses (calculated by the known extrinsic parameters) in both left and right wheels satisfy the manifold constraints. In the left subplot of Fig. 2, these constraints are divided into two parts, one for the active ground (the yellow ground) and the other for the static grounds (the blue grounds). The active trajectory poses (the yellow curves) are located on the active ground, which are jointly optimized by Eq. 17 with their corresponding control points (the yellow circles). However, the static poses (the blue curves) are on the static grounds, which are used to better estimate the active control points (the yellow circles) with the restriction of the other static control points (the blue circles). As a result, the manifold observation model for the active control points can be derived as follows:

$$\begin{aligned} \mathbf{A}\mathbf{M}_S Vec(\mathbf{C}_{active}) &= \boldsymbol{b} - \mathbf{A}\mathbf{M}_N Vec(\mathbf{C}_{id_x,id_y}), \\ \mathbf{A} = \begin{bmatrix} \boldsymbol{y}\mathbf{K}_y\mathbf{B}\otimes\boldsymbol{x}\mathbf{K}_x\mathbf{B} \\ \boldsymbol{\partial y}\mathbf{K}_y\mathbf{B}\otimes\boldsymbol{x}\mathbf{K}_x\mathbf{B} \\ \boldsymbol{y}\mathbf{K}_y\mathbf{B}\otimes\boldsymbol{\partial x}\mathbf{K}_x\mathbf{B} \end{bmatrix}, \boldsymbol{b} &= \begin{bmatrix} z \\ -2\frac{s_1\sin(\psi)+s_2\cos(\psi)}{1-s_1^2-s_2^2} \\ -2\frac{s_1\cos(\psi)-s_2\sin(\psi)}{1-s_1^2-s_2^2} \end{bmatrix}. \end{aligned} \tag{19}$$

where (id_x, id_y) is the relative coordinate in the 7×7 control points, $\mathbf{M}_S$ and $\mathbf{M}_N$ can be obtained from Alg. 1, $Vec(\mathbf{M})$ denotes the ordered stock of columns of a matrix $\mathbf{M}$.

D. Manifold update strategy

We conceive a space-based sliding window filtering system that considers two absolute directions in the xoy plane rather than only one direction as in the time-based sliding window state estimation algorithms [5, 29].

As shown in the left subplot of Fig. 2, when estimating the active ground in the left, it is necessary to use the active

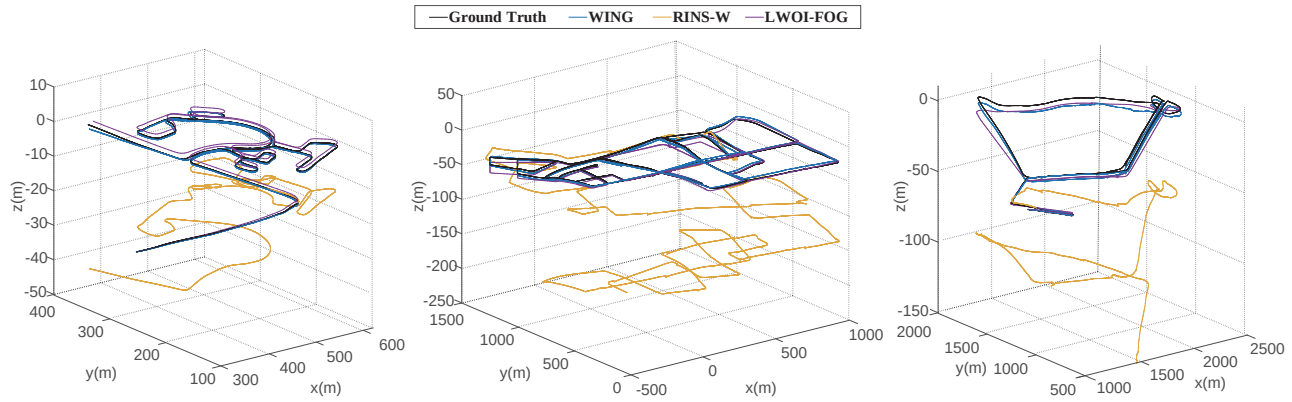

Fig. 3: Demonstration of different algorithms for pose estimation. From top to bottom, there are 3D estimated trajectories of *urban07*, *urban09*, and *urban17* of the KAIST Urban Dataset [15].

Algorithm 1: GET SELECT MATRIX

Input: input id_x, id_y
Output: $\boldsymbol{M}_S$, $\boldsymbol{M}_N$

InitCenMat $\leftarrow Zeros(16, 16)$
InitCenMat[3:7, 3:7] $\leftarrow Ones(4, 4)$
ActMat $\leftarrow$ InitCenMat[id_x:id_x+4, id_y:id_y+4]
ActVec $\leftarrow$ Vec(ActMat$^\top$)
IndexAct $\leftarrow$ Find(ActVec==1)
ActVecNegMat $\leftarrow Eyes(16)$ - diag(ActVec[:, 0])
$\boldsymbol{M}_N \leftarrow Zeros(4, 4)$
for *i = 0,1,...,15* **do**
  if *ActVec[i] == 1* **then**
    $\boldsymbol{M}_N$[i,:] $\leftarrow Zeros(1, 16)$
  else
    $\boldsymbol{M}_N$[i,:] $\leftarrow$ ActVecNegMat[i:i+1, :]
InitActMat $\leftarrow Zeros(4, 4)$
InitActMat[id_x:id_x+4, id_y:id_y+4] $\leftarrow Ones(4, 4)$
CenMat $\leftarrow$ InitActMat[3:7, 3:7]
CenVec $\leftarrow$ Vec(CenMat$^\top$)
IndexCen $\leftarrow$ Find(CenVec==1)
$\boldsymbol{M}_S \leftarrow Zeros(4, 4)$
if *len(IndexCen) > 0* **then**
  for *i = 0,1,...,len(IndexCen)* **do**
    $\boldsymbol{M}_S$[IndexAct[i], IndexCen[i]] $\leftarrow$ 1

poses in the active ground to determine the 4×4 active control points. However, it is also crucial to note that the poses belonging to all of the 7×7 grounds affect the 4×4 control points in the center. Therefore, we save all the pose trajectories of the 7×7 grounds and their corresponding 10×10 control points. Except for the active control points, other control points (the static control points) would not be updated. Whenever the ground robot is detected running over the boundary of each set ground grid, we use the ground manifold hypothesis to update the active poses within the active ground and its active control points.

In the right subplot of Fig. 2, after updating, we slide the ground grid in the direction through the boundary and margin the grounds (the marginalized grounds: the gray grids) and their corresponding pose trajectories (the marginalized poses: the gray curves) which are no longer belong to the current 7×7 grounds, as well as the control points (the marginalized control points: the gray circles) that do not determine the current central ground anymore.

V. Experiments

To better evaluate the proposed method, we compare estimation accuracy with other approaches and perform an ablation study 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. The proposed networks choose the Adam optimizer to minimize the loss function with an initial learning rate of 10^{-4}.

A. Pose Comparison

We select RINS-W [13], TLIO [21], LWOI [14] with FOG measurements (LWOI-FOG), and LWOI [14] with IMU's gyroscope measurements (LWOI-IMU) for comparison. For evaluation, we use the RMSE of absolute translation error (**ATE**) and rotation error (**ARE**) to represent the estimation accuracy:

- **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):= $\sqrt{\frac{1}{M}\sum_{i=1}^{M} \|\log(\widehat{\boldsymbol{q}}^* \otimes \boldsymbol{q})\|_2^2}$,

where $\boldsymbol{q}^*$ is the conjugate of the quaternion $\boldsymbol{q}$ and M is the number of the estimated poses.

The translation and rotation RMSE results of all competing algorithms are listed in the third to seventh rows of TABLE I. The results demonstrate that our proposed method WING can obtain a more accurate pose estimation than RINS-W [13], TLIO [21], and LWOI-IMU [14]. Moreover, although LWOI-FOG [14] utilizes FOG measurements, our method could compete with it. The 3D trajectories are drawn in Fig. 3, where TLIO [21] and LWOI-IMU [14] are not plotted as the divergent estimations. These results show the benefit of using ground manifold as well as neural bias and velocity estimation.

TABLE I: 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.	WING w.o. M-N.
urban07	ATE (m)	18.41	409.41	**5.17**	96.74	**3.90**	5.53	7.23
	ARE (deg.)	2.34	84.56	5.08	44.96	**1.22**	**1.25**	1.44
urban09	ATE (m)	214.88	2402.82	**19.31**	846.22	**18.73**	22.01	26.25
	ARE (deg.)	10.85	100.54	4.23	114.24	**1.71**	**1.91**	1.92
urban11	ATE (m)	113.90	156.29	**70.79**	4024.93	**88.88**	193.87	197.54
	ARE (deg.)	4.58	3.77	2.58	45.53	**1.77**	**2.04**	2.05
urban13	ATE (m)	449.29	624.22	**15.59**	425.82	**16.17**	16.42	16.56
	ARE (deg.)	28.37	86.17	2.51	59.723	**2.34**	**2.39**	2.41
urban15	ATE (m)	69.86	206.62	**8.08**	333.15	**9.81**	11.08	16.84
	ARE (deg.)	8.55	32.35	3.78	65.18	**1.30**	**1.31**	1.32
urban17	ATE (m)	86.10	1384.02	11.52	955.16	**7.82**	**9.84**	10.88
	ARE (deg.)	4.49	57.33	3.12	69.87	**0.53**	**0.54**	0.54

B. Ablation Study

In order to further illustrate the performance of modules in the proposed system, the complete system WING is compared with WING w.o. M (WING without manifold constraint) and WING w.o. M-N (WING without manifold constraint and neural velocity measurements). All RMSEs are listed in the seventh to ninth columns of TABLE I, and the uniaxial errors for rotation and translation are plotted in Fig. 4.

Compared with the system using raw encoder measurements, lateral and vertical constraints along with fixed covariance, it is more logical and practical to employ the triaxial velocity and covariance of network outputs in pose estimation for ground robots. Moreover, as shown in Fig. 4, the system with the manifold constraint has a smaller tilt and z-axis error because the manifold smooths the poses over a history window and provides additional position and attitude constraints. Besides, the neural velocity measurements improve the position estimation, especially the y-axis, as it can capture extra information like slipping or drifting.

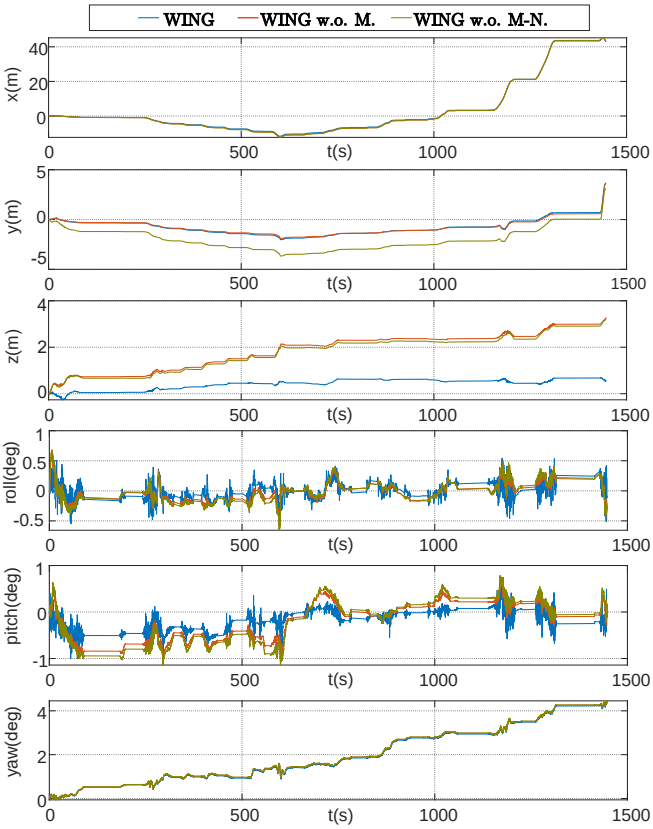

Fig. 4: Position and rotation errors in ablation study. From top to bottom, there are position errors in the three axes and rotation errors in the Euler angles (ZYX in the Tait-Bryan angles).

VI. Conclusion and Future Work

A. Conclusion

This work proposes a wheel-inertial odometry system that combines deep neural networks with ground manifold constraints within a space-based sliding window filtering framework. Since it is specifically designed for ground robots, we use a dual cubic B-spline surface to provide continuous manifold constraints. To take full advantage of IMU and wheel encoder, we design deep neural networks to reduce IMU biases, compensate for the measurement error of the wheel encoder and estimate the lateral slipping and vertical bouncing.

Experimental results show the proposed algorithm outperforms other learning-based methods. Besides, neural velocity measurements and continuous surface constraints are verified to be efficient for pose estimation.

B. 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, the covariance of added control points and the size of each grid in the B-spline surface are invariant. In the future, we could use a learning-based method to output the dynamic covariance and time-varying grid sizes, making the system more adaptive to match variant scenarios.

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] 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.

[4] 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.

[5] 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.

[6] 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.

[7] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, "Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam," *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[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] 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.

[17] 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.

[18] X. Niu, Y. Wu, and J. Kuang, "Wheel-ins: A wheel-mounted mems imu-based dead reckoning system," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 10, pp. 9814–9825, 2021.

[19] M. Brossard, A. Barrau, and S. Bonnabel, "Ai-imu dead-reckoning," *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 4, pp. 585–595, 2020.

[20] 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.

[21] 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

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] D. Chen, N. Wang, R. Xu, W. Xie, H. Bao, and G. Zhang, "Rnin-vio: Robust neural inertial navigation aided visual-inertial odometry in challenging scenes," in *2021 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 2021, pp. 275–283.

[29] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, "Openvins: A research platform for visual-inertial estimation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 4666–4672.