2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

VIVO: A Visual-Inertial-Velocity Odometry with Online Calibration in Challenging Condition

Fuzhang Han*, Shenhan Jia*, Jiyu Yu, Yufei Wei, Wenjun Huang, Yue Wang, Rong Xiong

***Abstract*— State estimation is a central component of autonomous navigation. To date, many methods presented have a disruptive potential for application, such as visual-inertial odometry (VIO), wheel and leg odometry (for short, body odometry). However, most of them are prone to fail in some challenging conditions like high-dynamic street scenes and sustain aggressive movements. To this end, in this paper, we present a novel visual-inertial-velocity odometry (VIVO) framework which incorporates velocity measurement provided by the proprioceptive sensing into the MSCKF-based VIO in a tightly coupled fashion. Furthermore, considering that the imprecise extrinsic parameters can severely undermine the state estimation performance, we hence perform VIVO along with online calibration of the body odometry's extrinsic parameters by adding them to the estimated state vector. The generic VIVO can be deployed for a broad spectrum of robot models ranging from wheeled robots to legged robots. Both simulation and real-world experiments are performed to extensively validate the robustness and accuracy of the proposed method in challenging scenarios using wheeled and legged robot models, respectively.**

I. Introduction

Over the last decade, the research on simultaneous localization and mapping (SLAM) with mobile robots has gained more and more importance. State estimation plays a critical role in the autonomous navigation of mobile robots, which can be achieved in many ways. The most representative one of these solutions is the visual-inertial navigation system (VINS). Notably, with the advent of consumer-grade cameras and IMUs, VINS has gained more attention and witnessed some particular eye-catching pieces of work.

However, in some challenging condition, such as search-and-rescue, robots are deployed for complex environments (*e.g.* dynamic street scene) and sustain extreme motion (*e.g.* high-frequency vibrations and high-velocity locomotion), as shown in Fig. 1. The moving objects in dynamic scenes lead to few reliable feature tracking, while the aggressive movements may result in image blur and less overlap between the adjacent frames. Alternatively, the extreme motion injects more noise into the IMU data compared to when it is static, thus introducing more errors into the state propagation due to first-order approximation and reducing convergence rate, especially with few image observations. The performance of these visual-inertial odometry (VIO) methods may suffer greatly from that.

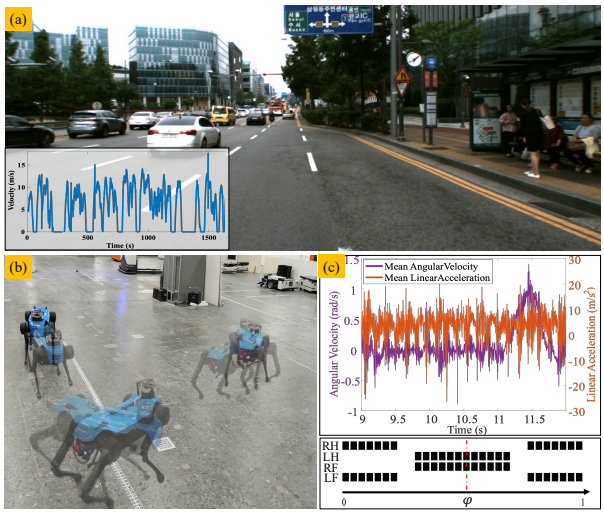


Fig. 1: The scenes in challenging condition. (a) Dynamic street scene. The scene with many moving objects in the complex urban dataset (kaist urban 38). The velocity of the car obtained from the wheel odometry is shown in the lower left corner. (b) High-dynamic movement of the quadruped robot. The raw data of the robot's gait pattern and IMU in (b) are presented in (c), which demonstrates the motion of the quadruped robot contains a full flight phase along with large linear accelerations and angular velocities.

Some studies try to incorporate proprioceptive perception, such as wheel odometry [1] and leg odometry (LO) [2], which can provide high-frequency velocity observation, into the VIO to improve the accuracy of the state estimation. To date, most of these methods can be broadly categorized into the Kalman filter-based method and smoothing approach.

The smoothing method can alleviate the influence of data noise through relinearization but with a high computational cost [3]. Some studies offer a compromise between accuracy and efficiency using the keyframe-based method [4] where some non-sequential past states are considered as keyframes and perform preintegration between each of them. Since the preintegration utilizes the first-order approximation, long intervals of keyframes will result in more noise that cannot be reduced by relinearization, while shortening the interval, which is more like a filter-based method, may fail to re-linearize in time. Thus, it is usually applied to offline applications or robots in smoothing motion. Although some quadruped platforms are deployed in the smoothing method [5] [6], they are still in slowing walking gait.

In comparison, filter-based algorithms enable fast inference due to no use of relinearization [7], but inevitably increase the sensitivity to noise. One of the most well-known

*Both authors contributed equally to this work.

All authors are with the State Key Laboratory of Industrial Control and Technology, Zhejiang University, Hangzhou, P.R. China. Yue Wang is the corresponding author wangyue@iipc.zju.edu.cn.

This work was supported by the National Nature Science Foundation of China under Grant 62373322, Zhejiang Provincial Natural Science Foundation of China under Grant No. LD24F030001.

works is MSF [8], where the state prediction is driven by IMU and regard other measurements (*e.g.* visual odometry (VO)) as the update. Despite enable fast deployment, this loosely coupled framework gives rise to the problem of accuracy and robustness since the single measurement like VO is greatly jeopardized during extreme motions to a point where may even result in algorithm failure. Alternatively, [9] fuses visual, inertial, and wheel odometry measurements in a tightly coupled fashion. However, it requires the wheel odometry to be preintegrated, which introduces additional linearization errors.

Moreover, since the filter-based methods do not contain a re-linearizing process, the performance of these algorithms may suffer greatly from the imprecise calibration parameters. Some methods [10] [11] that handle this problem in an offline fashion [12] ignore the error of calibration, yielding the offset in the linearization point, especially when the IMU is mounted far away from cameras.

Motivated by the discussion above, in this paper, we adopt a more direct approach which extend the VIO in MSCKF framework [13] to incorporate raw velocity measurements, which can be applied to those robots whose odometry information can be provided by a combination of kinematic model and proprioceptive sensors, as well as the online calibration of the extrinsic parameters between each odometry. Our method updates the velocity through the interpolation of original odometry data, which sidesteps the linearization noise of preintegration and introducing of high frequency states. In summary, the contributions of this paper are listed as follows:

- A lightweight and tightly coupled visual-inertial-velocity odometry (VIVO) is propsed, which performs robust and accurate state estimation even in the challenging motion and offers a marked advantage for the robots with limited computing resources.
- Online calibration is performed to avoid additional errors obtained in offline fashion, while also speeding up the deployment of algorithms.
- The algorithm is extensively evaluated in both simulation and real-world experiments, including trials using a high-dynamic quadruped robot and tens of kilometers traveling in the complex urban environment using the wheeled robot, which demonstrate the robustness and accuracy of our method in extreme motion where some algorithms even fail.

II. Problem Statement

Our work aims to track the 6 Degrees of Freedom (DoF) pose of the robot, together with estimating the extrinsic parameters between the velocity measurement and VIO module. To better describe our method, we provide some basic state and framework definitions in this section.

A. Frame Definitions and Transformation

The definition of each frame is illustrated in Fig. 2. Following typical VIO frame definitions, $\{C\}$ and $\{I\}$ are corresponding to the camera and IMU frame respectively,

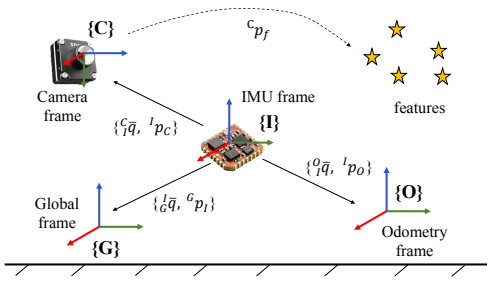


Fig. 2: Illustration of each frames.

while $\{G\}$ is a VIO reference frame whose origin coincides with the initial IMU position. $^{C}\boldsymbol{p}_f$ denotes the position of each feature to the camera frame. Furthermore, $\{O\}$ represents the wheel or leg odometry (for short, body odometry) frame provided by robot kinematic model.

As for the transformation of each frame, in the remainder of this paper, the unit quaternion $^{B}_{A}\bar{\boldsymbol{q}} = [\bar{\boldsymbol{k}}\sin(\theta/2)\ \cos(\theta/2)]$ represents the rotation with θ angle around the axis $\bar{\boldsymbol{k}}$ and is corresponding to the the 3×3 rotation matrix $^{B}_{A}\mathbf{R} \in \mathcal{SO}(3)$. The position of frame **B** in frame **A** is expressed as $^{A}\mathbf{p}_B \in \mathbb{R}^3$. In particular, $\{^{B}_{A}\bar{\boldsymbol{q}}, ^{A}\mathbf{p}_B\}$ represents the rotation and translation from frame **B** to **A**. A time-dependent vector **a** acquired at time t_k is shortened as $\mathbf{a}_k$

B. State Vector

The states estimated in our system containing current IMU states $\mathbf{x}_I$, the body odometry extrinsic parameters $\mathbf{x}_O$ and a sliding window (past m images) of cloned IMU poses $\mathbf{x}_{cl}$. At time t_k, the state is written as:

$$\mathbf{x}_k = [\boldsymbol{x}_{I,k}^\top \quad \boldsymbol{x}_O^\top \quad {}^{G}\mathbf{p}_f^\top \quad \boldsymbol{x}_{cl,k}^\top]^\top \tag{1}$$

$$\mathbf{x}_{I,k} = [{}^{I}_{G}\bar{\boldsymbol{q}}^\top \quad \boldsymbol{b}_{g,k}^\top \quad {}^{G}\boldsymbol{v}_{I,k}^\top \quad \boldsymbol{b}_{a,k}^\top \quad {}^{G}\boldsymbol{p}_{I,k}^\top]^\top \tag{2}$$

$$\mathbf{x}_O = [{}^{O}_{I}\bar{\boldsymbol{q}}^\top \quad {}^{I}\boldsymbol{p}_O^\top]^\top \tag{3}$$

$$\mathbf{x}_{cl,k} = [{}^{I}_{G}\bar{\boldsymbol{q}}_k^\top \ {}^{G}\boldsymbol{p}_{I,k}^\top \ {}^{G}\boldsymbol{v}_{I,k}^\top \ \dots \ {}^{I}_{G}\bar{\boldsymbol{q}}_\ell^\top \ {}^{G}\boldsymbol{p}_{I,\ell}^\top \ {}^{G}\boldsymbol{v}_{I,\ell}^\top]^\top \tag{4}$$

where ℓ is $k - m + 1$. $^{G}\boldsymbol{v}_I$ is the velocity of IMU in global frame; $\boldsymbol{b}_{g,k}$ and $\boldsymbol{b}_{a,k}$ represent the gyroscope and accelerometer biases respectively; $\mathbf{x}_O = \{^{O}_{I}\bar{\boldsymbol{q}},\ ^{I}\boldsymbol{p}_O\}$ denotes the transformation from body odometry frame $\{O\}$ to IMU frame $\{I\}$. Besides, $\{^{I}_{G}\bar{\boldsymbol{q}}_i,\ ^{G}\boldsymbol{p}_{I,i}\}$ $(i = \ell, \dots, k)$ are the cloned IMU poses at time t_i. $^{G}\mathbf{p}_f^\top$ is the position of visual feature point in global frame.

In this paper, we define $\boldsymbol{x} = \hat{\boldsymbol{x}} \oplus \tilde{\boldsymbol{x}}$, where $\hat{\boldsymbol{x}}$ and $\tilde{\boldsymbol{x}}$ are the estimation and error of state $\boldsymbol{x}$, respectively. $\oplus$ maps the error state vector to its corresponding manifold [14]. Since the error quaternion $\delta\bar{\boldsymbol{q}}$ can be approximately writtern as $\delta\bar{\boldsymbol{q}} = [\frac{1}{2}\tilde{\boldsymbol{\theta}}^\top \ 1]^\top$ when the error is small, we use the 3×1 angle-error vector $\tilde{\boldsymbol{\theta}}$ to represent the quaternion error in the error state.

III. VIVO Framework Description

As illustrated in Fig. 3, the measurements in our system mainly consist of three parts: body odometry, camera measurement (feature observation), and IMU measurement.

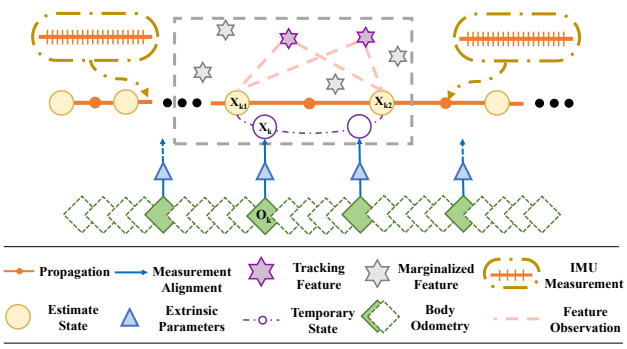


Fig. 3: Framework of the proposed VIVO.

Similar to the standard MSCKF framework, the state is propagated forward using IMU measurements and updated by the other two measurements. Subsequently, the detail of the aforementioned process is described in the following sections.

A. Propagation

Since measurements of IMU are affected by bias ($\boldsymbol{b}$) and zero-mean Gaussian noise ($\boldsymbol{n}$) [15], they can be modeled as:

$$\boldsymbol{\omega}_m(t) = {}^{I}\boldsymbol{\omega}(t) + \boldsymbol{b}_g(t) + \boldsymbol{n}_g(t) \tag{5}$$

$$\boldsymbol{a}_m(t) = {}^{I(t)}_{G}\boldsymbol{R}({}^{G}\boldsymbol{a}_I(t) + {}^{G}\boldsymbol{g}) + \boldsymbol{b}_a(t) + \boldsymbol{n}_a(t) \tag{6}$$

where $\boldsymbol{\omega}_m(t)$ and $\boldsymbol{a}_m(t)$ are the raw measurement data. ${}^{I}\boldsymbol{\omega}(t)$ is the angular velocity of IMU in local frame $\{I\}$. ${}^{G}\boldsymbol{g}$ and ${}^{G}\boldsymbol{a}_I(t)$ are the acceleration of gravity and IMU expressed in global frame. The IMU kinematics [16] can be described as:

$$\begin{aligned} {}^{I}_{G}\dot{\bar{\boldsymbol{q}}}(t) &= \frac{1}{2}\boldsymbol{\Omega}({}^{I}\boldsymbol{\omega}(t)){}^{I}_{G}\bar{\boldsymbol{q}}(t) \\ {}^{G}\dot{\boldsymbol{p}}_I(t) = {}^{G}\boldsymbol{v}_I(t), &\quad {}^{G}\dot{\boldsymbol{v}}_I(t) = {}^{G}\boldsymbol{a}(t) \\ \dot{\boldsymbol{b}}_g(t) = \boldsymbol{n}_{wg}(t), &\quad \dot{\boldsymbol{b}}_a(t) = \boldsymbol{n}_{wa}(t) \end{aligned} \tag{7}$$

where $\boldsymbol{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} -\lfloor\boldsymbol{\omega}\times\rfloor & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^{\top} & 0 \end{bmatrix}$ and $\lfloor\boldsymbol{\omega}\times\rfloor$ is the skew-symmetric matrix. To propagate the convariance matrix from time t_k to t_{k+1}, we have the generation format of linearized discrete-time model, following [17] as:

$$\tilde{\boldsymbol{x}}_{k+1} = \boldsymbol{\Phi}_k\tilde{\boldsymbol{x}}_k + \boldsymbol{w}_k \tag{8}$$

$$\boldsymbol{\Phi}_k = \begin{bmatrix} \boldsymbol{\Phi}_{I_k} & \boldsymbol{0}_{15\times 9} \\ \boldsymbol{0}_{9\times 15} & \boldsymbol{\Phi}_{Other} \end{bmatrix} \tag{9}$$

where $\boldsymbol{\Phi}_k$ is linearized system state transition matrix. $\boldsymbol{\Phi}_{I_k}$ represent the state transition matrix for $\boldsymbol{x}_I$. $\boldsymbol{\Phi}_{Other} = \boldsymbol{I}_9$ represent the state transition matrix for $\boldsymbol{x}_O$ and ${}^{G}\mathbf{p}_f^{\top}$ respectively. The format of the $\boldsymbol{\Phi}_k$ is commonplace in the MSCKF-based algorithm, we hence not to describe here. Readers can refer to (44) in [18].

Having the IMU propagation model, we can predict the state vector and propagate the covariance as[1]:

$$\boldsymbol{P}_{k+1|k} = \boldsymbol{\Phi}_k\boldsymbol{P}_{k|k}\boldsymbol{\Phi}_k^{\top} + \boldsymbol{Q}_{d,k} \tag{10}$$

[1]Throughout this paper, the subscript $\ell|j$ refers to estimate of a quantity at time ℓ, after all measurements up to time j have been processed. $\boldsymbol{I}_n$ and $\boldsymbol{O}_n$ are the $n \times n$ identity and zero matrices.

where $\boldsymbol{Q}_{d,k}$ is the discrete-time system noise covariance as described in [19]. The stochastic cloning [20] is performed after propagation to probabilistically augment $\boldsymbol{x}_{cl,k}$ and covariance matrix with the current pose estimate. In what follows, we discuss updating the state and its covariance with the camera measurement model.

B. Update

Since the linearized residual between the actual and expected feature measurement:

$$\tilde{\boldsymbol{r}}_k = \boldsymbol{H}_k\tilde{\boldsymbol{x}}_k + \boldsymbol{n}_k \tag{11}$$

where $\boldsymbol{H}_k$ is the measurement Jacobian matrix. Thereby we can have the update once the jacobian matrix of each measurement is computed.

Camera Measurement Model. Refer to [17], we have the camera measurement model as:

$$\boldsymbol{z}_k = \frac{1}{{}^{C_k}z_f}\begin{bmatrix} {}^{C_k}x_f \\ {}^{C_k}y_f \end{bmatrix} + \boldsymbol{n}_{f_k} \tag{12}$$

where ${}^{C_k}\boldsymbol{p}_f = \begin{bmatrix} {}^{C_k}x_f & {}^{C_k}y_f & {}^{C_k}z_f \end{bmatrix}$ represents the feature observed at time k. $\boldsymbol{n}_{f_k}$ is the zero-mean Gaussian noise of measurements. Besides we have:

$${}^{C_k}\boldsymbol{p}_f = {}^{C}_{I}\boldsymbol{R}^{I_k}_{G}\boldsymbol{R}({}^{G}\boldsymbol{p}_f - {}^{G}\boldsymbol{p}_{I_k}) + {}^{C}\boldsymbol{p}_I \tag{13}$$

where $\{{}^{C}_{I}\boldsymbol{R}, {}^{C}\boldsymbol{p}_I\}$ is the rotation and translation between the camera and IMU, which is calibrated in offline fashion [21]. ${}^{G}\boldsymbol{p}_f$ and ${}^{G}\boldsymbol{p}_{I_k}$ are the position of feature points and IMU at time k in global frame. Then we compute the camera measurement Jacobian as:

$$\boldsymbol{H}_c = \frac{\partial\tilde{\boldsymbol{z}}_c}{\partial\tilde{\boldsymbol{x}}} = \frac{\partial\tilde{\boldsymbol{z}}_c}{\partial{}^{C}\tilde{\boldsymbol{p}}_f}\frac{\partial{}^{C}\tilde{\boldsymbol{p}}_f}{\partial\tilde{\boldsymbol{x}}} = \boldsymbol{\Lambda}\begin{bmatrix} \boldsymbol{H}_r \\ \boldsymbol{H}_b \end{bmatrix}\frac{\partial{}^{C}\tilde{\boldsymbol{p}}_f}{\partial\tilde{\boldsymbol{x}}} \tag{14}$$

where $\boldsymbol{\Lambda}$ represents the measurement selection matrix [15]. $\boldsymbol{H}_r$ and $\boldsymbol{H}_b$ denotes the range and bearing measurement Jacobians with respect to ${}^{C}\boldsymbol{p}_f$. The detailed derivations of these Jacobians when using different sensors are referred to [22] and

$$\frac{\partial{}^{C}\tilde{\mathbf{p}}_f}{\partial\tilde{\mathbf{x}}} = {}^{C}_{I}\hat{\mathbf{R}}^{I_k}_{G}\hat{\mathbf{R}}[\lfloor({}^{G}\hat{\mathbf{p}}_f - {}^{G}\hat{\mathbf{p}}_{I_k})\times\rfloor{}^{I_k}_{G}\hat{\mathbf{R}}^{\top}\ \mathbf{0}_9\ -\mathbf{I}_3\ \mathbf{0}_6\ \mathbf{I}_3] \tag{15}$$

Measurement alignment. As the robot typically provides body odometry at higher rates than the camera, the time of each measurement may not be aligned, as shown in Fig. 3. Rather than in preintegration fashion, which may cause large linearization error, we adopt an on-manifold interpolation model and the body odometry measurements will be used to update the neighbor states.

Take linear velocity as an example, assume that the velocity observation is received at t_k, whose neighbor timestamp of cloned poses in the sliding windows are t_{k1} and t_{k2}. Then we have the defination of scale factor λ_k as :

$$\lambda_k = \frac{t_k - t_{k1}}{t_{k2} - t_{k1}} \tag{16}$$

According to (4), we have $\boldsymbol{x}_{cl,k1}$ and $\boldsymbol{x}_{cl,k2}$ corresponding to the cloned pose at t_{k1} and t_{k2}. Then we will have the

states at t_k, $\boldsymbol{x}_k$. Since the jacobian corresponding to some part of the temporary state is zero , we only perform the interpolation to the non-zero parts: ${}^{I_k}_{G}\boldsymbol{R}$ and ${}^{G}\boldsymbol{v}_{I_k}$, as :

$$ {}^{I_k}_{G}\boldsymbol{R} = \mathrm{Exp}(\,\lambda_k\,\mathrm{Log}(\,{}^{I_{k2}}_{G}\boldsymbol{R}\,{}^{I_{k1}}_{G}\boldsymbol{R}^{\top}\,)\,)\,{}^{I_{k1}}_{G}\boldsymbol{R} \tag{17} $$

$$ {}^{G}\boldsymbol{v}_{I_k} = (1-\lambda_k)\,{}^{G}\boldsymbol{v}_{I_{k1}} + \lambda_k\,{}^{G}\boldsymbol{v}_{I_{k2}} \tag{18} $$

where $\mathrm{Exp}(\cdot)$ maps a rotation matrix to a vector in $\mathbb{R}^3$ and $\mathrm{Log}(\cdot)$ is the inverse operation [23].

Odometry measurement model. Considering the velocity measurement we have obtained in section IV, we denote them by Γ:

$$ \boldsymbol{\Gamma}(t) = \begin{bmatrix} {}^{O}\boldsymbol{v}(t) \\ {}^{O}\boldsymbol{\omega}(t) \end{bmatrix} \tag{19} $$

where ${}^{O}\boldsymbol{v}$ and ${}^{O}\boldsymbol{\omega}$ are the linear and angular velocity of the body in body frame $\{O\}$. Considering the extrinsic parameters between $\{O\}$ and $\{I\}$, we have the below equation:

$$ {}^{G}\boldsymbol{p}_O = {}^{G}\boldsymbol{p}_I + {}^{I}_{G}\boldsymbol{R}^{\top}\,{}^{I}\boldsymbol{p}_O \tag{20} $$

By performing on-manifold derivation on the both sides of the equation (20), we can obtain the odometry measurement model as:

$$ {}^{O}\boldsymbol{\omega}_m = {}^{O}_{I}\boldsymbol{R}\,{}^{I}\boldsymbol{\omega} + \boldsymbol{n}_{og} \tag{21} $$

$$ {}^{O}\boldsymbol{v}_m = {}^{O}_{I}\boldsymbol{R}\,{}^{I}_{G}\boldsymbol{R}({}^{G}\boldsymbol{v}_I + {}^{I}_{G}\boldsymbol{R}^{\top}\lfloor {}^{I}\boldsymbol{\omega}\times\rfloor\,{}^{I}\boldsymbol{p}_O) + \boldsymbol{n}_{ov} \tag{22} $$

where ${}^{O}\boldsymbol{\omega}_m$ and ${}^{O}\boldsymbol{v}_m$ are linear and angular velocity measurement in $\{O\}$. $\boldsymbol{n}_{og}$ and $\boldsymbol{n}_{ov}$ are the zero-mean white Gaussian measurement noises. Then based on the chain rule of derivation, we have the Jacobian corresponding to the cloned poses as :

$$ \frac{\partial\tilde{\boldsymbol{\Gamma}}(t_k)}{\partial\tilde{\boldsymbol{x}}_{k1}} = \frac{\partial\tilde{\boldsymbol{\Gamma}}(t_k)}{\partial\tilde{\boldsymbol{x}}_{k}}\,\frac{\partial\tilde{\boldsymbol{x}}_{k}}{\partial\tilde{\boldsymbol{x}}_{k1}} \tag{23} $$

$$ \frac{\partial\tilde{\boldsymbol{\Gamma}}(t_k)}{\partial\tilde{\boldsymbol{x}}_{k2}} = \frac{\partial\tilde{\boldsymbol{\Gamma}}(t_k)}{\partial\tilde{\boldsymbol{x}}_{k}}\,\frac{\partial\tilde{\boldsymbol{x}}_{k}}{\partial\tilde{\boldsymbol{x}}_{k2}} \tag{24} $$

Hence combining (22) and (22), the jacobian of body odometry measurement is :

$$ \boldsymbol{V}_k = \frac{\partial^{O}\tilde{\boldsymbol{v}}_k}{\partial\tilde{\boldsymbol{x}}_k} = [\boldsymbol{\varphi}_1\ \boldsymbol{\varphi}_2\ \boldsymbol{\varphi}_3\ \mathbf{0}_3\ \mathbf{0}_3\ \boldsymbol{\varphi}_4\ \boldsymbol{\varphi}_5\ \mathbf{0}_3] \tag{25} $$

$$ \boldsymbol{W}_k = \frac{\partial^{O}\tilde{\boldsymbol{\omega}}_k}{\partial\tilde{\boldsymbol{x}}_k} = [\mathbf{0}_3\ \boldsymbol{\varepsilon}_1\ \mathbf{0}_3\ \mathbf{0}_3\ \mathbf{0}_3\ \mathbf{0}_3\ \boldsymbol{\varepsilon}_2\ \mathbf{0}_3] \tag{26} $$

where:

$$ \boldsymbol{\varphi}_1 = {}^{O}_{I}\hat{\boldsymbol{R}}\lfloor {}^{I_k}_{G}\hat{\boldsymbol{R}}\,{}^{G}\hat{\boldsymbol{v}}_{I_k}\times\rfloor \tag{27} $$

$$ \boldsymbol{\varphi}_2 = {}^{O}_{I}\hat{\boldsymbol{R}}\lfloor {}^{I}\hat{\boldsymbol{p}}_O\times\rfloor \tag{28} $$

$$ \boldsymbol{\varphi}_3 = {}^{O}_{I}\hat{\boldsymbol{R}}\,{}^{I_k}_{G}\hat{\boldsymbol{R}} \tag{29} $$

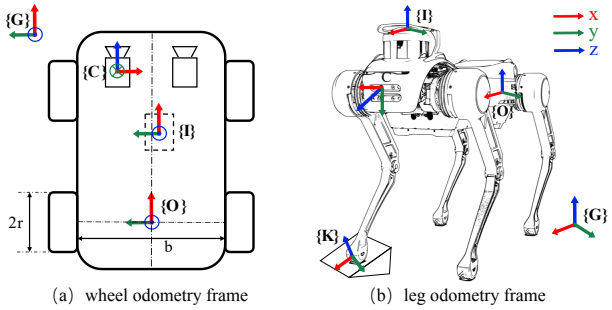
$$ \boldsymbol{\varphi}_4 = \lfloor {}^{O}_{I}\hat{\boldsymbol{R}}\,{}^{I_k}_{G}\hat{\boldsymbol{R}}\,{}^{G}\hat{\boldsymbol{v}}_{I_k}\times\rfloor + \lfloor {}^{I}\hat{\boldsymbol{p}}_O\times\rfloor\lfloor {}^{O}_{I}\hat{\boldsymbol{R}}^{I_k}\hat{\boldsymbol{\omega}}\times\rfloor \tag{30} $$

$$ \boldsymbol{\varphi}_5 = -\lfloor {}^{O}_{I}\hat{\boldsymbol{R}}^{I_k}\hat{\boldsymbol{\omega}}\times\rfloor \tag{31} $$

$$ \boldsymbol{\varepsilon}_1 = -{}^{O}_{I}\hat{\boldsymbol{R}} \tag{32} $$

$$ \boldsymbol{\varepsilon}_2 = \lfloor {}^{O}_{I}\hat{\boldsymbol{R}}^{I_k}\hat{\boldsymbol{\omega}}\times\rfloor \tag{33} $$

As for the remain part in the chain, $\frac{\partial\tilde{\boldsymbol{x}}_k}{\partial\tilde{\boldsymbol{x}}_{k1}}$ and $\frac{\partial\tilde{\boldsymbol{x}}_k}{\partial\tilde{\boldsymbol{x}}_{k2}}$, they are same as the deductions (eq. (30) and (31)) in [19]. To keep our presentation concise, we prefer not to write them here.



(a) wheel odometry frame (b) leg odometry frame

Fig. 4: Odometry frame for body odometry measurement. (a) is the top view of the wheel odometry frame and (b) is the side view of the leg odometry. The coordinate frame is color-coded with red(x). green(y) and blue(z) zxes, repsectively.

IV. Odometry Measurement Model

With the general odometry model, in this section, we provide the example of deployments on different robot models. Towards this goal, we give an overview of two representative odometry models, the wheel odometry model, and the leg odometry model, in this section. Fig. 4 shows the frame definition for each body frame.

A. Wheel Odometry Model

For the wheeled robot driven by two differential (left and right) wheels which are commonplace in ground vehicles, the local angular rate readings provided by the encoder is:

$$ \dot{\theta}_m = \dot{\theta} + n_{\dot{\theta}} \tag{34} $$

where $n_{\dot{\theta}}$ is zero-mean white Gaussian noises and $\dot{\theta}$ is the actual angular velocities. Thus, refer to the kinematic model of the wheeled robot, the wheel odometry can be often be modeled as:

$$ {}^{O}\boldsymbol{\omega} = (\dot{\theta}_r r_r - \dot{\theta}_l r_l)/b \tag{35} $$

$$ {}^{O}\boldsymbol{v} = (\dot{\theta}_r r_r + \dot{\theta}_l r_l)/2 \tag{36} $$

where r and b are the wheel radii and the baselink length respectively, as shown in Fig. 4 (a). The subscript l and r are refer to left and right.

B. Leg Odometry Model

As shown in Fig. 4 (b), $\{K_\ell\}$ is the frame of the leg ℓ ($\ell \in \{LF, RF, LH, RH\}$). Frame $\{K_\ell\}$ is located at the end of each leg, $\{O\}$ is located at the geometry center of the body and the camera optical frame $\{C\}$ is rigidly attached to the head of the quadruped.

For each leg ℓ, we define the contact status as $s_k^\ell \in \{0, 1\}$, where 1 represents the valid contact for foot ℓ at time t_k. The contact condition of each leg can be derived following [24]. Then we can estimate the velocity of the body from the contacted leg ℓ as:

$$ \tilde{\boldsymbol{v}}_k^\ell = -\mathrm{J}(\tilde{\boldsymbol{\theta}}_k)\dot{\tilde{\boldsymbol{\theta}}}_k - \boldsymbol{\omega}\times\mathrm{fk}(\tilde{\boldsymbol{\theta}}_k) + \boldsymbol{\eta}^v \tag{37} $$

where $\mathrm{fk}(\cdot)$ and $\mathrm{J}(\cdot)$ are the forward kinematics function and its Jacobian respectively. $\tilde{\boldsymbol{\theta}}_k$ and $\dot{\tilde{\boldsymbol{\theta}}}_k$ represents the sensed

TABLE I: Relative Pose Error (RPE) of different motion distance (degree/meter).

Algorithm	Short Distance			Long Distance		
	50m	100m	200m	1800m	5400m	9100m
Open VINS	0.152/0.093	0.188/0.145	0.167/0.193	0.104/0.364	0.197/0.713	0.268/0.989
true& w.cal	**0.127/0.031**	**0.149/0.057**	**0.133/0.063**	**0.074/0.215**	**0.131/0.382**	**0.166/0.510**
bad& w.cal	0.127/0.032	0.149/0.065	0.110/0.059	0.075/0.216	0.132/0.385	0.168/0.513
bad& wo.cal	1.456/0.748	3.515/2.160	5.594/3.02	-/-	-/-	-/-

joint positions and velocities. $\boldsymbol{\eta}^v$ represents all the efforts of noise [25].

Considering multiple legs in contact at the same time, we regard the contact probabilities as weighted and have the average of body velocity as:

$$\tilde{\boldsymbol{v}}_k = \frac{\sum_{\ell \in C} P_k(s_\ell = 1 | \boldsymbol{f}_\ell^k) \tilde{\boldsymbol{v}}_k^\ell}{\sum_{\ell \in C} P_k(s_\ell = 1 | \boldsymbol{f}_\ell^k)} + \boldsymbol{\eta}^v \tag{38}$$

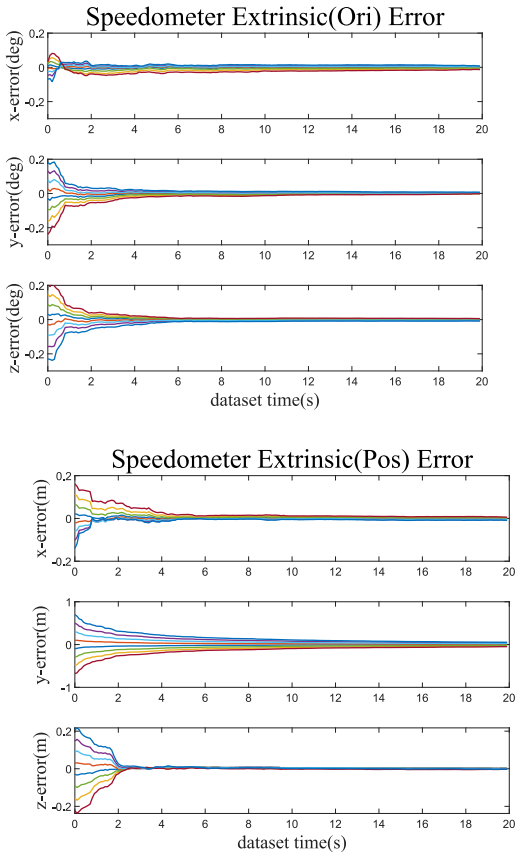


Fig. 5: Simulation calibration results. Each colors denote runs with different initial guess.

Following [26], the adaptive covariance $\boldsymbol{P}_k^v$ can be regard as a combination of three parts: sensor noise, inter-leg velovity coviance $\boldsymbol{D}_k$ and force discontinuities Δf which are caused by impacts. Assume that there are c_k contact legs at time t_k, the final covariance for velocity measurement is:

$$\boldsymbol{P}_k^v = \boldsymbol{P}_0^v + [\frac{1}{2}(\boldsymbol{D}_k + \boldsymbol{I}_3 \frac{\Delta f}{\alpha})]^2 \tag{39}$$

where α is a normalization factor, empirically determined.

TABLE II: Simulation parameters

Parameter	Value	Parameter	Value
Cam Freq. (hz)	10	IMU Freq. (hz)	400
Num. Feats Per Camera	100	Num. SLAM feats	25
Num. Base Cam. Clones	11	Feat. Rep.	GLOBAL
Gyro. White Noise	1.6968e-04	Gyro. Rand. Walk	1.9393e-05
Accel. White Noise	2.0000e-3	Accel. Rand. Walk	3.0000e-3
Pixel Noise	1	Vel. White Noise	1.0e-03
Speedometer. Ext(Ori). Ptrb	1.0e-2	Speedometer. Ext(Pos). Ptrb	1.0e-2

Based on (10) (38) and (39), we can calculate the Kalman gain $\boldsymbol{K}_k$ as:

$$\boldsymbol{K}_k = \boldsymbol{P}_k \boldsymbol{H}^\top (\boldsymbol{H} \boldsymbol{P}_k \boldsymbol{H}^\top + \boldsymbol{P}_k^v)^{-1} \tag{40}$$

where $\boldsymbol{H}$ is the Jacobian of the observaton function. Since $z_k = \tilde{\boldsymbol{v}}_k$, $\boldsymbol{H}$ here is a selector matrix for the linear velocity substate.

V. Experimental Results

A. simulation experiments

To fully evaluate the proposed VIVO algorithm, we deploy simulation experiments in OpenVINS [27], which provides both simulation and evaluation utilities[2]. We interpolate parametric trajectories with B-splines, and analytically derivative them to obtain the ground truth of velocity, acceleration, and angular velocity. Furthermore, we manually inject the Gaussian noise to the ground truth and regard it as the measurement of body odometry. The detailed configuration of the simulation experiment is shown in Table II.

To demonstrate the robustness of the online calibration algorithm, we simulated the trajectory and recorded the calibration results. As shown in Fig 5, we perform online calibration for eight different runs with varying initial extrinsic and all of them quickly (within 10 seconds) converge to the ground truth.

To validate the influence of the calibration on the accuracy of the algorithm, we take OpenVINS as a benchmark and test in different motion distance trajectory. Each distance trajectory has experimented with three different settings. The 'true' or 'bad' represents initialized with good or bad extrinsic parameters. We regard the prior distributions specified in Table II as "true" initial values and manually perturb these parameters as "bad" initial values. 'w. Cal' or 'wo. Cal' denotes with or without online calibration. The results are shown in Table I. Note that the estimators with substantially corrupted extrinsic parameters always diverge

[2]All of the simulation experiments are run on the computing platform, which is an AMD Ryzen7-4800@4.20GHz and 16-GB RAM.

TABLE III: Jueying sensors configurations

Sensors	Model	Frequency	Specs
IMU	Xsens	200Hz	Init Bias: $0.2^\circ/s \vert 5mg$ Bias Stab: $10^\circ/h \vert 15mg$
Stereo Camera	Realsense D435i	30Hz	Resolution: 640 $\times$ 480 px
Encoder	Jueying Robot	1kHz	Resolution: $< 0.025^\circ$
Torque	Jueying Robot	1kHz	Resolution: $< 0.1N \cdot m$

TABLE IV: Dataset overview, including max acceleration , for short max acc. (m/s^2) and max angular velocity, for short max angv. (rad/s) given by IMU.)

Scene	Max acc (x/y/z).	Max angv. (x/y/z)
urban38	15.13/ 15.22 / 19.11	0.48 / 0.44 / 0.37
urban39	14.67/ 13.20 / 16.06	0.60 / 0.34 / 0.38
indoor	80.73/ 67.96 / 42.54	3.37 / 2.51 / 4.35
outdoor	1.19/ 69.33 / 37.56	1.25 / 0.20 / 9.70

or fail without online calibration, we hence not record the result. As summarized in the table, the online calibration improves the accuracy even with poor initial values.

B. Real-World Experiment

To further evaluate the proposed VIVO, we migrate from the ideal simulation environment to the real application scene in real-world dataset and validate the accuracy and robustness of wheeled and legged robots, respectively.

Since there is no open source visual-inertial dataset containing leg odometry, we collect the dataset from the Jueying quadruped robot [28] which is equipped with stereo camera and can provide body odometry in 200 Hz frequency (see Table III for the specifications[3]). In particular, we add an additional IMU on the top of head for the requirement of the dual IMU designed for legged robot as shown in Fig. 7c.

To furnish the reader with further insight into the challenging characteristics of the dataset, we list the max acceleration and angular velocity of each dataset as shown in table IV.

1) KAIST urban dataset: For wheeled robots, we evaluate the proposed VIVO on an open-sourced real-world dataset: *KAIST urban38* and *KAIST urban39*, which are collected in the urban area with IMU, stereo camera, and wheel encoder measurements for estimation, and high-precision groundtruth is provided [29].

Specifically, we assume ground vehicles move on an appropriate 2D plane in this experiment, and there is no slippage between wheels and ground. The VIVO and Open-VINS are deployed with 15 clones and 100 features along with online calibration of IMU-camera extrinsic parameters.

The final trajectories are shown in Fig. 6, while the absolute trajectory error (ATE) of position and orientation of each algorithm are shown in Table V. It is evident that the proposed VIVO outperforms Open VINS and MSF on *urban38* and *urban39*.

[3]Although Jueying is equiped with 3D LiDAR, we only deploy the LIDAR to collect ground truth. So LiDAR is not listed in the table.

TABLE V: ATE of position and orientation error comparison result in *KAIST* dataset (meter/degree)

Algorithm	Open-VINS	MSF	VIVO
urban38	25.037 / 2.600	18.028 / 3.136	**11.966 /1.764**
urban39	15.307 / 2.247	13.118 / 3.734	**11.022/2.181**

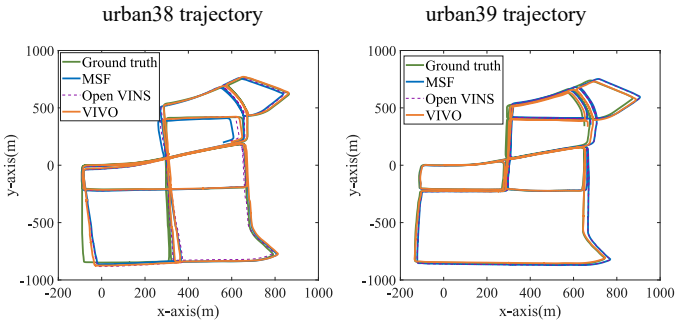


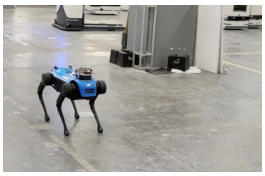
Fig. 6: *urban38* (left) and *urban39* (right) experiment results of VIVO (orange), MSF (blue), Open VINS (purple dotted line) and groundtruth (green).

2) Jueying dataset: To better evaluate our method, we compare it to other state-of-the-art VINS approaches including ORB-SLAM3 [30] (stereo inertial version) and Open-VINS, as well as the latest multi-sensor state estimator for legged robot, pronto [26]. Note that since our method does not contain the back end part, we only compare the results from tracking with different methods. Importantly, to demonstrate the advantages of our method in a high-dynamic scene, the quadruped robot moves at a high-speed trot-running gait [31] with a maximum speed of 3 m/s.

Some experiments are held indoors for evaluating the robustness and accuracy of our algorithm in challenging episodes (i.e. large external disturbances, high-dynamic locomotion), while others are held outdoors in order to access the performance of pose estimation while traversing through complex, dynamic outdoor terrain. Parts of experiment scenes are shown in Fig. 7a and Fig. 7b. The robot is tracked by motion capture system (6D pose) and Velodyne VLP-16 3D LiDAR separately for indoor and outdoor experiments to generate ground truth.

Indoor experiments. For the indoor experiment, we make comparisons under three different trajectories: pure linear line, pure rotation and random around. The result is summarized in Table VI. Bolded values denote the smallest error and "failed" means the drift is too large. "w. calib" and "wo. calib" represent with and without online calibration of the body odometry. For the algorithm without online calibration, we leverage Kalibr to obtain the extrinsic parameters between the leg odometry and vision odometry. As shown in Table VI, all algorithms fail when the robots turn around (pure rotation and random around) due to the rapid change of the image except for our method. As for the pure linear line, the view of camera remains almost the same which is friendly to the vision algorithm. However in such condition the online calibration of the body odometry extrinsic fails for the existence of degeneration condition. Thus the result

(a) Outdoor scene



(b) Indoor scene

(c) Sensor configurations

(d) Image blur

Fig. 7: Jueying experiments.

TABLE VI: RMSE of Absolute Trajectory Error (APE) in translation comparison between various method (meter)

Algorithm	Pure linear line	Pure rotation	Random around
VIVO w. calib	0.1306	**0.0931**	**0.8441**
VIVO wo. calib	**0.1046**	0.1766	1.1523
Open-VINS	0.1127	failed	failed
ORBSLAM3	1.3050	failed	failed
pronto	0.1883	failed	failed

without online calibration is better than the method with online calibration.

Outdoor experiments. The outdoor experiment are conducted by 12-DoF Jueying quadruped at a trot-running gait. To generate the ground truth, we perform the tightly coupled lidar inertial odometry, LIO-SAM [32], which includes loop closures at the back end and uses data collected by Jueying's Velodyne VLP-16 3D LiDAR at the 20 Hz frequency.

This 130.5 m long dataset contains climbing up and down slopes and stairs. Challenging situations include intermittent motion with fast forward and sharp turns, featureless areas and, foot slips caused by climbing stairs.

We compare our VIVO with OpenVINS, ORB SlAM3 and Pronto. The result is shown in Fig. 8. Note that we simply draw the trajectory of ORB SLAM3 before it fails. The root mean squared error (RMSE) of absolute position error (APE) of each algorithm are (unit: meter): VIVO 3.802, OpenVINS 4.819, Pronto 6.644. It is evident the proposed method outperforms other methods and the APE for VIVO is 20.75% and 43.8% lower compared to the OpenVINS and Pronto, respectively. For the limited field of view of the camera and long distance between camera and IMU, the performance of offline camera-IMU calibration methods are poor. Thus the algorithm containing online calibration, OpenVINS, works better than ORBSlAM3.

C. Timing

To furnish the reader with further insight into the efficiency of our algorithm, the computation time of each part in the proposed algorithm is shown in Table VII. We present the computational cost of the smoothing method, ORBSLAM3,

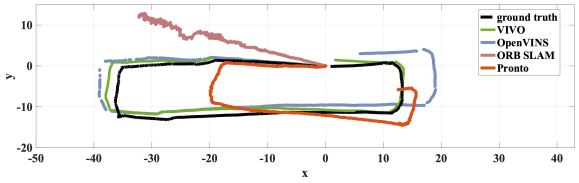


Fig. 8: Top view of the estimated trajectories of VIVO (green), OpenVINS (blue), ORB SLAM3 (pink), Pronto (pink) and ground truth (black).

TABLE VII: Mean (standard deviation) processing time for components of the proposed VIVO method and ORB Slam3.

Algorithm	Module		Mean (std.) [ms]
ORB SlAM3	Tracking	ORB extractio	19.606 (2.174)
		Stereo matching	22.31 (3.01)
		Preintegrate IMU	1.11 (0.60)
		Pose perdition	0.80 (1.66)
		Localmap track	11.83 (4.51)
		New keyframe decision	0.10 (0.13)
		Total	55.76 (5.29)
	Local Mapping*		299.59 (176.29)
VIVO	Feature tracking		7.94 (1.68)
	State propagation		0.60 (0.28)
	State update		17.41 (13.40)
	Slam delayed		1.31 (5.14)
	Re-triangulation & marginalization		2.71 (0.94)
	Total		29.97 (14.61)

* Traking and local mapping of ORB SLAM3 are run on different threads. Thus, the actual computational cost is more expensive than the recorded value due to thread conflicts.

for comparison. All tests are run on the same platform as the simulation experiments. It is evident that the computational cost of our method is reduced by almost half even when compared to the tracking part of the smoothing approach.

VI. Conclusion

In this paper, we propose a vision-inertial-velocity odometry (VIVO) system, which is a robust state estimation method, applying to those robots that can provide the body odometry information. Our key novelty is that we extend MSCKF-based VIO to incorporate velocity measurement from the body odometry in a tightly coupled approach, as well as online calibration of the extrinsic parameters between the visual odometry and body odometry. In particular, our method is suitable for the state estimation in high-dynamic scene, which is an intractable condition for the traditional VIO method.

The proposed algorithm is verified on simulation and real-world experiments using wheeled and legged robot model to show the robustness of online calibration and accuracy of the estimation in challenging scenarios. In comparison, our method outperforms the state-of-art visual-inertial algorithms and multi-sensor state estimator. In future work, we are motivated to advance the accuracy of the state estimation by replacing the linear interpolation of velocity with the high-order on-manifold state interpolation.

REFERENCES

[1] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. MIT press, 2011.

[2] G. P. Roston and E. P. Krotkov, "Dead reckoning navigation for walking robots," CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, Tech. Rep., 1991.

[3] G. Huang, "Visual-inertial navigation: A concise review," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 9572–9582.

[4] M. Bryson, M. Johnson-Roberson, and S. Sukkarieh, "Airborne smoothing and mapping using vision and inertial sensors," in *2009 IEEE International Conference on Robotics and Automation*. IEEE, 2009, pp. 2037–2042.

[5] D. Wisth, M. Camurri, and M. Fallon, "Preintegrated velocity bias estimation to overcome contact nonlinearities in legged robot odometry," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 392–398.

[6] ——, "Robust legged robot state estimation using factor graph optimization," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4507–4514, 2019.

[7] H. Strasdat, J. Montiel, and A. J. Davison, "Real-time monocular slam: Why filter?" in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 2657–2664.

[8] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A robust and modular multi-sensor fusion approach applied to mav navigation," in *2013 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2013, pp. 3923–3929.

[9] 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.

[10] R. Hartley, M. G. Jadidi, L. Gan, J.-K. Huang, J. W. Grizzle, and R. M. Eustice, "Hybrid contact preintegration for visual-inertial-contact state estimation using factor graphs," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3783–3790.

[11] 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.

[12] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," in *IEEE/RSJ International Conference on Intelligent Robots & Systems*. IEEE, 2013, pp. 1280–1286.

[13] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 3565–3572.

[14] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder, "Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds," *Information Fusion*, vol. 14, no. 1, pp. 57–77, 2013.

[15] Y. Yang, P. Geneva, K. Eckenhoff, and G. Huang, "Degenerate motion analysis for aided ins with online spatial and temporal sensor calibration," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2070–2077, 2019.

[16] N. Trawny and S. I. Roumeliotis, "Indirect kalman filter for 3d attitude estimation," *University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep*, vol. 2, p. 2005, 2005.

[17] M. Li and A. I. Mourikis, "High-precision, consistent ekf-based visual-inertial odometry," *The International Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, 2013.

[18] J. A. Hesch, D. G. Kottas, S. L. Bowman, and S. I. Roumeliotis, "Consistency analysis and improvement of vision-aided inertial navigation," *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 158–176, 2013.

[19] K. Eckenhoff, P. Geneva, and G. Huang, "Mimc-vins: A versatile and resilient multi-imu multi-camera visual-inertial navigation system," *IEEE Transactions on Robotics*, 2021.

[20] S. I. Roumeliotis and J. W. Burdick, "Stochastic cloning: A generalized framework for processing relative state measurements," in *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, vol. 2. IEEE, 2002, pp. 1788–1795.

[21] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 1280–1286.

[22] Y. Yang and G. Huang, "Observability analysis of aided ins with heterogeneous features of points, lines, and planes," *IEEE Transactions on Robotics*, vol. 35, no. 6, pp. 1399–1418, 2019.

[23] G. S. Chirikjian, *Stochastic models, information theory, and Lie groups, volume 2: Analytic methods and modern applications*. Springer Science & Business Media, 2011, vol. 2.

[24] M. Camurri, M. Fallon, S. Bazeille, A. Radulescu, V. Barasuol, D. G. Caldwell, and C. Semini, "Probabilistic contact estimation and impact detection for state estimation of quadruped robots," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 1023–1030, 2017.

[25] M. Bloesch, M. Hutter, M. A. Hoepflinger, S. Leutenegger, C. Gehring, C. D. Remy, and R. Siegwart, "State estimation for legged robots-consistent fusion of leg kinematics and imu," *Robotics*, vol. 17, pp. 17–24, 2013.

[26] M. Camurri, M. Ramezani, S. Nobili, and M. Fallon, "Pronto: a multi-sensor state estimator for legged robots in real world scenarios," *Frontiers in Robotics and AI*, vol. 7, p. 68, 2020.

[27] 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.

[28] C. Yang, K. Yuan, Q. Zhu, W. Yu, and Z. Li, "Multi-expert learning of adaptive legged locomotion," *Science Robotics*, vol. 5, no. 49, 2020.

[29] 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.

[30] 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 multi-map slam," *arXiv preprint arXiv:2007.11898*, 2020.

[31] D. J. Hyun, S. Seok, J. Lee, and S. Kim, "High speed trot-running: Implementation of a hierarchical controller using proprioceptive impedance control on the mit cheetah," *The International Journal of Robotics Research*, vol. 33, no. 11, pp. 1417–1445, 2014.

[32] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping," *arXiv preprint arXiv:2007.00258*, 2020.