2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Cerberus: Low-Drift Visual-Inertial-Leg Odometry For Agile Locomotion

Shuo Yang, Zixin Zhang, Zhengyu Fu, and Zachary Manchester

***Abstract*— We present an open-source Visual-Inertial-Leg Odometry (VILO) state estimation solution for legged robots, called Cerberus, which precisely estimates position on various terrains in real-time using a set of standard sensors, including stereo cameras, IMU, joint encoders, and contact sensors. In addition to estimating robot states, we perform online kinematic parameter calibration and outlier rejection to substantially reduce position drift. Hardware experiments in various indoor and outdoor environments validate that online calibration of kinematic parameters can reduce estimation drift to less than 1% during long-distance, high-speed locomotion. Our drift results are better than those of any other state estimation method using the same set of sensors reported in the literature. Moreover, our state estimator performs well even when the robot experiences large impacts and camera occlusion. The implementation of the state estimator, along with the datasets used to compute our results, is available at `https://github.com/ShuoYangRobotics/Cerberus`.**

I. Introduction

Using onboard sensors to estimate a robot's state (typically body pose and velocity) is a critical functionality for legged robots [1]–[4]. A sensor solution including only one pair of stereo cameras and critical proprioceptive sensors (IMU, joint encoders, and foot contact sensors) serves as an ideal choice for resource-constrained robots because this set of sensors is low cost, compact, and has low power consumption [5]. We call a state estimator using this sensing solution a Visual-Inertial-Leg-Odometry (VILO) estimator. VILO fuses data from different sensors by constructing observation models that predict measurements given robot states. Observation models combined with a dynamics model of the robot form a factor graph [6] describing a nonlinear optimization problem whose solution is the maximum-likelihood state estimate. Prior work [7]–[9] has shown that VILO outperforms methods that only utilize a subset of the aforementioned sensors, such as Visual-Inertial Odometry (VIO) [10] or Leg Odometry (LO) [3] alone.

A key feature of VIO estimators is online calibration of IMU biases using visual measurements [11]. Other key error sources in VIO have recently been systematically addressed [12]. However, in the VILO setting, systematic error analysis has yet to be established for leg sensors (joint encoders and contact sensors). Prior work has identified that, when generating body velocity estimates using LO, error sources such as foot slip, impacts, rolling contacts, and kinematic parameter errors [13]–[15] can degrade velocity estimation accuracy.

Authors are with the Robotics Institute and the Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213 USA. Emails: {shuoyang, zixinz, zhengyuf, zmanches}@andrew.cmu.edu

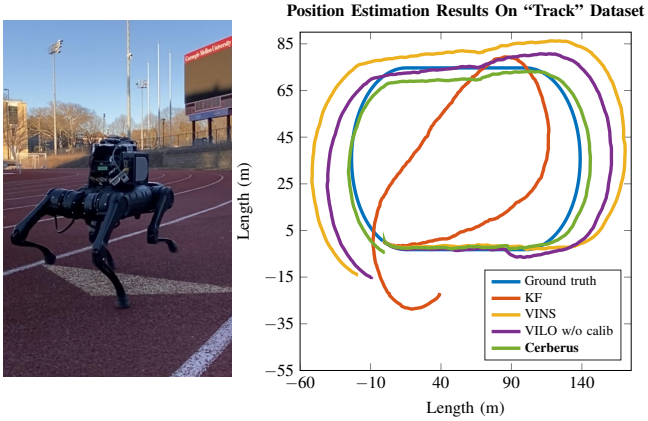

Fig. 1. On the A1 robot, the Cerberus algorithm has lower than 1% position estimation drift after traveling 450m on a standard stadium track, better than any baseline methods and better than any drift performance reported in the literature using the same set of sensors. The ground truth is obtained using the known dimensions of a standard running track.

However, no prior work has studied how to systematically handle these error sources in a VILO estimator.

Since different legged robots have different leg configurations, locomotion strategies, and sensor qualities, it is hard to fairly compare the performance of different VILO implementations. An open-source baseline VILO implementation and public datasets are needed for the benefit of the entire legged robotics community.

As a first step toward establishing a standard VILO benchmark, we present a state-of-the-art real-time VILO algorithm called Cerberus that incorporates kinematic calibration for improved accuracy, as well as several datasets from two different quadruped robots. The algorithm implementation uses standard ROS interfaces to process sensor data and publish estimation results, and the datasets are in the format of ROS bags [16]. Docker [17] provides easy installation of a unified testing environment. Our contributions are:

- Cerberus: a VILO algorithm that estimates kinematic parameters online to achieve drift rates lower than any other results reported in the literature.
- Datasets collected on multiple robots in various indoor and outdoor environments to benchmark the Cerberus implementation.
- Open-source algorithm implementations using standard ROS interfaces that can be readily adapted to different robots and sensor configurations.

This paper is organized as follows: In Section II we review related work. Section III introduces notation and

provides background. Section IV presents a basic VILO algorithm. Section V derives the online kinematic calibration method used in Cerberus. Section VI describes details of the algorithm implementation and presents hardware experiment results. Finally, section VII summarizes our conclusions.

II. Related Work

Using multiple sensors to estimate the physical state of a robot is one of the central topics of robotics. Although the Global Position System (GPS) can provide a good position estimation solution, many robots need to operate in GPS-denied environments. Visual odometry (VO) [18], which estimates robot pose using a monocular or a stereo camera, can provide a solution in these settings. By matching features across image sequences, feature locations constrain the possible motion of the camera so displacement can be solved from multiple-view geometry [19]. To improve robustness and accuracy, VIO [10] uses both the camera and the IMU as motion constraints. Preintegration [11], and factor graphs and their associated maximum a posteriori (MAP) estimation algorithms [6] can also help VIO to exploit problem structure, hence reducing computational cost. After the development of several VIO algorithms [10], [20], [21], researchers continue to study how to reject different error sources in VIO — including IMU biases, sensor delays, and extrinsic parameter errors [12]. The position drift percentage, measuring how many meters the estimation deviates from the ground truth after traveling 100 meters, is often used as an important performance metric. Once the error sources are properly addressed, position drift of a VIO estimator can be as low as 0.29% on drones [21].

Early legged robot state estimation work focused on fusing IMU and LO data using a Kalman Filter (KF), and analyzed error sources in this setting. A legged robot often experiences link deformations, foot slip, and excessive body rotation due to repeated impacts with the ground, all of which may lead to incorrect or biased velocity estimation. Bloesh et. al. showed that robot pose, velocity and IMU biases can be recovered using body IMU, joint encoders, and foot contact sensors [1]. A similar linear KF formulation is proposed in [2]. The invariant EKF is proposed in [22] to improve orientation estimation convergence. The non-slipping assumption of LO relies on accurate contact sensing [23] or slipping rejection mechanisms [13]. Some algorithms estimate contacts using kinematic information [24], eliminating the dependency on foot contact sensors. [15] identifies forward kinematic parameter errors due to link length changes and rolling contacts as another major error source in LO.

The factor graph formulation used in VIO can be easily extended to include the LO motion constraint, which leads to the VILO estimator [7], [8], [25]. [8] uses the velocity estimation result of a KF as the motion constraint. Contact preintegration is developed in [7], but bias correction is not performed. [14] describes the LO velocity bias and models it as a linear term that can be corrected in the preintegration. However, this bias model does not explain the source of the bias and its physical meaning. With the velocity bias model,

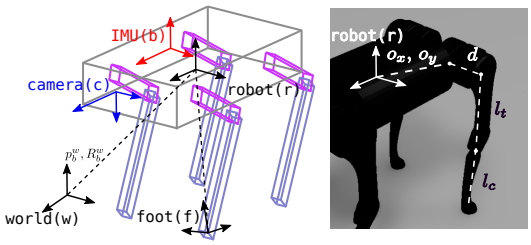

Fig. 2. Coordinate frame definitions & kinematic parameters used for the Unitree A1 robot.

[4] further shows that VILO can reach around 1% position drift with the aid of lidar, though their VILO implementation and datasets are not publicly available.

III. Background

We now introduce relevant notation and review some concepts from legged robot state estimation that were previously used in [15]. In general, we use lowercase letters for scalars and frame abbreviations, boldface lowercase letters for vectors, and upper case letters for matrices and vector sets. The operation $[a;b;c]$ vertically concatenates elements a, b and c. The operator $\lfloor \boldsymbol{v} \rfloor^{\times}$ converts a vector $\boldsymbol{v} = [v_1; v_2; v_3] \in \mathbb{R}^3$ into the skew-symmetric "cross-product matrix,"

$$\lfloor \boldsymbol{v} \rfloor^{\times} = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}, \tag{1}$$

such that $\boldsymbol{v} \times \boldsymbol{x} = \lfloor \boldsymbol{v} \rfloor^{\times} \boldsymbol{x}$. Lastly, $\hat{\boldsymbol{a}}$ indicates an estimate of $\boldsymbol{a}$.

A. Coordinate Frames & Quaternions

Important coordinate frames are shown in Fig. 2. For simplicity, we assume that the IMU frame and the robot's body frame coincide. We use $\boldsymbol{p}$ to denote the translation vector and and $\boldsymbol{q}$ to denote the unit-quaternion rotation from the robot's body frame to the world frame. We follow the quaternion convention defined in [26]. A quaternion $\boldsymbol{q} = [q_s; \boldsymbol{q}_v]$ has a scalar part q_s and a vector part $\boldsymbol{q}_v = [q_x; q_y; q_z] \in \mathbb{R}^3$. We define the two matrices,

$$L(\boldsymbol{q}) = \begin{bmatrix} q_s & -\boldsymbol{q}_v^{\top} \\ \boldsymbol{q}_v & q_s I + \lfloor \boldsymbol{q}_v \rfloor^{\times} \end{bmatrix} \quad \text{and} \quad R(\boldsymbol{q}) = \begin{bmatrix} q_s & -\boldsymbol{q}_v^{\top} \\ \boldsymbol{q}_v & q_s I - \lfloor \boldsymbol{q}_v \rfloor^{\times} \end{bmatrix},$$

such that the product of two quaternions can be written as,

$$\boldsymbol{q}_1 \otimes \boldsymbol{q}_2 = L(\boldsymbol{q}_1)\boldsymbol{q}_2 = R(\boldsymbol{q}_2)\boldsymbol{q}_1. \tag{2}$$

It can also be shown that the inverse of a unit quaternion $\boldsymbol{q}$ is $\boldsymbol{q}^{-1} = [q_s; -\boldsymbol{q}_v]$ and $\boldsymbol{q} \otimes \boldsymbol{q}^{-1} = \boldsymbol{q}_I = [1; \boldsymbol{0}]$, the identity quaternion. We also introduce a matrix $B = \begin{bmatrix} 0 \\ I_{3x3} \end{bmatrix}$ that converts a vector in $\mathbb{R}^3$ to a quaternion with zero scalar part. The rotation matrix $A(\boldsymbol{q})$ can then be written in terms of $\boldsymbol{q}$ as,

$$A(\boldsymbol{q}) = B^{\top} L(\boldsymbol{q}) R(\boldsymbol{q})^{\top} B. \tag{3}$$

Small rotation approximations play an important role in orientation estimation. We parameterize small rotations using

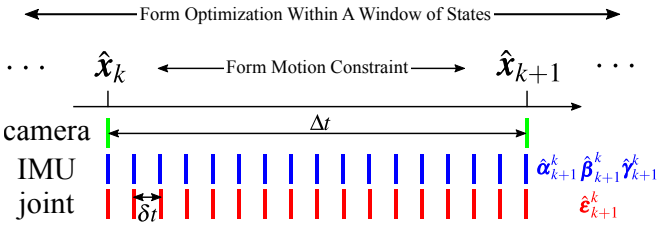

Fig. 3. Preintergation of IMU or joint measurements. The estimator adds a new state whenever a new camera image arrives. Within the time interval Δt between two consecutive states, there are multiple other sensor measurements with a sub-interval δt. We use k to refer to estimator time indices and i to indicate a measurement within Δt. These measurements are integrated according to equations (11), (12), (10), and (19), forming a motion constraint on states as in equation (18). The VILO Problem (9) involves such motion constraints and other constraints due to visual observations over a window of states.

Rodrigues parameters $\delta\boldsymbol{\theta} \in \mathbb{R}^3$ and map them into unit quaternions using the Cayley map [26]:

$$\delta\boldsymbol{q} = \Phi(\delta\boldsymbol{\theta}) = \frac{1}{\sqrt{1+\|\delta\boldsymbol{\theta}\|^2}}\begin{bmatrix}1\\ \delta\boldsymbol{\theta}\end{bmatrix}. \tag{4}$$

Assuming the true orientation of a robot is $\boldsymbol{q}$ and our estimate is $\hat{\boldsymbol{q}}$, we define the error as $\delta\boldsymbol{q} = \hat{\boldsymbol{q}}^{-1} \otimes \boldsymbol{q}$. We use the inverse Cayley map [26] $\Phi^{-1}(\boldsymbol{q}) = \boldsymbol{q}_v/q_s$ to convert the estimation error into Rodrigues parameters $\delta\boldsymbol{\theta}$. Therefore $\boldsymbol{q} = L(\hat{\boldsymbol{q}})\Phi(\delta\boldsymbol{\theta})$.

Where necessary, we use superscripts and subscripts to explicitly indicate the frames associated with rotation matrices and vectors, so $A^a_b \cdot p$ means the matrix transforms a vector p represented in coordinate frame b into coordinate frame a [27]. For brevity, if frame b is time-varying, and the context of frame association is clear, we write A_k instead of $A^w_{b(k)}$ to indicate the rotation matrix is also time dependent. Similarly, p_k defines the origin vector of frame b_k in the world.

B. Forward Kinematics & Leg Odometry Velocity

In this section we review forward kinematics and describe how to infer body velocity. We define $\boldsymbol{\phi}$ as a vector containing all joint angles of the robot's j'th leg, and $\dot{\boldsymbol{\phi}}$ the corresponding joint angle velocities. The forward kinematics function is denoted as $\boldsymbol{p}_f = g(\boldsymbol{\phi},\boldsymbol{\rho}) \in \mathbb{R}^3$, whose output is the foot position in the robot body frame. $\boldsymbol{\rho}$ is a set of kinematic parameters of interest, such as link lengths and motor offsets [15]. The derivative of this equation with respect to $\boldsymbol{\phi}$ leads to the Jacobian matrix $J(\boldsymbol{\phi},\boldsymbol{\rho})$ that maps $\dot{\boldsymbol{\phi}}$ into the foot's linear velocity in the body frame:

$$\boldsymbol{v}_f = \dot{\boldsymbol{p}}_f = J(\boldsymbol{\phi},\boldsymbol{\rho})\dot{\boldsymbol{\phi}}. \tag{5}$$

Assuming the j'th foot is in contact with the ground and does not slip, g and J can be used to calculate the body velocity of the robot. Let $\boldsymbol{p}^w_f$ denote the foot position in the world frame (see Fig. 2); It is a function of the robot's body position $\boldsymbol{p}$ and joint angles $\boldsymbol{\phi}$:

$$\boldsymbol{p}^w_f = \boldsymbol{p} + A(\boldsymbol{q})\boldsymbol{p}_f = \boldsymbol{p} + A(\boldsymbol{q})g(\boldsymbol{\phi},\boldsymbol{\rho}). \tag{6}$$

Let the time derivative of $\boldsymbol{p}^w_f$ be $\boldsymbol{v}^w_f$. The no-slip assumption means $\boldsymbol{v}^w_f = 0$. Therefore, by differentiating (6), we have

$$0 = \boldsymbol{v}^w_f = \dot{\boldsymbol{p}}^w_f = \dot{\boldsymbol{p}} + A(\boldsymbol{q})\frac{d}{dt}g(\boldsymbol{\phi},\boldsymbol{\rho}) + \frac{d}{dt}A(\boldsymbol{q})g(\boldsymbol{\phi},\boldsymbol{\rho}). \tag{7}$$

It is shown in [27] that $\frac{d}{dt}A(\boldsymbol{q}) = A(\boldsymbol{q})\lfloor\boldsymbol{\omega}\rfloor^\times$, where $\boldsymbol{\omega}$ is the robot body angular velocity. We define $\boldsymbol{v} = \dot{\boldsymbol{p}}$, then from (7) we derive an expression for the body velocity in the world frame:

$$\boldsymbol{v} = -A(\boldsymbol{q})[J(\boldsymbol{\phi},\boldsymbol{\rho})\dot{\boldsymbol{\phi}} + \lfloor\boldsymbol{\omega}\rfloor^\times g(\boldsymbol{\phi},\boldsymbol{\rho})]. \tag{8}$$

This velocity is called the LO velocity because its integration is the body displacement [28]. During legged locomotion, the kinematic parameters $\boldsymbol{\rho}$, which conventionally are deemed constant, change due to link deformations and rolling contacts [15]. Therefore, the parameter error can be viewed as a "bias" of the LO velocity measurement.

IV. Visual-Inertial-Leg Odometry

A typical VILO framework [7], [9], [14] keeps track of the estimation of a list of past N states $\hat{\boldsymbol{x}}_k$ and M camera feature locations $\hat{\lambda}_l$ as $\mathscr{X} = \{\hat{\boldsymbol{x}}_0, \hat{\boldsymbol{x}}_1, \ldots \hat{\boldsymbol{x}}_N, \hat{\lambda}_0, \hat{\lambda}_1, \ldots \hat{\lambda}_M\}$. The robot state is $\hat{\boldsymbol{x}}_k = [\hat{\boldsymbol{p}}_k; \hat{\boldsymbol{q}}_k; \hat{\boldsymbol{v}}_k; \hat{\boldsymbol{b}}_{ak}; \hat{\boldsymbol{b}}_{\omega k}]$, where $\hat{\boldsymbol{p}}_k \in \mathbb{R}^3$ is the robot position in the world frame, $\hat{\boldsymbol{q}}_k$ is the robot's orientation quaternion, and $\hat{\boldsymbol{v}}_k \in \mathbb{R}^3$ is the linear velocity of the robot's body represented in the world frame. $\hat{\boldsymbol{b}}_{ak} \in \mathbb{R}^3$ and $\hat{\boldsymbol{b}}_{\omega k} \in \mathbb{R}^3$ are IMU accelerometer bias and gyroscope bias. A new state $\hat{\boldsymbol{x}}_k$ is created each time t_k when a new camera image arrives. Also, sensors on the robot generate measurements $Z_t = \{\hat{\boldsymbol{a}}_m(t), \hat{\boldsymbol{\omega}}_m(t), \hat{\boldsymbol{\phi}}_j(t), \hat{\dot{\boldsymbol{\phi}}}_j(t)\}$ and Λ_t periodically, where $\hat{\boldsymbol{a}}_m$ and $\hat{\boldsymbol{\omega}}_m$ are IMU linear acceleration and angular velocity, $\hat{\boldsymbol{\phi}}_j$ and $\hat{\dot{\boldsymbol{\phi}}}_j$ are joint angle and joint angle velocity for each leg j, and Λ_t is a set of feature coordinates in the camera images that have known associations with feature locations in $\mathscr{X}$. We denote $\mathscr{Z}$ as all measurements between state $\hat{\boldsymbol{x}}_0$ and $\hat{\boldsymbol{x}}_N$. We also denote subsets $\mathscr{X}_{sub} \subset \mathscr{X}$ and $\mathscr{Z}_{sub} \subset \mathscr{Z}$. VILO constructs a nonlinear least-squares problem to find $\mathscr{X}$ as the solution of

$$\min_{\mathscr{X}} \left\{ \sum_i \left\| \boldsymbol{r}_i(\mathscr{X}_{sub}, \mathscr{Z}_{sub}) \right\|^2_{P_i} \right\}, \tag{9}$$

where each term $\boldsymbol{r}_i(\mathscr{X}_{sub}, \mathscr{Z}_{sub})$ defines a measurement residual function. Ideally the cost should be $\mathbf{0}$ at optimal solution $\mathscr{X}^*$. P_i is a weighting matrix that encodes the relative uncertainty in each $\boldsymbol{r}_i$, and also takes the same set of inputs. Problem (9) can be solved by nonlinear optimization methods [6]. The core technical challenge is to design cost functions leveraging all available sensor data. Additionally, a VILO estimator usually has other mechanisms to ensure real-time computation, such as visual feature tracking and sliding window marginalization. See [8], [21] for more details.

A. Preintegration

A key technique used in VIO and VILO to improve computation efficiency is preintergration. When fusing camera data and IMU data with different frequencies, preintegration [11] is used to integrate multiple IMU measurements between two camera image times into a single "motion

constraint" in the cost function, so the estimator only needs to add states at the camera frequency instead of keeping up with the much higher frequency of the IMU. More importantly, it is well known that IMUs are biased [29], and biases should be estimated along with robot's physical state. When the estimator updates IMU biases, IMU preintegration can avoid integrating measurements again by directly updating the integration term using its Jacobian. IMU preintegration is used in several real-time VIO algorithms [20], [21], [30]. Similarly, contact preintegration is used to integrate joint encoder data into motion constraints in VILO [31].

We assume there are L IMU measurements between state $\hat{\boldsymbol{x}}_k$ and $\hat{\boldsymbol{x}}_{k+1}$, and that each IMU measurement arrives δt after the previous one. Let $i \in \{1 \dots L\}$ be the measurement index and $\Delta t = t_{k+1} - t_k$, then $t_1 = t_k$ and $t_L = t_{k+1}$. As shown in Fig. 3, we can integrate these IMU measurements into a single motion measurement.

First, let $\hat{\boldsymbol{\gamma}}_i^k$ denote quaternion rotation from frame b_k to frame b_i, the robot body frame at time t_i. Starting from $\hat{\boldsymbol{\gamma}}_k^k = \boldsymbol{q}_I$, we can calculate,

$$\hat{\boldsymbol{\gamma}}_{i+1}^k = \hat{\boldsymbol{\gamma}}_i^k \Phi\left(\frac{\delta t}{2}(\hat{\boldsymbol{\omega}}_m(t_i) - \hat{\boldsymbol{b}}_{\omega k})\right), \tag{10}$$

which recursively leads to $\hat{\boldsymbol{\gamma}}_{k+1}^k$, a measurement of the rotation difference between $\hat{\boldsymbol{q}}_k$ and $\hat{\boldsymbol{q}}_{k+1}$. Another two recursive relations can be derived using acceleration data as

$$\hat{\boldsymbol{\alpha}}_{i+1}^k = \hat{\boldsymbol{\alpha}}_i^k + \hat{\boldsymbol{\beta}}_i^k \delta t, \text{ and} \tag{11}$$

$$\hat{\boldsymbol{\beta}}_{i+1}^k = \hat{\boldsymbol{\beta}}_i^k + A(\hat{\boldsymbol{\gamma}}_i^k)(\hat{\boldsymbol{a}}_m(t_i) - \hat{\boldsymbol{b}}_{ak})\delta t, \tag{12}$$

such that $\hat{\boldsymbol{\alpha}}_{k+1}^k$ and $\hat{\boldsymbol{\beta}}_{k+1}^k$ measure position and velocity differences between two states. These so called preintegration terms [11] describe a cost function on states as [21]

$$\begin{aligned}&\boldsymbol{r}(\hat{\boldsymbol{x}}_k, \hat{\boldsymbol{x}}_{k+1}, Z_{\Delta k}) = \\ &\begin{bmatrix} A(\hat{\boldsymbol{q}}_k)^T(\hat{\boldsymbol{p}}_{k+1} - \hat{\boldsymbol{p}}_k + \frac{1}{2}g^w\Delta t^2 - \hat{\boldsymbol{v}}_k\Delta t) - \hat{\boldsymbol{\alpha}}_{k+1}^k \\ \Phi^{-1}(\hat{\boldsymbol{q}}_k^{-1} \otimes \hat{\boldsymbol{q}}_{k+1} \otimes (\hat{\boldsymbol{\gamma}}_{k+1}^k)^{-1}) \\ A(\hat{\boldsymbol{q}}_k)^T(\hat{\boldsymbol{v}}_{k+1} + g^w\Delta t - \hat{\boldsymbol{v}}_k) - \hat{\boldsymbol{\beta}}_{k+1}^k \\ \hat{\boldsymbol{b}}_{ak+1} - \hat{\boldsymbol{b}}_{ak} \\ \hat{\boldsymbol{b}}_{\omega k+1} - \hat{\boldsymbol{b}}_{\omega k} \end{bmatrix},\end{aligned} \tag{13}$$

where $Z_{\Delta k}$ represents all measurements during Δt.

The error dynamics [21] of $\boldsymbol{r}$ as

$$\begin{aligned}\boldsymbol{e}_{i+1} = &\begin{bmatrix} I & I\delta t & 0 & 0 & 0 \\ 0 & I & -A(\hat{\boldsymbol{\gamma}}_i^k)\lfloor \hat{\boldsymbol{a}}_m(t_i) - \hat{\boldsymbol{b}}_{ak} \rfloor^\times \delta t & -A(\hat{\boldsymbol{\gamma}}_i^k)\delta t & 0 \\ 0 & 0 & I - \lfloor \hat{\boldsymbol{\omega}}_m(t_i) - \hat{\boldsymbol{b}}_{\omega k} \rfloor^\times \delta t & 0 & -I\delta t \\ 0 & 0 & 0 & I & 0 \\ 0 & 0 & 0 & 0 & I \end{bmatrix} \boldsymbol{e}_i \\ + &\begin{bmatrix} 0 & 0 & 0 & 0 \\ -A(\hat{\boldsymbol{\gamma}}_i^{b_k})\delta t & 0 & 0 & 0 \\ 0 & -I\delta t & 0 & 0 \\ 0 & 0 & I\delta t & 0 \\ 0 & 0 & 0 & I\delta t \end{bmatrix} \begin{bmatrix} \boldsymbol{n}_a \\ \boldsymbol{n}_\omega \\ \boldsymbol{n}_{ba} \\ \boldsymbol{n}_{b\omega} \end{bmatrix} = F_i \boldsymbol{e}_i + G_i \boldsymbol{n}_{IMU},\end{aligned} \tag{14}$$

where $\boldsymbol{n}_a$ and $\boldsymbol{n}_\omega$ are IMU sensor measurement noises and

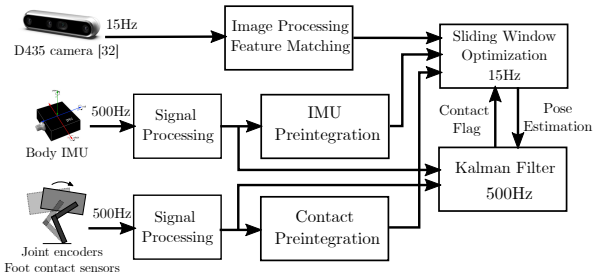

Fig. 4. The software architecture of Cerberus.

$\boldsymbol{n}_{ba}$ and $\boldsymbol{n}_{b\omega}$ are random walk processes for IMU biases. $\boldsymbol{e}_i = [\delta\boldsymbol{\alpha}_i^k; \delta\boldsymbol{\beta}_i^k; \delta\boldsymbol{\theta}_i^k; \delta\boldsymbol{b}_{ai}; \delta\boldsymbol{b}_{\omega i}]$ is a vector describing the errors between preintegration terms and their "true" values after each IMU measurement integration [21]. $\delta\boldsymbol{\alpha}_i^k = \boldsymbol{\alpha}_i^k - \hat{\boldsymbol{\alpha}}_i^k$, $\delta\boldsymbol{\beta}_i^k$, and $\boldsymbol{\gamma}_i^k = L(\hat{\boldsymbol{\gamma}}_i^k)\Phi(\delta\boldsymbol{\theta}_i^k)$. Details of the derivation can be seen in [21].

Let Q be the noise covairance matrix of $\boldsymbol{n}_{IMU}$. We can also recursively calculate P_{k+1}^k and the error Jacobian J_{k+1} as follows:

$$P_{i+1}^k = F_i P_i^k F_i^T + G_i Q G_i^T, P_1^k = 0, \tag{15}$$

$$J_{i+1} = F_i J_i, J_i = I. \tag{16}$$

The error Jacobian can greatly reduce VILO computation time: When solving Problem (9) using numerical methods, a solver iteratively calculates state update vectors $\delta\boldsymbol{x}$, and the update will change IMU biases. Instead of reintegrating the preintegration terms that depend on IMU biases, with the error Jacobian, we can directly update the preintegration terms, for example, as

$$\boldsymbol{\alpha}_{k+1}^k = \hat{\boldsymbol{\alpha}}_{k+1}^k + J_a^\alpha \delta\boldsymbol{b}_a + J_\omega^\alpha \delta\boldsymbol{b}_\omega \tag{17}$$

to get their revised values, where J_a^α and J_ω^α are blocks in J_{k+1} that correspond to $\partial\boldsymbol{\alpha}/\partial\boldsymbol{b}_a$ and $\partial\boldsymbol{\alpha}/\partial\boldsymbol{b}_\omega$.

V. Kinematic Calibration In Preintegration

In this section, we show how Cerberus estimates $\boldsymbol{\rho}$ for each leg by including them into the state so $\hat{\boldsymbol{x}}_k = [\hat{\boldsymbol{p}}_k; \hat{\boldsymbol{q}}_k; \hat{\boldsymbol{v}}_k; \hat{\boldsymbol{b}}_{ak}; \hat{\boldsymbol{b}}_{\omega k}; \hat{\boldsymbol{\rho}}_{jk}]$, where j is the leg index. For brevity, we only describe the case $j = 1$, but the algorithm applies to robots with any number of legs.

A. Contact Preintegration

For a leg that has non-slipping contact with the ground, (8) describes body velocity estimation through LO. This velocity can be integrated into a body displacement. We again focus on integrating measurements between state $\hat{\boldsymbol{x}}_k$ and $\hat{\boldsymbol{x}}_{k+1}$ including sensor data from leg sensors, then have a revised constraint equation

$$\boldsymbol{r}'(\hat{\boldsymbol{x}}_k, \hat{\boldsymbol{x}}_{k+1}, Z_{\Delta k}) = \begin{bmatrix} \boldsymbol{r}(\hat{\boldsymbol{x}}_k, \hat{\boldsymbol{x}}_{k+1}, Z_{\Delta k}) \\ A(\hat{\boldsymbol{q}}_k)^T(\hat{\boldsymbol{p}}_{k+1} - \hat{\boldsymbol{p}}_k) - \hat{\boldsymbol{\varepsilon}}_{k+1}^k \\ \hat{\boldsymbol{\rho}}_{k+1} - \hat{\boldsymbol{\rho}}_k \end{bmatrix}, \tag{18}$$

where $\hat{\boldsymbol{\varepsilon}}_{k+1}^k$ is the integration result of

$$\hat{\boldsymbol{\varepsilon}}_{i+1}^k = \hat{\boldsymbol{\varepsilon}}_i^k + A(\hat{\boldsymbol{\gamma}}_i^{b_k})\hat{\boldsymbol{v}}_i \delta t, \tag{19}$$

where,

$$\hat{\boldsymbol{v}}_i = -[J(\hat{\boldsymbol{\phi}},\hat{\boldsymbol{\rho}})\dot{\hat{\boldsymbol{\phi}}} + \lfloor \hat{\boldsymbol{\omega}} - \hat{\mathbf{b}}_{\omega k} \rfloor^{\times} g(\hat{\boldsymbol{\phi}},\hat{\boldsymbol{\rho}})]. \tag{20}$$

Comparing to (13), (18) introduces the LO velocity integration as a measurement of change in body position. The term $\hat{\boldsymbol{\varepsilon}}^k_{k+1}$ depends on sensor measurements, $\hat{\boldsymbol{b}}_{\omega k}$, and $\hat{\boldsymbol{\rho}}_k$. A version without kinematic parameter dependency was previously derived in [7]. The error of this measurement, defined as $\boldsymbol{e}'_i = [\boldsymbol{e}_i; \delta\boldsymbol{\varepsilon}^k_i; \delta\boldsymbol{\rho}_i]$, has dynamics

$$\boldsymbol{e}'_{i+1} = \begin{bmatrix} & & F_i & & & & & \mathbf{0} & \\ 0 & I & -A(\hat{\boldsymbol{\gamma}}^{j_k}_i)\lfloor \hat{\boldsymbol{v}}_i \rfloor^{\times}\delta t & 0 & \boldsymbol{\zeta}\delta t & 0 & \boldsymbol{\kappa}\delta t \\ 0 & 0 & 0 & & 0 & 0 & 0 & 0 \end{bmatrix} \boldsymbol{e}'_i$$
$$+ \begin{bmatrix} & & G_i & & & & & \mathbf{0} & \\ 0 & \boldsymbol{\zeta}\delta t & 0 & 0 & \boldsymbol{\eta}\delta t & A(\hat{\boldsymbol{\gamma}}^{j_k}_i)J\delta t & I\delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & I\delta t \end{bmatrix} \begin{bmatrix} \boldsymbol{n}_{IMU} \\ \boldsymbol{n}_{\phi} \\ \boldsymbol{n}_{\dot{\phi}} \\ \boldsymbol{n}_{v} \\ \boldsymbol{n}_{\rho} \end{bmatrix}, \tag{21}$$

in which J is short for $J(\boldsymbol{\phi},\hat{\boldsymbol{\rho}})$, the forward kinematics Jacobian. $\boldsymbol{e}_t$, $\boldsymbol{n}_t$, F_t, and G_t are defined in 14. The definitions of $\boldsymbol{\zeta}$, $\boldsymbol{\eta}$, and $\boldsymbol{\kappa}$, along with the derivation of the error dynamics, are in the Appendix. $\boldsymbol{n}_{\phi} \sim \mathcal{N}(0,\sigma^2_{\phi})$ and $\boldsymbol{n}_{\dot{\phi}} \sim \mathcal{N}(0,\sigma^2_{\dot{\phi}})$ are the measurement noise of joint angle and joint angle velocity. $\boldsymbol{n}_{\rho} \sim \mathcal{N}(0,\sigma^2_{\rho})$ is the kinematic parameter random walk noise. $\boldsymbol{n}_v \sim \mathcal{N}(0,\sigma^2_v)$ is the uncertainty of the contact preintegration motion constraint.

From the error dynamics, we can get P^k_{k+1} and J_{k+1} as in (15) and (16). Then Jacobians such as $J^{\varepsilon}_{\rho} = \frac{\partial \varepsilon^k_{k+1}}{\partial \rho}$ extracted from J^k_{k+1} can allow fast preintegration updates:

$$\boldsymbol{\varepsilon}^k_{k+1} = \hat{\boldsymbol{\varepsilon}}^k_{k+1} + J^{\varepsilon}_{\omega}\delta\boldsymbol{b}_{\omega} + J^{\varepsilon}_{\rho}\delta\boldsymbol{\rho}. \tag{22}$$

This technique is critical for enabling real-time computation in Cerberus while doing kinematic calibration.

B. Contact-Aware Measurement Noise

Contact preintegration can only serve as a valid measurement when the robot foot is stationary between two time steps. We reflect this fact in the measurement noise.

Assume the robot has access to a contact flag $c \in \{0,1\}$ indicating whether the foot is in contact (1) or not (0). The flag may come from a foot contact sensor or an estimation algorithm [24]. For the noise covariances in (21), we let

$$\sigma_{\rho} = c\sigma_c + (1-c)\sigma_{nc}, \tag{23}$$
$$\sigma_v = c\sigma_0 + (1-c)\sigma_1, \tag{24}$$

which means we give the kinematic parameter and velocity measurement low uncertainty values when the foot has contact, otherwise the uncertainty is high. σ_c, σ_{nc}, σ_0, and σ_1 are all tunable hyper-parameters of the measurement model.

For robots without contact sensors, we leverage a standard outlier-rejection method common in Kalman Filter implementations [13] that fuses IMU information and the LO velocity. If the filter treats a leg as stationary according to a prior contact schedule, then velocity calculated using (8) should agree with the current robot body velocity estimation. Otherwise, the prior contact schedule is wrong so the actual contact flag should be reverted.

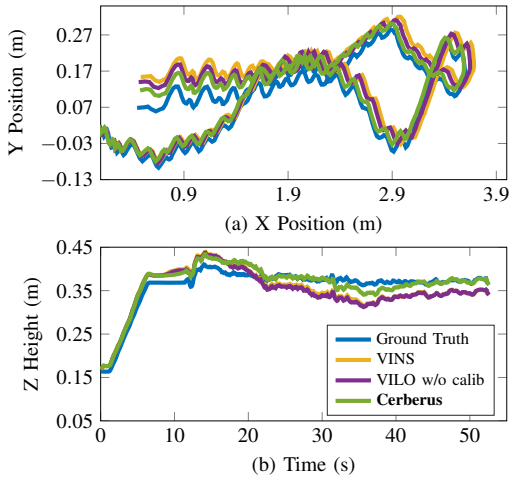

Fig. 5. Compared to baseline estimators, Cerberus achieves lower drift in all directions. The final drift of the VINS trajectory (red) is 1.73% while the drift of the VILO w/o calib trajectory (yellow) is 1.25% and that of Cerberus (purple) is 1.13%.

Dataset	KF-PO	VIO	VILO w/o calib	Cerberus
Indoor (average 10)	6.53%	1.31%	1.02%	**0.92%**
Street	> 10%	0.89%	**0.70%**	0.85%
Track	> 10%	3.9%	2.6%	**0.98%**
Campus	> 10%	break	3.32%	**1.65%**

TABLE I
HARDWARE EXPERIMENT FINAL DRIFTS COMPARISON

VI. EXPERIMENTS

Our C++ implementation of Cerberus uses the factor-graph optimizer and vision front end of the open-source visual-inertial odometry software VINS-Fusion [21]. The IMU factor in VINS-Fusion is replaced with our cost function (18). We set $\boldsymbol{\rho} = [l_c]$, the calf length shown in Fig. 2, as it is changing during locomotion [15]. We conducted experiments on sensor data collected on two quadruped robot platforms, the Unitree A1 and Go1 [33]. Both robots perform trotting using different controller implementations. The system architecture and the list of sensors that provide data to Cerberus are summarized in Fig. 4.

We focus on comparing the position drift percentages of a Kalman Filter (KF) [1], visual-inertial odometry (VINS) [21], VILO without kinematics calibration (VILO w/o calib) as described in Section IV, and Cerberus. The only difference between the last two is the VILO w/o calib uses a fixed value $\boldsymbol{\rho} = 0.21$ m while Cerberus calibrates the kinematic parameters. The parameter calibration results of $\boldsymbol{\rho}$ can be seen in [15].

A. Indoor Experiments

In a lab space equipped with an OptiTrack [34] motion-capture system, the robot moves on flat ground following

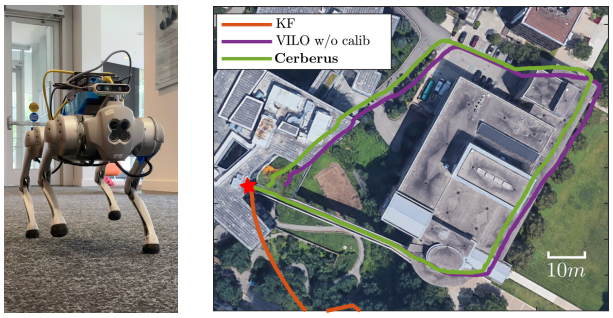

Fig. 6. In the "Campus" dataset, the Go1 robot ran 345 m with an average speed of 1 m/s in indoor and outdoor environments. VINS fails in the middle, so no result is shown. Cerberus has the smallest final position drift after returning to the starting point (red star).

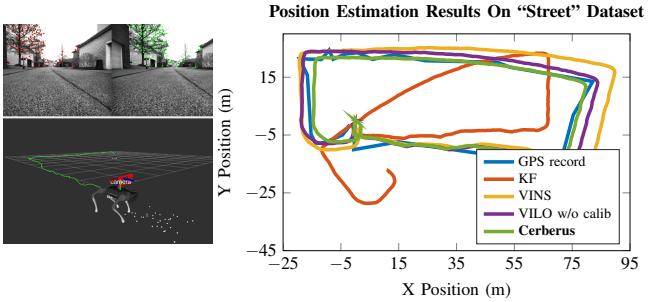

Fig. 7. Algorithm comparisons on the "Street" dataset. The final drift of Cerberus is 2.22 m (0.85% after 260m travel) compared to 1.84 m (0.71%) for VILO w/o calib. In this case, both estimators achieve very low drift and the differences are not statistically significant.

different paths with an average speed of 0.5 m/s. We record sensor data and ground-truth positions. We then run Cerberus on a desktop computer with Intel i7-7800X 3.50GHz CPU. The processing time is 50 ms per camera frame on average, which is faster than the camera sample rate (66 ms). Therefore, the state estimator is able to run in real time. Fig. 5 compares the ground truth trajectory (blue) with the estimated trajectory using VINS (red), VILO w/o calib (yellow), and Cerberus (purple) in one dataset. Table I shows average performance over 10 datasets.

B. Outdoor Experiments

The contribution of kinematics calibration to long-term position estimation is verified in outdoor experiments. Two robots collected datasets in several outdoor environments while traveling over 1.5 km with an average velocity of 0.5 m/s. Note that our robots move at a much faster speed than prior works (for example, [9] is 0.125 m/s and [4] is 0.25 m/s). In each dataset, the robot moves in a large loop and we evaluate the final position estimation drift after the robot returns to the starting point. We also note that 1% drift is equivalent to 0.1 m of the 10M Relative Translation Error (RTE) metric used in [4] and [9]. Dataset details can be found in the open-source code base.

Figures 1, 6, and 7 compare the estimated trajectories for three datasets, "Track", "Campus", and "Street." Table I contains a quantitative analysis of drift percentage for different datasets. The "Campus" dataset is particularly difficult because the robot runs at over 1 m/s on various indoor and outdoor terrains with different slopes (see the supplementary video). Cerberus outperforms all other methods across all datasets except for "Street," where both methods have very small drift values that have no statistically significant difference. Even though our datasets are longer and contain faster and more challenging dynamics, the Cerberus algorithm achieves $< 1\%$ drift on most of them and 1.65% drift on the hardest case. No prior work has achieved this level of performance.

C. Robust Estimation

Since Cerberus combines various sensor sources, the position estimation is robust against camera occlusion, foot slippage, and excessive body vibration. The supplementary video contains more challenging scenarios that demonstrate the robustness of the estimator.

VII. Conclusions

We have presented Cerberus, a VILO algorithm using kinematics calibration in contact preintegration and contact outlier rejection to improve performance. Indoor and outdoor experiments on two robots have demonstrated that our state estimator outperforms many existing methods. We believe kinematics parameter errors, like IMU biases, should be considered to achieve precise long-term estimation for legged robots. Cerberus can serve as a baseline for studying the influence of different gaits, movement directions, and terrain types on odometry accuracy in future legged robot research.

Appendix

In (20), the $\hat{\boldsymbol{v}}$ is an estimate. Now write a "true" measurement considering noisy system state and expand as

$$\begin{aligned}\boldsymbol{v}_m =& -J(\boldsymbol{\phi}-\boldsymbol{n}_\phi,\boldsymbol{\rho})(\dot{\boldsymbol{\phi}}-\boldsymbol{n}_{\dot\phi}) \\ &-\lfloor \boldsymbol{\omega}_m-\boldsymbol{b}_\omega-\boldsymbol{n}_\omega\rfloor^\times g(\boldsymbol{\phi}-\boldsymbol{n}_\phi,\boldsymbol{\rho}) \qquad (25)\\ =& -J(\boldsymbol{\phi},\boldsymbol{\rho})\dot{\boldsymbol{\phi}}-\lfloor \boldsymbol{\omega}_m-\boldsymbol{b}_\omega\rfloor^\times g(\boldsymbol{\phi},\boldsymbol{\rho}) \\ &-\lfloor g(\boldsymbol{\phi},\boldsymbol{\rho})\rfloor^\times \boldsymbol{n}_\omega + J(\boldsymbol{\phi},\boldsymbol{\rho})\boldsymbol{n}_{\dot\phi} \\ &+[(\dot{\boldsymbol{\phi}}^T\otimes I_3)\frac{\partial vec(J)}{\partial \boldsymbol{\phi}}+\lfloor \boldsymbol{\omega}_m-\boldsymbol{b}_\omega\rfloor^\times J]\boldsymbol{n}_\phi, \qquad (26)\end{aligned}$$

where $vec(J)$ is a vertical stack of columns of J. $\otimes$ is the kronecker product. According to the definition of $\boldsymbol{e}'_t$ and (19)

$$\delta\dot{\boldsymbol{\varepsilon}}^k_t=\dot{\boldsymbol{\varepsilon}}^k_t-\dot{\hat{\boldsymbol{\varepsilon}}}^k_t=A(\boldsymbol{\gamma}^{b_k}_t)(\boldsymbol{v}_{m_t}+\boldsymbol{n}_v)-A(\hat{\boldsymbol{\gamma}}^{b_k}_t)\hat{\boldsymbol{v}}_{m_t}. \qquad (27)$$

Recall that $\boldsymbol{\rho}=\hat{\boldsymbol{\rho}}+\delta\boldsymbol{\rho}$ and $\boldsymbol{b}_\omega=\hat{\boldsymbol{b}}_\omega+\delta\boldsymbol{b}_\omega$. Continue expanding (27) while ignoring second order delta terms [21],

$$\begin{aligned}\delta\dot{\boldsymbol{\varepsilon}}^k_t=&\dot{\boldsymbol{\varepsilon}}^k_t-\dot{\hat{\boldsymbol{\varepsilon}}}^k_t \\ =&-A(\hat{\boldsymbol{\gamma}}^{b_k}_t)\lfloor\hat{\boldsymbol{v}}_m\rfloor^\times\delta\boldsymbol{\theta}^k_t+\boldsymbol{\zeta}\delta\boldsymbol{b}_\omega+\boldsymbol{\kappa}\delta\boldsymbol{\rho} \\ &+\boldsymbol{\zeta}\boldsymbol{n}_\omega+\boldsymbol{\eta}\boldsymbol{n}_\phi+A(\hat{\boldsymbol{\gamma}}^{b_k}_t)J\boldsymbol{n}_{\dot\phi}+\boldsymbol{n}_v, \qquad (28)\end{aligned}$$

where $\boldsymbol{\zeta}=-A(\hat{\boldsymbol{\gamma}}^{b_k}_t)\lfloor g\rfloor^\times$, $\boldsymbol{\kappa}=-A[(\dot{\boldsymbol{\phi}}^T\otimes I_3)\frac{\partial vec(J)}{\partial\hat{\rho}}+\lfloor\boldsymbol{\omega}_m-\hat{\boldsymbol{b}}_\omega\rfloor^\times J]$ and $\boldsymbol{\eta}=A[(\dot{\boldsymbol{\phi}}^T\otimes I_3)\frac{\partial vec(J)}{\partial\phi}+\lfloor\boldsymbol{\omega}_m-\hat{\boldsymbol{b}}_\omega\rfloor^\times J]$.

REFERENCES

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

[2] G. Bledt, M. J. Powell, B. Katz, J. Di Carlo, P. M. Wensing, and S. Kim, "Mit cheetah 3: Design and control of a robust, dynamic quadruped robot," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 2245–2252.

[3] 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.

[4] D. Wisth, M. Camurri, and M. Fallon, "Vilens: Visual, inertial, lidar, and leg odometry for all-terrain legged robots," *IEEE Transactions on Robotics*, 2022.

[5] M. Bloesch, "State estimation for legged robots-kinematics, inertial sensing, and computer vision," Ph.D. dissertation, ETH Zurich, 2017.

[6] F. Dellaert, M. Kaess *et al.*, "Factor graphs for robot perception," *Foundations and Trends® in Robotics*, vol. 6, no. 1-2, pp. 1–139, 2017.

[7] 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 *International Conference on Intelligent Robots and Systems*, 2018, pp. 3783–3790.

[8] D. Wisth, M. Camurri, and M. Fallon, "Robust legged robot state estimation using factor graph optimization," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4507–4514, 2019.

[9] Y. Kim, B. Yu, E. M. Lee, J.-h. Kim, H.-w. Park, and H. Myung, "Step: State estimator for legged robots using a preintegrated foot velocity factor," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4456–4463, 2022.

[10] 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.

[11] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation." Georgia Institute of Technology, 2015.

[12] T. Qin and S. Shen, "Online temporal calibration for monocular visual-inertial systems," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 3662–3669.

[13] M. Bloesch, C. Gehring, P. Fankhauser, M. Hutter, M. A. Hoepflinger, and R. Siegwart, "State estimation for legged robots on unstable and slippery terrain," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 6058–6064.

[14] 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.

[15] S. Yang, H. Choset, and Z. Manchester, "Online kinematic calibration for legged robots," *IEEE Robotics and Automation Letters*, 2022.

[16] Stanford Artificial Intelligence Laboratory et al., "Robotic operating system." [Online]. Available: https://www.ros.org

[17] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux journal*, vol. 2014, no. 239, p. 2, 2014.

[18] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE robotics & automation magazine*, vol. 18, no. 4, pp. 80–92, 2011.

[19] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[20] K. Sun, K. Mohta, B. Pfrommer, M. Watterson, S. Liu, Y. Mulgaonkar, C. J. Taylor, and V. Kumar, "Robust stereo visual inertial odometry for fast autonomous flight," *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 965–972, 2018.

[21] 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.

[22] R. Hartley, M. Ghaffari, R. M. Eustice, and J. W. Grizzle, "Contact-aided invariant extended kalman filtering for robot state estimation," *The International Journal of Robotics Research*, vol. 39, no. 4, pp. 402–430, 2020.

[23] 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.

[24] J. Hwangbo, C. D. Bellicoso, P. Fankhauser, and M. Hutter, "Probabilistic foot contact estimation by fusing information from dynamics and differential/forward kinematics," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 3872–3878.

[25] J.-H. Kim, S. Hong, G. Ji, S. Jeon, J. Hwangbo, J.-H. Oh, and H.-W. Park, "Legged robot state estimation with dynamic contact event information," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6733–6740, 2021.

[26] B. E. Jackson, K. Tracy, and Z. Manchester, "Planning with attitude," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 5658–5664, 2021.

[27] R. M. Murray, Z. Li, and S. S. Sastry, *A mathematical introduction to robotic manipulation*. CRC press, 2017.

[28] P.-C. Lin, H. Komsuoglu, and D. E. Koditschek, "A leg configuration measurement system for full-body pose estimates in a hexapod robot," *IEEE Transactions on robotics*, vol. 21, no. 3, pp. 411–422, 2005.

[29] D. Adams, "Introduction to inertial navigation," *The Journal of Navigation*, vol. 9, no. 3, pp. 249–259, 1956.

[30] V. Usenko, J. Engel, J. Stückler, and D. Cremers, "Direct visual-inertial odometry with stereo cameras," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1885–1892.

[31] R. Hartley, J. Mangelson, L. Gan, M. G. Jadidi, J. M. Walls, R. M. Eustice, and J. W. Grizzle, "Legged robot state-estimation through combined forward kinematic and preintegrated contact factors," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 4422–4429.

[32] Intel, "Intel Realsense D435," https://www.intelrealsense.com/depth-camera-d435/, 2022, [Online; accessed 10-Sep-2022].

[33] Unitree, "A1," https://www.unitree.com/products/a1/, 2022, [Online; accessed 10-Sep-2022].

[34] OptiTrack, "OptiTrack," https://optitrack.com/, 2022, [Online; accessed 10-Sep-2022].