Online Kinematic Calibration for Legged Robots

Shuo Yang, *Graduate Student Member, IEEE*, Howie Choset, *Fellow, IEEE*, and Zachary Manchester, *Member, IEEE*

***Abstract*—This paper describes an online method to calibrate certain kinematic parameters of legged robots, including leg lengths, that can be difficult to measure offline due to dynamic deformation effects and rolling contacts. A kinematic model of the robot's legs that depends on these parameters is used, along with measurements from joint encoders, foot contact sensors, and aninertial measurement unit (IMU) to predict the robot's body velocity. This predicted velocity is then compared to another velocity measurement from, for example, a camera or motion capture system, and the difference between them is used to compute anupdate on the kinematic parameters. The method can be incorporated into both Kalman filter or sliding-window optimization-based state estimator. We provide a theoretical observability analysis of our method, as well as validation both in simulation and on hardware. Hardware experiments demonstrate that online kinematic calibration can significantly reduce position drift when relying on odometry.**

***Index Terms*—Legged robots, calibration and identification, probability and statistical methods.**

I. Introduction

STATE estimation and control algorithms for legged robots depend critically on leg kinematic parameters. During locomotion, a planner calculates foot positions to generate collision-free foot-swing trajectories [1], and an estimator computes foot velocities to estimate the robot body's velocity [2], [3]. Foot positions and velocities are calculated using the forward kinematics [3], [4], which depend on kinematic parameters such as leg link lengths and motor offset distances.

On legged robots, the Leg Odometry (LO) [5] is commonly used to estimate robot body velocity. Assuming non-slipping contact, the joint angle velocity measurements of a leg can be mapped into a body velocity estimation by the Jacobian of the forward kinematics. Inaccurate leg-length knowledge can lead to velocity estimation errors, as shown in Fig. 1(b), which prevent the robot state estimator from getting correct odometry information.

Existing legged robot controllers usually use fixed leg-length values obtained from a 3D model of the robot or manual measurement [3], [6], [7]. However, actual kinematic parameters are often not precisely known due to manufacturing variations, wear over time, rolling contacts, and dynamic deformation during normal operation (see Fig. 1(a)). Taking the Unitree A1 robot's kinematic structure as an example [8], its deformable foot makes the calf link length vary between 0.19–0.22 m. When the robot moves at 2.0 m/s, joint angle velocity can reach 20 rd/s, then a 0.01 m error in link length leads to 0.2 m/s velocity estimation error. If we integrate this poor velocity estimate to get a position estimate, position drift can grow by 0.2 m every second. Moreover, it is possible for the leg to experience sudden length changes due to external impacts or damage. Although the controller may be robust enough to maintain balance [9], knowledge of the new length can greatly improve stability and reduce foot slip. Therefore, online calibration of kinematic parameters can be hugely beneficial during robot operation.

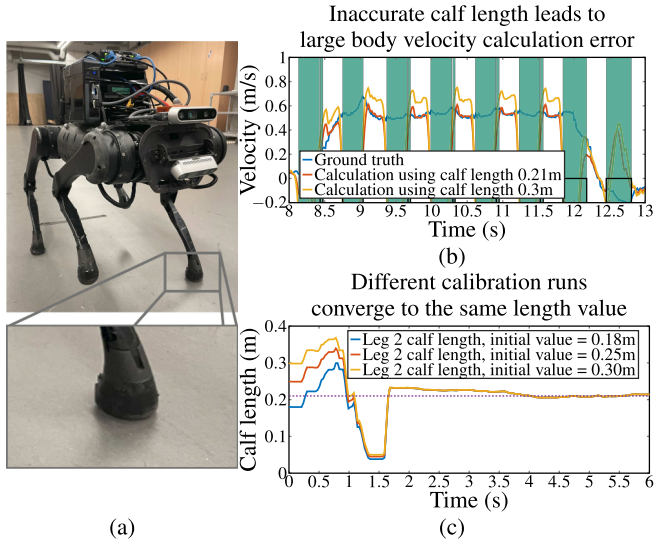

Fig. 1. (a) Foot compression during locomotion changes calf length. (b) Body velocity inferred from kinematics of the Leg 2 (front-left) with wrong calf length (yellow) has larger error comparing to that using calibrated length (red). we enlarge possible leg length error to make velocity error more perceptible. The shaded green regions are periods when the leg not contacts the ground, during which the LO velocity is meaningless. (c) Our method calibrates calf length of the Leg 2 to around 0.21 m (dash line) whatever the initial length value is.

We propose a method to enable legged robots to calibrate unknown kinematic parameters online. We design a velocity measurement model that compares the LO velocity with accurate body velocity information from an external motion-capture system or visual odometry [10], and use the difference between these measurements to update the kinematic parameters. This can be integrated into many standard state-estimation algorithms so that kinematic parameters can be estimated in real time together with the robot's state.

Manuscript received 24 February 2022; accepted 14 June 2022. Date of publication 27 June 2022; date of current version 7 July 2022. This work was supported in part by NVIDIA Corporation. This letter was recommended for publication by Associate Editor Y. Aoustin and Editor A. Kheddar upon evaluation of the reviewers' comments.

The authors are with the Robotics Institute and Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh USA (e-mail: shuoyang@andrew.cmu.edu; choset@cs.cmu.edu; zacm@cmu.edu).

Digital Object Identifier 10.1109/LRA.2022.3186501

We present experiments with a Kalman filter that demonstrate position drift can be reduced by up to an order of magnitude through online calibration. We also show doing online calibration in an optimization-based sliding-window state estimator [11] is possible.

The paper proceeds as follows: In Section II we survey related work in kinematics calibration and legged robot state estimation. Section III reviews Kalman filtering and forward kinematics. In Section IV we define the calibration problem, derive our solution, and provide an observability analysis. The results of simulation and hardware experiments are presented in Section V. Finally, Section VI concludes the paper.

Our contributions include:

- A measurement model to calibrate unknown kinematic parameters of legged robots online and reduce odometry drift.
- A theoretical observability analysis to examine which kinematic parameters are observable.
- Integration of the measurement model into two different state estimators.
- Experimental validation on a Unitree A1 robot.

II. Related Work

Kinematics calibration for robot manipulators has been studied for decades [12]. Standard methods use the error between the end-effector position output from the forward kinematics model and position measurement from an accurate sensor to update the model parameters. Early calibration approaches relied on laser trackers [13]. Today, "hand-eye" calibration with computer vision is accurate enough to calibrate manipulators and even humanoid robot arms [14].

Online parameter calibration during state estimation has also been investigated in multi-sensor fusion. When using different sensors together to estimate robot pose (position and orientation), rigorous observability proofs have demonstrated that the extended Kalman Filter (EKF) [10], [15] can estimate spatial transformations among sensors (extrinsics). The visual-inertial system (VINS) [16] calibrates not only IMU-camera extrinsics parameters, but also sensor time delay in a sliding-window (also called fix-lag smoother [17] or receding-horizon [18]) optimization based estimation scheme. This work shows that, in visual-inertial odometry, jointly estimating robot state, sensor bias, and sensor transformations can reduce long-term position estimation drift. The observability study of these parameters to guarantee that they can be estimated in both EKF and optimization-based state estimators is done in [19].

State estimation for legged locomotion has been a thriving research area in recent years, but only a few existing works consider online kinematics-parameter calibration, and the effects of parameter variations over time on estimator performance has not been well studied in the literature. Fusing IMU data and LO velocity in Kalman filters has been used on several different robots [2], [3], [5]. The invariant EKF was introduced in [20] to improve filter convergence. Optimization-based methods were proposed in [7], [21], [22]. The position drift due to leg odometry

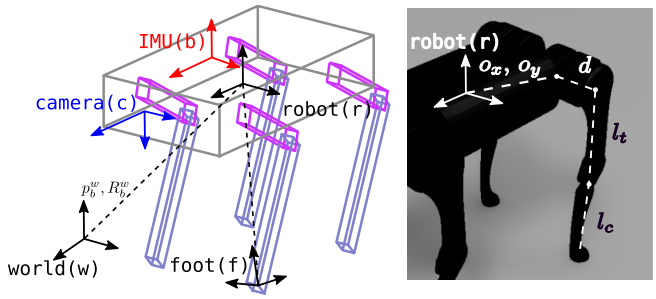

Fig. 2. Frames & Kinematic parameters of A1 robot.

was studied in [23], where drift was compensated by a body velocity bias.

With the aid of special markers, "foot-eye" kinematics calibration on legged robots was demonstrated in [24]. The authors of [25] calibrated camera extrinsics in an optimization-based legged robot state estimator. A measurement model to allow the robot's controller to adapt to dynamic model changes online was proposed in [26]. Dynamic deformation during bipedal locomotion was considered in [27], where the deformation is modeled as rotations among links. Estimating the deformation improves both position estimation and control. Finally, the kinematics calibration method proposed in [28] is similar to ours. However, it runs an expensive offline batch optimization and no observability analysis is provided.

III. Background

We now review some concepts from legged robot state estimation and introduce notations. 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 with the same type (scalar, vector, or matrix). 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 $\boldsymbol{v} \times \boldsymbol{x} = \lfloor \boldsymbol{v} \rfloor^\times \boldsymbol{x}$.

A. Coordinate Frames & Quaternion

Important coordinate frames are shown in Fig. 2. To simplify discussion, we assume that the IMU frame and the robot's body frame coincide. We use $\boldsymbol{p}$ and $\boldsymbol{q}$ to denote the translation vector and the unit-quaternion rotation, respectively, from the robot body frame to the world frame. We follow the quaternion convention defined in [29]. A quaternion $\boldsymbol{q} = [q_w; \boldsymbol{q}_v]$ has a scalar part q_w and a vector part $\boldsymbol{q}_v = [q_x; q_y; q_z] \in \mathbb{R}^3$. We define the two matrices,

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

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 $\boldsymbol{q}^{-1} = [q_w; -\boldsymbol{q}_v]$ and $\boldsymbol{q} \otimes \boldsymbol{q}^{-1} = \boldsymbol{q}_I = [1; \mathbf{0}]$, 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 approximation plays an important role in orientation estimation. We parameterize small rotations using Rodrigues parameters $\delta\boldsymbol{\theta} \in \mathbb{R}^3$ and map them into unit quaternions using the Cayley map [29]:

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

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 [29] $\Phi^{-1}(\boldsymbol{q}) = \boldsymbol{q}_v/q_s$ to convert the estimation error into Rodrigues parameters $\delta\boldsymbol{\theta}$. Therefore $\boldsymbol{q} = L(\hat{\boldsymbol{q}})\Phi(\delta\boldsymbol{\theta})$.

B. IMU-Driven Error-State Kalman Filter

One of the standard approaches to estimate a robot's pose and velocity is the error-state KF [30]. Let $\boldsymbol{x}_k = [\boldsymbol{p}; \boldsymbol{q}; \boldsymbol{v}] \in \mathbb{R}^{10}$ be the true robot state at time step k, where $\boldsymbol{p} \in \mathbb{R}^3$ is the robot position in the world frame, $\boldsymbol{q}$ is the robot's orientation quaternion, and $\boldsymbol{v} \in \mathbb{R}^3$ is the linear velocity of the robot's body represented in the world frame. We also denote the estimate of the robot's state as $\hat{\boldsymbol{x}}_k = [\hat{\boldsymbol{p}}; \hat{\boldsymbol{q}}; \hat{\boldsymbol{v}}]$. State errors are parameterized as $\delta\boldsymbol{x}_k = [\delta\boldsymbol{p}; \delta\boldsymbol{\theta}; \delta\boldsymbol{v}] \in \mathbb{R}^9 = [\boldsymbol{p} - \hat{\boldsymbol{p}}; \Phi^{-1}(\hat{\boldsymbol{q}}^{-1} \otimes \boldsymbol{q}); \boldsymbol{v} - \hat{\boldsymbol{v}}]$.

Assuming the robot's body has an IMU that outputs bias-free linear acceleration $\boldsymbol{a} \in \mathbb{R}^3$ and angular velocity $\boldsymbol{\omega} \in \mathbb{R}^3$ at time k, The discrete error-state process dynamics $\delta\boldsymbol{x}_{k+1} = f(\delta\boldsymbol{x}_k, \boldsymbol{a}, \boldsymbol{\omega}, \boldsymbol{n})$ are [30]:

$$\delta\boldsymbol{p}_{k+1} = \delta\boldsymbol{p}_k + \delta\boldsymbol{v}_k\Delta t + \boldsymbol{n}_v \tag{5}$$

$$\delta\boldsymbol{\theta}_{k+1} = (I - \lfloor\boldsymbol{\omega}\Delta t\rfloor^\times)\delta\boldsymbol{\theta}_k + \boldsymbol{n}_\omega \tag{6}$$

$$\delta\boldsymbol{v}_{k+1} = \delta\boldsymbol{v}_k - A(\boldsymbol{q})\lfloor\boldsymbol{a}\Delta t\rfloor^\times\delta\boldsymbol{\theta}_k + A(\boldsymbol{q})\boldsymbol{n}_a, \tag{7}$$

where Δt is the time between two IMU readings and $\boldsymbol{n} = [\boldsymbol{n}_v; \boldsymbol{n}_\omega; \boldsymbol{n}_a]$ contains random noise sampled from a Gaussian distribution. We refer readers to [30] for a detailed derivation of the process dynamics.

In addition to an IMU, other sensors may provide observations of the robot's state. For example, a motion-capture system or cameras can measure the robot's pose [10]. We use $\boldsymbol{z}_k = h(\boldsymbol{x}_k) + \boldsymbol{n}_r$ to denote such noisy sensor measurements, where $\boldsymbol{n}_r$ is assumed to be Gaussian noise.

Assuming an estimate of the current state $\hat{\boldsymbol{x}}$ and its covariance P_k, the error-state KF uses linearizations of the state dynamics f and the measurement model h,

$$F_x = \frac{\partial f}{\partial \delta\boldsymbol{x}}\bigg|_{\hat{\boldsymbol{x}}, \boldsymbol{\omega}, \boldsymbol{a}}, \; F_n = \frac{\partial f}{\partial \boldsymbol{n}}\bigg|_{\hat{\boldsymbol{x}}, \boldsymbol{\omega}, \boldsymbol{a}}, \text{ and } H = \frac{\partial h}{\partial \boldsymbol{x}}|_{\hat{\boldsymbol{x}}}, \tag{8}$$

to propagate the covariance forward in time, [30]

$$P_{k+1|k} \leftarrow F_x P_k F_x^\top + F_n \Sigma F_n, \tag{9}$$

where Σ is the covariance of $\boldsymbol{n}$. This is often called the *process update*. Next, the Kalman gain is calculated [31] and a *measurement update* is performed:

$$K = P_{k+1|k}H^\top(HP_{k+1|k}H^\top + Q)^{-1}, \tag{10}$$

$$\delta x = K[\boldsymbol{z}_k - h(\hat{\boldsymbol{x}}_k)], \tag{11}$$

$$P_{k+1} \leftarrow (I - KH)P_{k+1|k}, \tag{12}$$

where Q is the covariance of the measurement noise $\boldsymbol{n}_r$. Finally we update the state estimate using the error state:

$$\hat{\boldsymbol{p}}_{k+1} \leftarrow \hat{\boldsymbol{p}}_k + \delta\hat{\boldsymbol{p}}, \tag{13}$$

$$\hat{\boldsymbol{q}}_{k+1} \leftarrow L(\hat{\boldsymbol{q}}_k)\Phi(\delta\boldsymbol{\theta}), \tag{14}$$

$$\hat{\boldsymbol{v}}_{k+1} \leftarrow \hat{\boldsymbol{v}_k} + \delta\hat{\boldsymbol{p}}. \tag{15}$$

C. Sliding-Window Optimization-Based State Estimation

Sliding-window or receding-horizon optimization-based state estimators are preferred when a vision sensor is involved because handling large numbers of camera image features is difficult in a KF [32]. The key idea of sliding-window optimization is to construct a nonlinear least-squares problem at time k

$$\min_{\mathscr{X}} \left\{ \sum_i \left\| \mathbf{r}_i(\mathscr{X}, \mathscr{Z}) \right\|^2_{\Sigma_i} \right\}, \tag{16}$$

where $\mathscr{X}$ is a set of robot states at past N time steps and $\mathscr{Z}$ is a set of past sensor measurements, and $\mathbf{r}_i(\mathscr{X}, \mathscr{Z})$ is a residual or "innovation" function based on the same measurement model used in the KF.

D. Forward Kinematics & Leg Odometry Velocity

In this section we review the forward kinematics and describe how to infer the body velocity from the kinematic functions. For the jth leg of a legged robot, we define $\boldsymbol{\phi}$ as a vector containing all joint angles. Joint angle velocities are put in $\dot{\boldsymbol{\phi}}$. The forward kinematics function is denoted as $\boldsymbol{p}_f = g(\boldsymbol{\phi}) \in \mathbb{R}^3$, whose output is the foot position in the robot body frame. The derivative of this equation leads to the Jacobian matrix $J(\boldsymbol{\phi})$ that maps $\dot{\boldsymbol{\phi}}$ into the foot's linear velocity in the body frame:

$$\boldsymbol{v}_f = \dot{\boldsymbol{p}}_f = J(\boldsymbol{\phi})\dot{\boldsymbol{\phi}}. \tag{17}$$

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 $\boldsymbol{p}_f^w$ 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 $\boldsymbol{\phi}$:

$$\boldsymbol{p}_f^w = \boldsymbol{p} + A(\boldsymbol{q})\boldsymbol{p_f} = \boldsymbol{p} + A(\boldsymbol{q})g(\boldsymbol{\phi}). \tag{18}$$

Let the time derivative of $\boldsymbol{p}_f^w$ be $\boldsymbol{v}_f^w$. The no-slip assumption means $\boldsymbol{v}_f^w = 0$. Therefore, by differentiating (18), we have

$$0 = \boldsymbol{v}_f^w = \dot{\boldsymbol{p}}_f^w = \dot{\boldsymbol{p}} + A(\boldsymbol{q})\frac{d}{dt}g(\boldsymbol{\phi}) + \frac{d}{dt}A(\boldsymbol{q})g(\boldsymbol{\phi}). \tag{19}$$

It is shown in [4] that $\frac{d}{dt}A(\boldsymbol{q}) = A(\boldsymbol{q})\lfloor\boldsymbol{\omega}\rfloor^{\times}$, and we defined $\boldsymbol{v} = \dot{\boldsymbol{p}}$. Therefore from (19) we derive an expression for the body velocity in the world frame:

$$\boldsymbol{v} = -A(\boldsymbol{q})[J(\boldsymbol{\phi})\dot{\boldsymbol{\phi}} + \lfloor\boldsymbol{\omega}\rfloor^{\times} g(\boldsymbol{\phi})]. \tag{20}$$

This velocity is called the LO velocity because its integration is the body displacement [33].

IV. Technical Approach

The key idea underlying our technical approach is to treat the LO velocity (20) as a measurement of the robot body's velocity that is dependent on a set of kinematic parameters $\boldsymbol{\rho}$ for each leg. We then append $\boldsymbol{\rho}$ to the filter state. The dimension of $\boldsymbol{\rho}$ depends on the kinematic structure of the leg. For example, if we are estimating calf leg lengths of a quadruped robot, $\boldsymbol{\rho} = [l_{c1}; l_{c2}; l_{c3}; l_{c4}] \in \mathbb{R}^4$, where l_{ci} is the calf length of leg i. If both the calf and the thigh leg lengths need to be considered, then $\boldsymbol{\rho} = [\ldots, l_{ci}; l_{ti}, \ldots]$, for $i = 1, 2, 3, 4$, has dimension 8. We also write ρ_i to indicate the kinematic parameters related to leg i.

A. Body Velocity Measurement Model

We modify (20) to explicitly include kinematics parameters and sensor noise. For leg i, assuming the joint encoders on the leg measure joint angles $\boldsymbol{\phi}_i$ and angular velocities $\dot{\boldsymbol{\phi}}_i$ with additive Gaussian noise Gaussian $\boldsymbol{n}_{\phi}$ and $\boldsymbol{n}_{\dot{\phi}}$, repsectively, then the true LO velocity in the world frame is

$$\begin{aligned}\boldsymbol{v}_{m,i} = &- A(\boldsymbol{q})[J(\boldsymbol{\phi}_i - \boldsymbol{n}_{\phi}, \boldsymbol{\rho}_i)(\dot{\boldsymbol{\phi}}_i - \boldsymbol{n}_{\dot{\phi}}) \\ &- \lfloor\boldsymbol{\omega}\rfloor^{\times} g(\boldsymbol{\phi}_i - \boldsymbol{n}_{\phi}, \boldsymbol{\rho}_i)].\end{aligned} \tag{21}$$

We also define an estimated LO velocity as

$$\hat{\boldsymbol{v}}_{m,i} = -A(\hat{\boldsymbol{q}})[J(\boldsymbol{\phi}_i, \hat{\boldsymbol{\rho}}_i)\dot{\boldsymbol{\phi}}_i - \lfloor\boldsymbol{\omega}\rfloor^{\times} g(\boldsymbol{\phi}_i, \hat{\boldsymbol{\rho}}_i)]. \tag{22}$$

When the leg i has non-slipping contact with the ground, it contributes to a measurement model

$$\begin{aligned}\boldsymbol{z}_{leg} &= h_{leg}(\hat{\boldsymbol{x}}, \boldsymbol{\omega}, \boldsymbol{\phi}, \dot{\boldsymbol{\phi}}) + \boldsymbol{n}_c \\ &= \hat{\boldsymbol{v}} - \sum_i c_i \hat{\boldsymbol{v}}_{m,i} + \boldsymbol{n}_l(\boldsymbol{n}_{\phi}, \boldsymbol{n}_{\dot{\phi}}) + \boldsymbol{n}_c,\end{aligned} \tag{23}$$

where $\boldsymbol{n}_l$ is a noise function related to joint encoder noise, which can be derived from linearizing (21). We assume each foot of the legged robot has a contact detector [2], [34] that generates a binary contact flag c_i. Therefore, $c_i = 1$ means the foot has nonzero velocity relative to the ground. Otherwise, the non-slipping assumption of (22) is invalid. $\boldsymbol{n}_c$ is a Gaussian measurement noise whose variance is a tunable hyper-parameter.

B. Kalman Filter Kinematics Calibration

To achieve kinematics calibration using the error-state KF, we use a new process dynamics model and add the body velocity measurement (23) to the measurement model.

In the process dynamics, in addition to (5), (6), and (7), we model the evolution of $\delta\boldsymbol{\rho}$ as a random-walk process,

$$\delta\boldsymbol{\rho}_{k+1} = \delta\boldsymbol{\rho}_k + \boldsymbol{n}_{\rho}, \tag{24}$$

where $\boldsymbol{n}_{\rho}$ is a Gaussian white noise.

For the measurement model, we calculate a measurement vector as $\boldsymbol{z} = [\boldsymbol{z}_{mocap}; \boldsymbol{z}_{cam}; \boldsymbol{z}_{leg}]$, where $\boldsymbol{z}_{mocap}$ and $\boldsymbol{z}_{cam}$ are measurements from a motion-capture system or camera. $\boldsymbol{z}_{leg}$ is described in (23). The algorithm then proceeds as in Section III-B.

C. Observability Analysis

Prior research has shown that robot pose and velocity are observable when the measurement model contains information from a motion-capture system or camera [10], [15]. Therefore, we only focus on the observability of the kinematics parameters $\boldsymbol{\rho}$ in this section. We also note that our observability analysis applies to both EKF and sliding-window estimators.

Neglecting sensor noise, the dynamics and observation model for a system with only leg measurements can be written as,

$$\begin{aligned}\dot{\boldsymbol{x}} &= f_c(\boldsymbol{x}, \boldsymbol{a}, \boldsymbol{\omega}) \\ \boldsymbol{z} &= h(\boldsymbol{x}, \boldsymbol{\phi}, \dot{\boldsymbol{\phi}}),\end{aligned} \tag{25}$$

where f_c is the continuous state process dynamics (closely related to the error-state dynamics (5), (6), and (7)); h is the measurement model (23) considering just a single leg; and $\boldsymbol{a}$, $\boldsymbol{\omega}$, $\boldsymbol{\phi}$, and $\dot{\boldsymbol{\phi}}$ are IMU acceleration, IMU angular velocity, joint angles, and joint angle velocities respectively. We then compute the observability Gramian [35], [36]

$$\mathscr{W}(\boldsymbol{x}_0) = \int_0^T \Phi^{\top}(t) H^{\top}(\boldsymbol{x}_t) H(\boldsymbol{x}_t) \Phi(t) \mathrm{d}t, \tag{26}$$

where $\Phi(t)$ is the state transition matrix associated with the linearized dynamics:

$$\dot{\Phi}(t) = F_x(\boldsymbol{x}_t)\Phi(t), \;\; \Phi(0) = I. \tag{27}$$

If $\mathscr{W}(\boldsymbol{x}_0)$ is positive definite along the trajectory from $\boldsymbol{x}_0$ to $\boldsymbol{x}_T$, the system is locally observable [36].

For the Unitree A1 quadruped, its important kinematic parameters are indicated in Fig. 2. o_x, o_y are offsets distances between the robot COM and leg base. d is an offset between motor 2 and 3. l_t is the upper leg (thigh) length and l_c is the lower leg (calf) length. The analytical form of the forward kinematics function is provided in the Appendix. Among these parameters, we may choose to calibrate $\boldsymbol{\rho} = [l_c]$ (just the calf length) or $\boldsymbol{\rho} = [l_t; l_c]$ (both the calf and the thigh).

We analyze how parameter $\boldsymbol{\rho}$ is related to the observability gramian by expanding blocks in $\mathscr{W}(\boldsymbol{x}_0)$ analytically. From (5), (6), (7), and (24), we get

$$F_x = \begin{bmatrix} I & 0 & I\Delta t & 0 \\ 0 & I - \lfloor\boldsymbol{\omega}\Delta t\rfloor^{\times} & 0 & 0 \\ 0 & -A(q)\lfloor\boldsymbol{a}\Delta t\rfloor^{\times} & I & 0 \\ 0 & 0 & 0 & I \end{bmatrix}. \tag{28}$$

From (28) and (27), $\Phi(t)$ is always in the form of

$$\Phi(t) = \begin{bmatrix} I & * & * & 0 \\ 0 & * & 0 & 0 \\ 0 & * & I & 0 \\ 0 & 0 & 0 & I \end{bmatrix}, \tag{29}$$

where "$*$"s are nonzero block terms that are not relevant to our discussion. We linearize (23) to compute,

$$H_{leg} = \begin{bmatrix} \mathbf{0} & A(\hat{\boldsymbol{q}})\lfloor J\dot{\boldsymbol{\phi}} - \lfloor \boldsymbol{\omega} \rfloor^{\times} g \rfloor^{\times} & I & -A(\hat{\boldsymbol{q}})D \end{bmatrix}, \tag{30}$$

where,

$$D = (\dot{\boldsymbol{\phi}}^{\top} \widehat{\otimes} I_3) \frac{\partial vec(J(\boldsymbol{\phi}, \hat{\boldsymbol{\rho}}))}{\partial \hat{\boldsymbol{\rho}}} + \lfloor \boldsymbol{\omega} \rfloor^{\times} \frac{\partial g(\boldsymbol{\phi}, \hat{\boldsymbol{\rho}})}{\partial \hat{\boldsymbol{\rho}}}, \tag{31}$$

and the $vec(\cdot)$ operator returns a column vector by stacking the columns of the input matrix [37], and the $\widehat{\otimes}$ is the Kronecker product [37].

Plugging (29) and (30) into (26), we can see the last block of the integrand of $\mathscr{W}(\boldsymbol{x}_0)$ is $D^{\top}D$. Therefore, a sufficient condition for fully observable $\boldsymbol{\rho}$ is the null space of D is empty or, equivalently, that D has full column rank. An immediate conclusion we can draw is $\boldsymbol{\omega}$ and $\dot{\boldsymbol{\phi}}$ cannot both be zero, otherwise D will become a zero matrix. Therefore the robot cannot stand still or trot in place, since the joint velocities of stance legs will be close to zero. The rank condition of D also depends on the forward kinematics function, which is problem specific since $g(\boldsymbol{\phi}, \boldsymbol{\rho})$ may have different forms depending the robot leg structure. In the Appendix we show the forward kinematics of the Unitree A1 robot and how to calculate D. When $\boldsymbol{\rho} = [l_c]$ or $\boldsymbol{\rho} = [l_t; l_c]$, D has full rank so $\boldsymbol{\rho}$ is observable.

The observability analysis also suggests we trust the body velocity measurement model less when D is near singular. We change the covariance σ_c of the noise term $\boldsymbol{n}_c$ in (23) to be related to the mean of singular values of D.

$$\sigma_c = \sigma_0 + \frac{\alpha_1}{1 + \exp(\alpha_2(mean(SVD(D)) - \alpha_3))} \tag{32}$$

where σ_0 is a constant term, the second term is a logistic regressor [38] that assigns large value to D close to singular, which is equivalent to the local unobservablility index (LUI) proposed in [36]. Thus we call it LUI noise. This noise function can prevent parameter estimation fluctuation. We will study its effect in Section V.

V. Experiments

All of our hardware and simulation experiments are based on a Unitree A1 [8] robot. We first verify that the algorithm is able to converge to unbiased parameter estimates in simulation. The simulated robot has the same leg structure as that on the A1 robot, but kinematic parameter values are varied for testing. We then perform hardware experiments on a real A1 to demonstrate the practical performance of the algorithm. MATLAB implementations of the error-state KF and the sensor data we collected are available on GitHub[1].

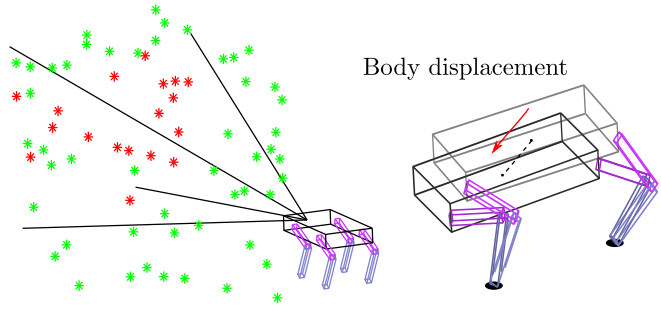

Fig. 3. **Left:** The simulated robot and environment landmark locations. **Right:** In the simulation we focus on analyzing how state is estimated within one gait cycle, during which the body shifts a small distance, feet stand on the ground without moving, and joints change configuration accordingly.

A. Simulation

We implement a simulator in MATLAB to generate simulated sensor data. We assume a periodic gait and known initial and final poses of the robot at the beginning and end of each gait cycle. We also assume perfect contact knowledge so that, during this gait cycle, contact feet have known fixed world positions. We use cubic Hermite splines, which are twice differentiable, to interpolate positions, and quaternion SLERP [39] to interpolate orientations. Therefore, we can query the robot's body position, orientation, velocity, and acceleration at any time in the gait cycle. From these quantities, we can calculate body-frame acceleration and angular velocities to generate simulated IMU data. Additionally, by using inverse kinematics, we can calculate the joint angles of a leg given a body position and a foot position. We also include a camera based on the pin-hole model [40] to observe landmarks with known locations in the environment. We generate camera observations by projecting landmarks onto the camera image frame [10]. Finally we add random noise to all simulated sensor data.

We run an error-state KF with the camera and the leg measurement models on the simulated data. Fig. 4 shows the calf length estimation result for a 0.1 s body trajectory with a linear displacement of $(0.1\text{ m}, 0.1\text{ m}, 0.05\text{ m})$ and orientation displacement of 5 degrees about the pitch, roll, and yaw axes. The KF state includes robot body pose, velocity, and calf lengths l_c of all legs. Even if the initial l_c values have large errors, the filter converges to ground truth values quickly with final errors less than 0.01 m. The detailed setup of the error-state KF can be seen in the open source codebase.

B. Error-State KF Hardware Experiment

We test our calibration method using sensor data from an actual A1 robot. Its sensors include one IMU, 12 joint encoders and 4 ft contact sensors. We implement a robot controller in C++

[1][Online]. Available: https://github.com/ShuoYangRobotics/legged-kinematics-calibration

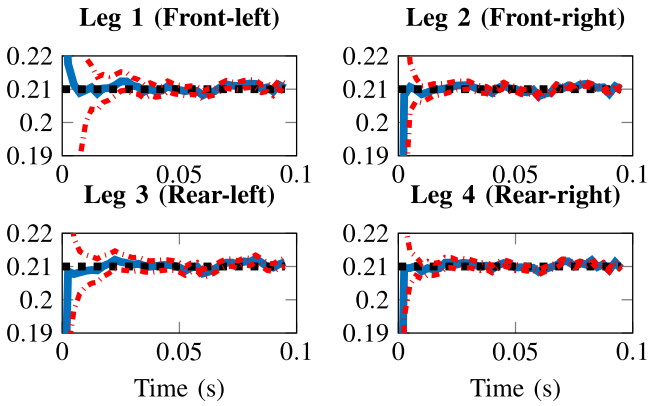

Fig. 4. The calf length estimation result using simulation data. $\boldsymbol{\rho} = [l_c]$. In all plots, blue lines are estimated length. Red dash lines show the 3σ uncertainty envelope. Black dash lines indicate the ground truth length 0.21 m for reference. All estimations converge to ground truth quickly with final errors < 0.01 m.

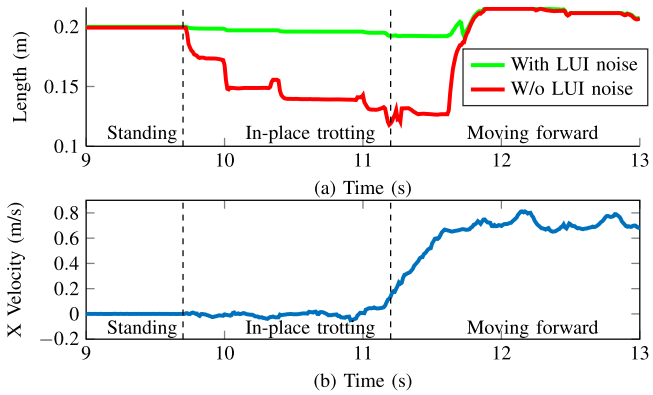

Fig. 5. (a) The calf length estimation for leg 1 during two calibration runs. The green line comes from a filter that has the LUI noise term (32), while the red line is generated by a filter does not have the term. Both filters have the same other configurations. Black dash lines indicate time instances when the robot changes behavior modes. During in-place trotting the red line drifts significantly. (b) Velocity profile in each mode. The robot has small body velocity hence small joint angle velocities when trotting. According to (31), the observability matrix is very close to singular so the measurement update is inaccurate.

following [3]. The robot moves in an arena equipped with an OptiTrack motion capture system, which provides high-quality body pose data. Robot sensor data and motion-capture data with timestamps are recorded as datasets. Although our filter can easily be run in real time, we perform all experiments offline so that we can replay the sensor datasets and run the filter with different settings. We refer interested readers to our open-source implementation for implementation details.

1) Calibration During Standing Up: In the first experiment, we record data while the robot stands up from a crouched pose. All feet are always in contact with the ground. The process is repeated for four trials and four standing-up datasets are collected. We then run the error-state KF with kinematics calibration on each dataset three times with different initial values $l_c = 0.1$ m, 0.2 m, and 0.3 m. In Fig. 1(c), we compare estimated values of l_c against time for each run on one dataset. In all three runs, the final value reaches around 0.21 m after about 3 s. This calibrated value is roughly consistent with the CAD model value of 0.20 m and the foot sensor head radius of 0.02 m, and implies that the soft, deformable foot is compressed to half of its original size under the robot's weight.

2) Calibration During Walking: We move the robot on flat ground to examine how kinematics calibration performs during walking. We collect ten datasets with the robot moving at different speeds and different total travel distances ranging from 5 m to 15 m. The calibration results using one of the datasets is shown in Fig. 6(a). Initially all leg calf lengths are set to 0.2 m. After the robot starts to walk, the leg length estimation fluctuates between 0.19m-0.23 m, the range is larger than the longest possible leg length in CAD model (0.22 m). Comparing Fig. 6(a), Fig. 6(b), and the experiment videos, the maximum leg length happens when the robot moves forward with relatively high speed, and the robot feet have rolling contact with the ground. The rolling contact is equivalent to a slightly longer leg with fixed point contact.

3) LUI Noise Ablation Study: In Section IV-C we present an LUI noise term (32). In Fig. 5 we show that, without this term (red line), the calf-length estimation drifts quickly when the robot is doing in-place trotting, since the observability matrix is poorly conditioned. When the LUI noise term is included, the estimation does not change much, as expected.

TABLE I
THE TABLE SHOWS THE AVERAGE PEFORMANCE METRICS OF TEN DATASETS AND THE IMPROVEMENT OF USING CALIBRATED LENGTH. MAX POS DRIFT IS THE MAXIMUM DEVIATION OF ESTIMATED POSITION FROM THE GROUND TRUTH. FINAL POS DRIFT IS THE POSITION DEVIATION AT THE END OF THE TRAVELING TRAJECTORY

	w/o calib	with calib	Improvement
Vel MSE	0.0023	0.0022	4.3%
Pos MSE	0.0070	0.0016	77.1%
Max Pos Drift	15.0cm	6.2cm	58.6%
Final Pos Drift	7.1cm	4.1cm	42.3%

4) Position Drift Reduction: We show that adding kinematics calibration to a baseline IMU and leg odometry filter can dramatically reduce position estimation drift. Our baseline filter follows Section III-B, with the IMU driving the process model and leg odometry captured by the measurement model (23). We also treat body orientation, as observed by the motion-capture system, as a known quantity following [3]. The baseline filter always uses a fixed leg length of 0.20 m (referred to as "KF w/o calib" in the figure legend). Fig. 6(b) and Fig. 6(c) compare the estimated X-direction velocity and position using either fixed length or calibrated length shown in Fig. 6(a). It can be seen that the KF with calibrated leg length achieves an order of magnitude better precision than that using fixed length. Table I summarizes the mean-squared error in the position and velocity estimates, final position drifts, and maximum position drifts across the ten datasets. The kinematics calibration significantly improves position estimation accuracy in all cases by providing the estimator with time-varying kinematic parameters.

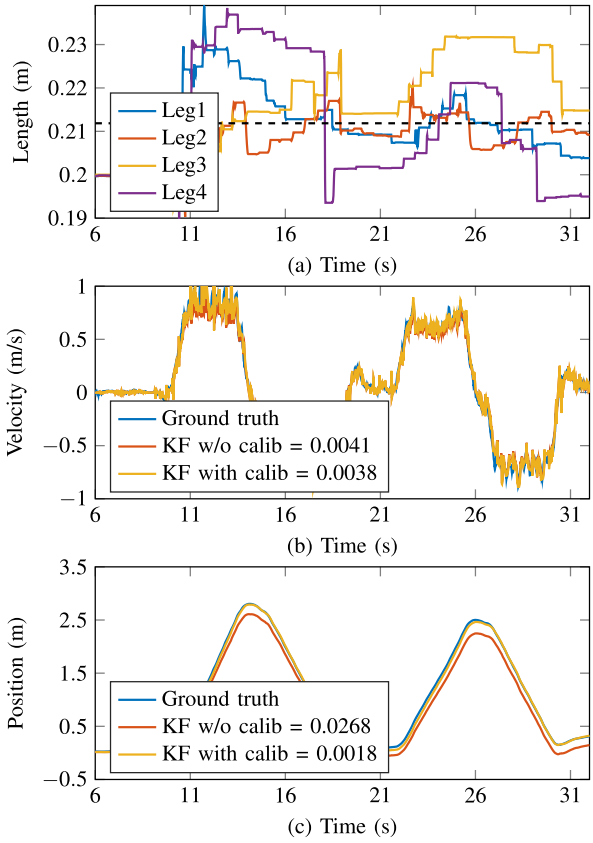

Fig. 6. Results of a hardware dataset run. (a) Calibrated calf lengths of each leg. The black dash line indicates the mean value of all lengths (0.2113 m). (b) The velocity estimations using either fixed or calibrated length do not differ much. The mean square error (MSE) of them from the ground truth velocity are 0.0041 and 0.0038 respectively. (c) The KF with calibrated calf length has much smaller position drift than the KF using fixed calf length. The MSEs from the ground truth are 0.0018 and 0.0268 respectively.

C. Sliding-Window Estimator Hardware Experiment

We also test our measurement model in the context of an optimization-based sliding-window estimator. We add the kinematic parameters and the body velocity measurement model (23) to the open-source VINS-Fusion [11], one of the most popular sliding-window estimators. The modified estimator takes inputs from a single Intel Realsense D435 camera and other legged robot proprioceptive sensors as measurements. The total cost of the sensor hardware is less than $2000. We run the estimator on the standing-up datasets. The kinematic calibration can be done quickly when the robot stands up. Fig. 7 shows the estimated l_c values of different runs with different initial calf length values. The final mean length value after the robot finished standing up (6 s-11 s) is 0.215 m, which agrees with the previous experiment using motion-capture data. However, we observe larger variance with the sliding-window estimator. We attribute this difference to the use of visual sensors, which are less accurate than the motion capture system used in the EKF. This experimental result confirms that we can perform kinematics calibration within an optimization-based estimator using low-cost onboard sensors, while quantitative analysis remains future work.

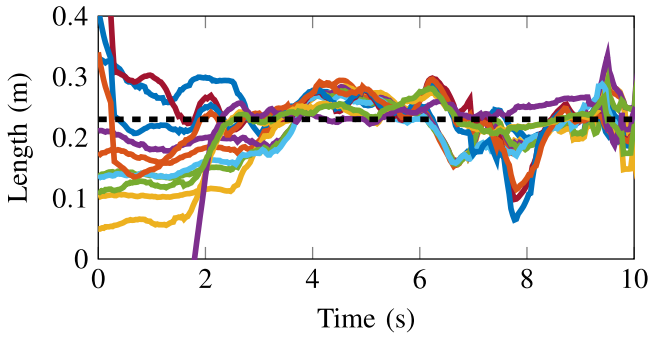

Fig. 7. The calf length calibration results using the optimization based state estimator (a modified VINS-Fusion algorithm). For each of the four standing-up datasets, we run the estimator three times with different random initial calf lengths. Each solid curve shows the estimated calf length of each run. The horizontal dash reference line indicates 0.215 m, the mean value of estimations between 6 s to 11 s.

VI. Conclusion & Future Work

We have presented a method to calibrate kinematic parameters of legged robots online. A detailed observability analysis, along with simulation and hardware experiments, validate our method. Kinematics calibration of deformable leg lengths results in more accurate body velocity estimation and, hence, significantly lower odometry drift. The calibration method can be easily integrated into standard state estimator formulations.

In future work, we will investigate kinematic parameter formulations that can better capture rolling contacts. We will also research whether jointly estimating robot states and kinematics parameters can achieve sub-centimeter calibration accuracy and reduce long term position estimation drift using the optimization based state estimator.

Appendix

The forward kinematics function g of a leg of a Unitree A1 robot with $\boldsymbol{\phi} = [\phi_1; \phi_2; \phi_3]$ and $\boldsymbol{\rho} = [l_c]$ is

$$g(\boldsymbol{\phi}, \boldsymbol{\rho}) = \begin{bmatrix} o_x - l_c s_{23} - l_t s_2 \\ o_y + dc_1 + l_t c_2 s_1 + l_c s_1 c_{23} \\ ds_1 - l_t c_1 c_2 - l_c c_1 c_{23} \end{bmatrix}, \tag{33}$$

where s_i denotes $\sin(\phi_i)$ and $c_i = \cos(\phi_i)$, where $i = 1, 2$. Also $s_{23} = \sin(\phi_2 + \phi_3)$ and $c_{23} = \cos(\phi_2 + \phi_3)$. The expression is derived using the product of exponentials (POE) method [4]. The Jacobian of g is

$$J(\boldsymbol{\phi}, \boldsymbol{\rho}) = \begin{bmatrix} 0 & -l_c c_{23} - l_t c_2 & -l_c c_{23} \\ l_t c_1 c_2 - ds_1 + l_c c_1 c_{23} & -s_1(l_c s_{23} + l_t s_2) & -l_c s_{23} s_1 \\ l_t c_2 s_1 + dc_1 + l_c s_1 c_{23} & c_1(l_c s_{23} + l_t s_2) & l_c s_{23} c_1 \end{bmatrix}, \tag{34}$$

It is easy to calculate their derivatives with respect to $\boldsymbol{\rho}$ through symbolic computation tools. So

$$\frac{\partial g(\boldsymbol{\phi},\boldsymbol{\rho})}{\partial \boldsymbol{\rho}} = \begin{bmatrix} -s_{23} \\ c_{23}s_1 \\ -c_{23}c_1 \end{bmatrix} \text{ and } \frac{\partial vec(J(\boldsymbol{\phi},\boldsymbol{\rho}))}{\partial \boldsymbol{\rho}} = \begin{bmatrix} 0 \\ c_{23}c_1 \\ c_{23}s_1 \\ -c_{23} \\ -s_{23}s_1 \\ s_{23}c_1 \\ -c_{23} \\ -s_{23}s_1 \\ s_{23}c_1 \end{bmatrix}.$$

If we let $\dot{\boldsymbol{\phi}} = [\dot{\phi}_1; \dot{\phi}_2; \dot{\phi}_3]$ and $\boldsymbol{\omega} = [\omega_1; \omega_2; \omega_3]$, then

$$D = \begin{bmatrix} -\dot{\phi}_2 c_{23} - \dot{\phi}_3 c_{23} - \omega_2 c_{23} c_1 - \omega_3 c_{23} s_1 \\ \dot{\phi}_1 c_{23} c_1 - \omega_3 s_{23} + \omega_1 c_{23} c_1 - \dot{\phi}_2 s_{23} s_1 - \dot{\phi}_3 s_{23} s_1 \\ \omega_2 s_{23} + \dot{\phi}_1 c_{23} s_1 + \dot{\phi}_2 s_{23} c_1 + \dot{\phi}_3 s_{23} c_1 + \omega_1 c_{23} s_1 \end{bmatrix}. \tag{35}$$

We can confirm $\boldsymbol{\rho}$ is observable because when $\dot{\boldsymbol{\phi}}$ and $\boldsymbol{\omega}$ are non-zero vectors, the rank of D is 1 regardless of the value of $\boldsymbol{\phi}$. Then $D^\top D$ is non-singular and the observability gramian will always be positive definite, thus $\boldsymbol{\rho}$ is observable. The same proof for $\boldsymbol{\rho} = [l_t; l_c]$ can show it is observable as well.

ACKNOWLEDGMENT

The authors would also like to thank Prof. Aaron Johnson for the use of his lab's motion capture system and the anonymous reviewers for their valuable feedback.

REFERENCES

[1] N. Ratliff, M. Zucker, J. A. Bagnell, and S. Srinivasa, "CHOMP: Gradient optimization techniques for efficient motion planning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2009, pp. 489–494.
[2] M. Bloesch *et al.*, "State estimation for legged robots - consistent fusion of leg kinematics and IMU," *Robotics*, vol. 17, pp. 17–24, 2013.
[3] 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 *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 2245–2252.
[4] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. Boca Raton, FL, USA: CRC, 2017.
[5] M. Camurri, M. Ramezani, S. Nobili, and M. Fallon, "Pronto: A multi-sensor state estimator for legged robots in real-world scenarios," *Front. Robot. AI*, vol. 7, 2020, Art. no. 68.
[6] M. Hutter *et al.*, "Anymal-a highly mobile and dynamic quadrupedal robot," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 38–44.
[7] D. Wisth, M. Camurri, and M. Fallon, "Robust legged robot state estimation using factor graph optimization," *IEEE Robot. Automat. Lett.*, vol. 4, no. 4, pp. 4507–4514, Oct. 2019.
[8] Unitree, "A1," 2021. Accessed: Sep. 10, 2021. [Online]. Available: https://www.unitree.com/products/a1/
[9] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, no. 7553, pp. 503–507, 2015.
[10] M. Li and A. I. Mourikis, "High-precision, consistent EKF-based visual-inertial odometry," *Int. J. Robot. Res.*, vol. 32, no. 6, pp. 690–711, 2013.
[11] T. Qin, P. Li, and S. Shen, "VINS-Mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Trans. Robot.*, vol. 34, no. 4, pp. 1004–1020, Aug. 2018.
[12] Z. Roth, B. Mooring, and B. Ravani, "An overview of robot calibration," *IEEE J. Robot. Autom.*, vol. 3, no. 5, pp. 377–385, Oct. 1987.
[13] W. S. Newman, C. E. Birkhimer, R. J. Horning, and A. T. Wilkey, "Calibration of a Motoman P8 robot based on laser tracking," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2000, vol. 4, pp. 3597–3602.
[14] K. Nickels, E. Huber, and M. DiCicco, "Hand-eye calibratilon using active vision," in *Proc. IEEE Aerosp. Conf.*, 2007, pp. 1–9.
[15] J. Kelly and G. S. Sukhatme, "Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration," *Int. J. Robot. Res.*, vol. 30, no. 1, pp. 56–79, 2011.
[16] T. Qin and S. Shen, "Online temporal calibration for monocular visual-inertial systems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 3662–3669.
[17] J. B. Moore, "Discrete-time fixed-lag smoothing algorithms," *Automatica*, vol. 9, no. 2, pp. 163–173, 1973.
[18] K. R. Muske, J. B. Rawlings, and J. H. Lee, "Receding horizon recursive state estimation," in *Proc. IEEE Amer. Control Conf.*, 1993, pp. 900–904.
[19] Y. Zhang, T. Zhang, and S. Huang, "Comparison of EKF based SLAM and optimization based SLAM algorithms," in *Proc. 13th IEEE Conf. Ind. Electron. Appl.*, 2018, pp. 1308–1313.
[20] R. Hartley, M. Ghaffari, R. M. Eustice, and J. W. Grizzle, "Contact-aided invariant extended Kalman filtering for robot state estimation," *Int. J. Robot. Res.*, vol. 39, no. 4, pp. 402–430, 2020.
[21] 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 *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018, pp. 3783–3790.
[22] D. Wisth, M. Camurri, and M. Fallon, "VILENS: Visual, inertial, lidar, and leg odometry for all-terrain legged robots," 2021, *arXiv:2107.07243*.
[23] D. Wisth, M. Camurri, and M. Fallon, "Preintegrated velocity bias estimation to overcome contact nonlinearities in legged robot odometry," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 392–398.
[24] F. Blöchliger, M. Blösch, P. Fankhauser, M. Hutter, and R. Siegwart, "Foot-eye calibration of legged robot kinematics," in *Advances in Cooperative Robotics*. Singapore: World Scientific, 2017, pp. 420–427.
[25] A. Reinke, M. Camurri, and C. Semini, "A factor graph approach to multi-camera extrinsic calibration on legged robots," in *Proc. 3rd IEEE Int. Conf. Robotic Comput.*, 2019, pp. 391–394.
[26] Y. Sun *et al.*, "Online learning of unknown dynamics for model-based controllers in legged locomotion," *IEEE Robot. Automat. Lett.*, vol. 6, no. 4, pp. 8442–8449, Oct. 2021.
[27] M. Vigne, A. El Khoury, M. Pétriaux, F. Di Meglio, and N. Petit, "MOVIE: A velocity-aided IMU attitude estimator for observing and controlling multiple deformations on legged robots," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 3969–3976, Apr. 2022.
[28] M. Bloesch, M. Hutter, C. Gehring, M. A. Hoepflinger, and R. Siegwart, "Kinematic batch calibration for legged robots," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2013, pp. 2542–2547.
[29] B. E. Jackson, K. Tracy, and Z. Manchester, "Planning with attitude," *IEEE Robot. Automat. Lett.*, vol. 6, no. 3, pp. 5658–5664, Jul. 2021.
[30] J. Sola, "Quaternion kinematics for the error-state Kalman filter," 2017, *arXiv:1711.02508*.
[31] P. S. Maybeck, *Stochastic Models, Estimation, and Control*, Cambridge, MA, USA: Academic Press, 1982.
[32] F. Dellaert and M. Kaess, "Factor graphs for robot perception," *Foundations Trends Robot.*, vol. 6, no. 1–2, pp. 1–139, 2017.
[33] P.-C. Lin, H. Komsuoglu, and D. E. Koditschek, "A leg configuration measurement system for full-body pose estimates in a hexapod robot," *IEEE Trans. Robot.*, vol. 21, no. 3, pp. 411–422, Jun. 2005.
[34] M. Camurri *et al.*, "Probabilistic contact estimation and impact detection for state estimation of quadruped robots," *IEEE Robot. Automat. Lett.*, vol. 2, no. 2, pp. 1023–1030, Apr. 2017.
[35] C.-T. Chen, *Linear System Theory and Design*. Oxford, U.K.: Oxford Univ. Press, 1999.
[36] A. J. Krener and K. Ide, "Measures of unobservability," in *Proc. 48th IEEE Conf. Decis. Control Held Jointly 28th Chin. Control Conf.*, 2009, pp. 6401–6406.
[37] J. R. Magnus and H. Neudecker, *Matrix Differential Calculus With Applications in Statistics and Econometrics*. Hoboken, NJ, USA: Wiley, 2019.
[38] D. R. Cox, "The regression analysis of binary sequences," *J. Roy. Stat. Society: Ser. B. (Methodological)*, vol. 20, no. 2, pp. 215–232, 1958.
[39] K. Shoemake, "Animating rotation with quaternion curves," in *Proc. 12th Annu. Conf. Comput. Graph. Interactive Techn.*, 1985, pp. 245–254.
[40] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge, U.K.: Cambridge Univ. Press, 2003.