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

Enhancing Leg Odometry in Legged Robots with Learned Contact Bias: An LSTM Recurrent Neural Network Approach

Yaru Gu[1], Ze Liu[2], and Ting Zou[1]

***Abstract*— To address the leg odometry drift caused by the non-stationary foot contact, this paper introduces a novel data-driven based leg odometry technique for legged robots. By leveraging a Long Short-Term Memory (LSTM) Recurrent Neural Network (RNN), the method learns the biases in the robot's foot contact locations from sequential IMU measurements and ground reaction forces (GRF). This learned contact bias is then incorporated into the state estimation process using a Kalman filter (KF), significantly improving the precision of leg odometry for legged robots operating in real time. This method, which combines deep learning approaches with conventional filtering techniques, is named the Deep Learning Kalman Filter (DLKF). The effectiveness of the DLKF is demonstrated through simulation and experimental trials using a Unitree Go1 robot across various challenging environments, including uneven terrain, slopes, and stairs, where foot slippage occurs frequently. Our results indicate an average 64.93% reduction in translational errors in leg odometry when the learned contact bias is applied. Further improvements are observed in a fused LiDAR and leg odometry state estimation system, especially in feature-deprived areas, indicating that the proposed leg odometry system can be easily fused with other sensor measurements to get a more precise state estimation.**

I. Introduction

State estimation is pivotal in facilitating various aspects of legged robot locomotion, including control, motion planning, and mapping. Without a precise estimate of its states, the robot will not be able to construct an accurate map of the environment, and thus leading to less ideal performance when planning a trajectory navigating through it.

This paper aims to improve the leg odometry of legged robots using a Kalman filter (KF) method with a particular focus on compensating the foot contact bias, which is learned from a deep neural network. The overview of the proposed method is illustrated in Fig. 1.

A. Motivation

Leg odometry involves determining the robot's states through the application of forward kinematics, utilizing readings from encoders. Contact states of the stance legs play a key role in this process. Previous studies have assumed that the contact point of a leg remains fixed during the stance phase and is subject to Gaussian noise [1], [2]. However, such adverse contact conditions as slippery and terrain deformation may break the assumption and induce non-stationary foot contacts. Leg flexibility and modeling errors [3] are other sources causing unreliable contact estimation and leg odometry drift. When integrating the leg odometry with other sensor measurements, the overall state estimate of a legged robot can be adversely affected by the leg odometry drift. According to [4], leg odometry drift tends to follow a roughly linear pattern. To compensate for this bias in leg odometry, the authors expanded the state to include velocity biases in leg odometry within a factor graph framework.

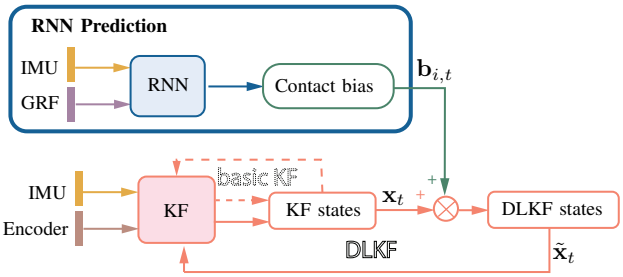

Fig. 1. **System Architecture Block Diagram:** An RNN-based model is developed to learn the contact bias ($\mathbf{b}_{i,t}$) for each leg i at time t, using data sequences from IMU and GRF inputs. The derived contact bias information is subsequently integrated into a KF framework to adjust the estimated states ($\mathbf{x}_t$), thereby significantly refining the leg odometry estimations ($\tilde{\mathbf{x}}_t$). This process, effectively blending deep learning techniques with traditional filtering methods, is identified as the Deep Learning Kalman Filter (DLKF).

Inspired by the body velocity biases in [4], we seek to improve the leg odometry performance by compensating the contact bias. However, the complexity of robot kinematics and dynamics, coupled with the variability of contact surfaces, makes it challenging to precisely model the contact bias mathematically. Therefore, using data driven methods in this domain becomes worthy exploring. In light of the proven effectiveness of the Long Short-Term Memory (LSTM) across various tasks [5], particularly with time series data, this paper proposes leveraging deep learning to complement traditional model-based leg odometry. As depicted in Fig. 1, the contact bias predicted from the inertial measurement unit (IMU) and ground reaction forces (GRF) measurements is subsequently incorporated into the KF estimated states, resulting in enhanced precision of the state estimation.

B. Contribution

The principal contributions of this work are threefold:

1. An LSTM-based deep neural network is developed to learn the contact location bias for each leg, in both stance and swing phases.

*This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) through a Discovery Grant (No. 213541) and the VP Startup Fund from Memorial University (No. 212948).

[1] Y. Gu and T. Zou (corresponding author) are with the Department of Mechanical and Mechatronics Engineering, Memorial University of Newfoundland, St. John's, A1B 3X5, Canada yarug@mun.ca tzou@mun.ca

[2]Z. Liu is with the School of Mechatronic Systems Engineering, Simon Fraser University, Burnaby, V5A 1S6, Canada zla286@sfu.ca

2. The acquired knowledge of contact bias is integrated into a conventional KF framework to improve the accuracy of leg odometry.

3. A series of comprehensive simulations and experimental trials have been conducted to verify the efficacy of our proposed leg odometry system. The outcomes demonstrate a substantial reduction in leg odometry drift with the incorporation of the learned contact bias, in comparison to standard KF estimates.

The remainder of this paper is structured as follows: Section II presents a review of the leg odometry and contact estimation for legged robots, alongside data-driven based state estimation techniques. Section III details the estimation of leg odometry using the basic KF and introduces our method for contact bias compensation. The deep learning framework used to learn the contact bias is described in Section IV. Section V and VI discusses the simulation and experimental results, respectively. Finally, Section VII provides the conclusions.

II. Related Work

A. Leg Odometry of Legged Robots

For legged robots, leg odometry is an important source of information for state estimation [6]. However, issues like foot slippage, terrain deformation, and modeling errors during movement can cause significant odometry drift [7].

Previous studies have aimed to improve the accuracy of leg odometry by incorporating data from additional sensors. These fusion methods generally fall into two categories: filtering and smoothing. Filtering techniques, such as the Kalman filter (KF) [8], [9], extended KF [6], unscented KF [10], and invariant extended KF [11], have been used to combine leg odometry with data from IMUs, LiDAR, or cameras. On the other hand, smoothing methods estimate robot states by maximizing the likelihood of all sensor data within a probabilistic graphical model framework [4], [12]–[14]. Despite these advances, enhancing leg odometry within the context of multisensor fusion continues to be beneficial.

One approach to reducing leg odometry drift is to calibrate kinematic parameters, ensuring a more accurate kinematic model. Bloesch et al. [15] introduced a batch optimization method for calibrating kinematic parameters (e.g., segment lengths, body dimensions) and other sensors' calibration parameters (e.g., accelerometer and gyroscope biases) by augmenting the robot states with these parameters. Yang et al. [3] performed online calibration of parameters (e.g., leg length) by comparing leg odometry based robot velocity measurements with more reliable velocity data from motion capture or visual odometry. Both studies assume static foot contact during body position or velocity computation from leg odometry. Yet, beyond modeling inaccuracies, foot slippage also significantly contributes to leg odometry errors.

To tackle potential slippage, some studies have estimated the velocity biases of leg odometry. Wisth et al. [12] developed a factor graph that incorporates data from cameras, IMUs, and leg odometry — obtained using an external filter from [16]. Extending this work, subsequent research [4] pre-integrated leg odometry within the factor graph, eliminating the need for an external filter. These studies convincingly show that the precision of state estimation can be enhanced by incorporating the velocity biases in the leg odometry. However, the measurement of the velocity bias term is modeled as a Gaussian random walk, due to the lack of an effective measuring system.

B. Contact Estimation

The implementations of leg odometry in the previous studies have assumed that the contact positions are stationary and subject to Gaussian noise [1], [2]. While this assumption can be compromised due to slippage, rolling contacts, and foot/ground deformation, researchers have focused on identifying the stance feet that are not fixed on the ground and reducing their effect on the state estimation.

For instance, [10] used a Mahalanobis distance based threshold to reject the measurements from legs without secure ground contact. Similarly, Kim et al. [17] detected and excluded slipping feet by comparing the estimated foot velocity with a tunable threshold. Hwangbo et al. [18] proposed a contact detector that integrates dynamics with differential and forward kinematics. Additionally, Bledt et al. [19] designed a discrete-time disturbance observer to estimate foot contact forces, assuming that all disturbance forces on the leg come from the foot.

Despite the application of various methods, a primary goal of contact estimation is to identify feet with unreliable contact and minimize their impact on leg odometry. Accurate contact information is essential for enhancing the precision of leg odometry. Nonetheless, in environments where foot slippage is frequent and severe, significant information loss may occur. Rather than merely discarding data from unreliable contacts, acquiring the contact bias is believed to substantially improve leg odometry.

C. Data-driven Based State Estimation

The use of data-driven methods in state estimation has recently become an area of growing interest. Deep neural networks (DNNs) have been used to model IMU systems and enhance inertial navigation performance [20]. For instance, a DNN was trained in [21] to simulate IMU measurement noise and estimate attitude states, which was tested on an unmanned aerial vehicle. Zhang et al. [22] utilized a recurrent neural network to learn the only observable IMU integration terms, in comparison to the previous studies which attempted to regress both observable or unobservable system terms [23], [24].

There are studies focusing on predicting foot-ground contact through data-driven methods. Lin et al. [1] implemented a DNN to learn the contact states of each foot. When integrating contact predictions into an invariant KF, however, they assumed non-slip foot contact, and contact velocity noise was characterized as white Gaussian noise. Camurri et al. [25] used a logistic regression classifier to learn ground reaction force thresholds, thus rejecting feet not in firm contact.

The dynamics behind contact location bias are too intricate for analytical modeling. Leveraging DNNs, our paper predicts contact location bias for state estimation enhancement. Existing literature shows a notable gap in using data-driven methods directly to model kinematic bias for state estimation. To our knowledge, this is the first study to employ a data-driven approach to learn contact location bias for state estimation in legged robots.

III. Leg Odometry Formulation

A. Notation and Coordinate Frames

In this paper, scalars are indicated with lowercase italics (e.g., a, b, c), and vectors and matrices are boldface lowercase (e.g., $\mathbf{a}, \mathbf{b}, \mathbf{c}$) and uppercase Roman (e.g., $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$), respectively. Uppercase calligraphics (e.g., $\mathcal{A}, \mathcal{B}, \mathcal{C}$) are reserved for coordinate frames. The critical coordinate frames include the world-fixed frame $\mathcal{W}$, the body-fixed frame $\mathcal{B}$, and the IMU frame $\mathcal{I}$. We use a subscript to indicate the coordinate frame in which the vector in question is expressed (e.g., $[\mathbf{a}]_{\mathcal{B}}$). For vectors expressed in the world frame, the subscript $\mathcal{W}$ is neglected for the sake of notation-simplicity.

B. Forward Kinematics

The joint positions and velocities can be accessed from the encoders integrated into the actuators. For the ith leg of a legged robot, we define the joint position vector as $\boldsymbol{\alpha}_i \in \mathbb{R}^j$, and the velocity vector $\dot{\boldsymbol{\alpha}}_i \in \mathbb{R}^j$, where j represents the number of joints in one leg. Based on the known leg kinematics, the foot position with respect to the robot base $[\mathbf{f}_i]_{\mathcal{B}} \in \mathbb{R}^3$ is a function of the joint position vector:

$$[\mathbf{f}_i]_{\mathcal{B}} = \mathrm{k}(\boldsymbol{\alpha}_i) \tag{1}$$

Deriving Eq. (1) yields the Jacobian matrix $\mathbf{J}(\boldsymbol{\alpha}_i)$, which converts joint velocities $\dot{\boldsymbol{\alpha}}_i$ into the foot's velocity relative to the body frame:

$$\left[\dot{\mathbf{f}}_i\right]_{\mathcal{B}} = \mathbf{J}(\boldsymbol{\alpha}_i)\dot{\boldsymbol{\alpha}}_i \tag{2}$$

C. IMU-Driven Leg Odometry

Leg odometry involves estimating the robot states with joint encoders and the forward kinematics. In this paper, we use the conventional Kalman filter to represent the leg odometry, similar to [8].

The robot states estimated from the leg odometry include robot's position $\mathbf{p} \in \mathbb{R}^3$ and velocity $\mathbf{v} \in \mathbb{R}^3$, both measured from the center of mass, as well as the foot contact positions $\mathbf{f}$ for all legs. Thus, the state vector of a legged robot with n legs at time t is defined as

$$\mathbf{x}_t := (\mathbf{p}_t; \mathbf{v}_t; \mathbf{f}_{1,t}; \ldots; \mathbf{f}_{n,t}) \tag{3}$$

1) Process Update: The uncertainties of the estimated states are represented by the covariance matrix $\boldsymbol{\Sigma}_t$. The predicted states $\bar{\mathbf{x}}_t$ and its covariance $\bar{\boldsymbol{\Sigma}}_t$ are calculated after updating the process model and before incorporating the measurements. The process model in the basic KF defines the state transition from the prior state $\mathbf{x}_{t-1}$ and the control input $\mathbf{u}_t$:

$$\bar{\mathbf{x}}_t = \mathrm{f}(\mathbf{x}_{t-1}, \mathbf{u}_t) + \boldsymbol{\varepsilon}_t \tag{4}$$

The random variable $\boldsymbol{\varepsilon}_t = [\boldsymbol{\varepsilon}_{p,t}; \boldsymbol{\varepsilon}_{v,t}; \boldsymbol{\varepsilon}_{f1,t}; \ldots; \boldsymbol{\varepsilon}_{fn,t}]$ represents the uncertainties in the state transition process, sampled from a Gaussian distribution with a zero mean and covariance matrix $\mathbf{R}_t$.

Assume the IMU on the robot body outputs bias-free acceleration $\mathbf{a}_t \in \mathbb{R}^3$ at time t, which will be used as the control input. The discrete process model can then be represented as:

$$\bar{\mathbf{p}}_t = \mathbf{p}_{t-1} + \mathbf{v}_{t-1}\triangle t + \boldsymbol{\varepsilon}_{p,t} \tag{5}$$

$$\bar{\mathbf{v}}_t = \mathbf{v}_{t-1} + \mathbf{a}_t\triangle t + \boldsymbol{\varepsilon}_{v,t} \tag{6}$$

$$\bar{\mathbf{f}}_{i,t} = \mathbf{f}_{i,t-1} + \boldsymbol{\varepsilon}_{f,i,t}, \quad i \in [1, n] \tag{7}$$

$\bar{\boldsymbol{\Sigma}}_t$ is updated by

$$\bar{\boldsymbol{\Sigma}}_t = \mathbf{A}_t\boldsymbol{\Sigma}_{t-1}\mathbf{A}_t^\top + \mathbf{R}_t \tag{8}$$

where

$$\mathbf{A}_t = \frac{\partial \mathrm{f}(\mathbf{x}_{t-1}, \mathbf{u}_t)}{\partial \mathbf{x}_{t-1}} \tag{9}$$

2) Measurement Update: The measurement model of the KF is

$$\mathbf{z}_t = \mathbf{C}_t\mathbf{x}_t + \boldsymbol{\delta}_t \tag{10}$$

Here $\mathbf{C}_t$ is a matrix projecting the state vector into the measurement vector $\mathbf{z}_t$. The measurement noise is denoted with the vector $\boldsymbol{\delta}_t$, having a multivariate Gaussian distribution with zero mean and covariance $\mathbf{P}_t$.

The measurement vector of leg odometry is composed of the relative foot position of each leg with respect to the body and the body velocity. The foot position in the world frame can be obtained from:

$$\mathbf{f}_{i,t} = \mathbf{p}_t + \mathbf{Q}_t\left[\mathbf{f}_{i,t}\right]_{\mathcal{B}} \tag{11}$$

where $\mathbf{Q}_t \in \mathrm{SO}(3)$ is the rotation matrix rotating a vector in the body frame into the world frame.

Given $\dot{\mathbf{Q}}_t = \mathbf{Q}_t\boldsymbol{\Omega}_t$, where $\boldsymbol{\Omega}_t$ is the cross-product matrix of the angular velocity vector measured in the body frame [26], differentiating both sides of Eq. (11) leads to

$$\dot{\mathbf{f}}_{i,t} = \mathbf{v}_t + \mathbf{Q}_t\left(\boldsymbol{\Omega}_t\left[\mathbf{f}_{i,t}\right]_{\mathcal{B}} + \left[\dot{\mathbf{f}}_{i,t}\right]_{\mathcal{B}}\right) \tag{12}$$

Under the non-slippage assumption, the velocity of a stance foot $\dot{\mathbf{f}}_{i,t} = \mathbf{0}$. Considering Eqs. (1) and (2), and after rearranging Eqs. (11) and (12), we can express the leg odometry's measurement model as follows:

$$\mathbf{Q}_t\mathrm{k}(\boldsymbol{\alpha}_{i,t}) = \mathbf{f}_{i,t} - \mathbf{p}_t \tag{13}$$

$$-\mathbf{J}(\boldsymbol{\alpha}_{i,t})\dot{\boldsymbol{\alpha}}_{i,t} - \boldsymbol{\Omega}_t\mathrm{k}(\boldsymbol{\alpha}_{i,t}) = \mathbf{Q}_t^\top\mathbf{v}_t \tag{14}$$

The Kalman gain $\mathbf{K}_t$ can then be updated by

$$\mathbf{K}_t = \bar{\boldsymbol{\Sigma}}_t\mathbf{C}_t^\top(\mathbf{C}_t\bar{\boldsymbol{\Sigma}}_t\mathbf{C}_t^\top + \mathbf{P}_t)^{-1} \tag{15}$$

Finally, the robot state and its covariance can be updated by the measurement update process:

$$\mathbf{x}_t = \bar{\mathbf{x}}_t + \mathbf{K}_t(\mathbf{z}_t - \mathbf{C}_t\bar{\mathbf{x}}_t) \tag{16}$$

$$\mathbf{\Sigma}_t = (\mathbf{I} - \mathbf{K}_t\mathbf{C}_t)\bar{\mathbf{\Sigma}}_t \tag{17}$$

where $\mathbf{I}$ is an identity matrix.

D. Leg Odometry with Contact Bias

The foot contact bias is defined by the difference between the contact location ground truth and the estimated contact location from the basic KF in Eq. (16). A deep neural network will be trained (see Section IV) to capture the pattern of these errors. The learned contact bias $\mathbf{b}_{i,t}$ for leg i is incorporated into the KF framework by compensating the estimated foot positions from Eq. (16) as

$$\tilde{\mathbf{f}}_{i,t} = \mathbf{f}_{i,t} + \boldsymbol{\beta}^\top \mathbf{b}_{i,t} \tag{18}$$

where $\boldsymbol{\beta}$ indicates the weights. The newly obtained contact estimates $\tilde{\mathbf{f}}_{i,t}$ will facilitate the updating of other states, thereby enabling the deep learning-based state estimation represented as $\tilde{\mathbf{x}}_t$, as illustrated in Fig. 1.

IV. Deep Learning Architecture

In prior research, leg odometry has been computed considering the legs in stance phases [6], [8], [9]. However, there is potential for bias in the contact position at the very moment a leg transitions to a stance leg, influenced by inaccuracies accumulated during the leg's preceding swing phase. Our methodology advances the field by not only learning the contact bias during the stance phase but also addressing the bias as the leg transitions from swing to stance. To this end, we train two distinct Recurrent Neural Network (RNN) models tailored to both the stance and swing phases, aiming to provide a comprehensive analysis of contact bias.

A. Model Architecture

For leg i at time t, its contact bias $\mathbf{b}_{i,t}$ can be regarded as an aggregation of errors from the state estimation over a given time period. Factors such as foot slippage, modeling inaccuracies, or deformations of the foot occurring within this interval all contribute to $\mathbf{b}_{i,t}$. Given this temporal nature, an LSTM model is adept at capturing these dependencies and the associated biases. The LSTM layer is configured with 64 units and is designed to receive input data in the form of sequences (time steps) of features. The output from the LSTM layer is fed to a dense output layer with three units. This RNN model's architecture, replicated for both swing and stance phases, is depicted in Fig. 2. Each phase is modeled separately but with identical configuration settings.

B. Data Structure for Input and Output

The RNN models for swing and stance phases output a three-dimensional vector $\mathbf{b}_{i,t}$, which represents the contact bias for leg i across N time steps, measured at time t. Model training employs contact biases derived from discrepancies between ground truth and KF estimates.

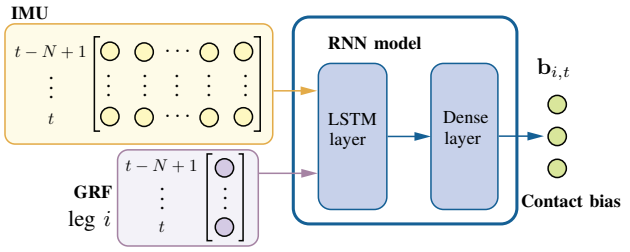

Fig. 2. **RNN Model Architecture:** This figure presents the structure of the RNN model, which includes an LSTM layer designed to identify temporal relationships between input sequences and their outcomes, complemented by a dense output layer. The model processes sequential measurements from the IMU and vertical GRF across N time steps as its input, and generates a three-dimensional contact bias for each leg as its output.

To predict the contact bias, measurements from the GRF and the IMU are used. The LSTM requires sequential input; hence, the prediction at time t utilizes inputs from N time steps, ranging from $(t+1-N)$ to t. Each time step corresponds to a 2.5-millisecond interval, given the data collection frequency of 400 Hz. Specifically, GRF for leg i is calculated through a *Balance Controller* introduced in [8], and only the vertical force component $n_{i,k} \in \mathbb{R}, k \in [(t+1-N), t]$ is used for inputs. The input IMU measurements include orientation represented by Euler angles $[\boldsymbol{\theta}_k^\top]_\mathcal{I} \in \mathbb{R}^3$, angular velocity $[\boldsymbol{\omega}_k^\top]_\mathcal{I} \in \mathbb{R}^3$, and acceleration $[\mathbf{a}_k^\top]_\mathcal{I} \in \mathbb{R}^3$, each specified within the IMU coordinate frame. Therefore, the input matrix, denoted by $\mathbf{I}_{i,t} \in \mathbb{R}^{N\times 10}$ is defined as

$$\begin{bmatrix}[\mathbf{\Theta}_{t-N+1:t}]_\mathcal{I} & [\mathbf{W}_{t-N+1:t}]_\mathcal{I} & [\mathbf{A}_{t-N+1:t}]_\mathcal{I} & \mathbf{n}_{i,t-N+1:t}\end{bmatrix} \tag{19}$$

where submatrices $[\mathbf{\Theta}_{t-N+1:t}]_\mathcal{I}$, $[\mathbf{W}_{t-N+1:t}]_\mathcal{I}$, and $[\mathbf{A}_{t-N+1:t}]_\mathcal{I}$ are composed of the stacked vectors of Euler angles, angular velocities, and accelerations, respectively, over the window $(t-N+1)$ to t, vector $\mathbf{n}_{i,t-N+1:t}$ stacking the vertical GRF components across the same time window. To balance computational efficiency and model performance, $N = 30$ is used in this paper.

For the model training purpose, data collected from all four legs are joined together after being divided into swing and stance phases, i.e., $(\mathbf{I}_{i,t}^{sw}, \mathbf{b}_{i,t}^{sw})$ and $(\mathbf{I}_{i,t}^{st}, \mathbf{b}_{i,t}^{st})$. The swing or stance RNN model acts as a function f^{sw} or f^{st} that maps the IMU and GRF measurements to a contact location bias over a window of N time steps:

$$\mathrm{f}^{sw} : \mathbf{I}_{i,t}^{sw} \longrightarrow \mathbf{b}_{i,t}^{sw} \tag{20}$$

$$\mathrm{f}^{st} : \mathbf{I}_{i,t}^{st} \longrightarrow \mathbf{b}_{i,t}^{st} \tag{21}$$

C. Sampling Time Frame

The bias in foot positioning accumulated during the swing phase is reflected by the contact bias observed at the transition moment where the leg becomes the stance leg. Despite the planned intermittent contact of the foot with the ground, there are instances where the leg, expected to be in the stance phase, fails to establish firm contact, as well as the contrary scenario where the leg, scheduled for the swing phase, remains in contact [19]. To address this, our sampling strategy for the swing phase captures a larger dataset (20 time

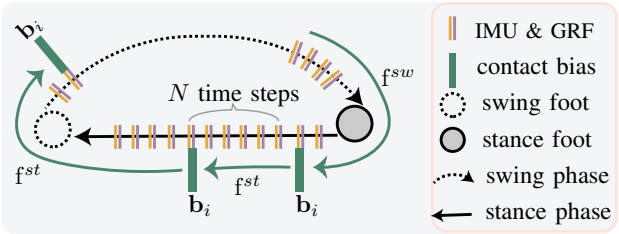

Fig. 3. **Sampling Time Frame:** The sampling strategy for the swing phase model includes data from 20 time steps before to 10 time steps after the anticipated moment of contact. The stance phase sampling encompasses data up to 10 time steps following the scheduled start of the swing phase. For a single leg, this methodology results in the generation of two samples from each stance phase and one sample from each swing phase.

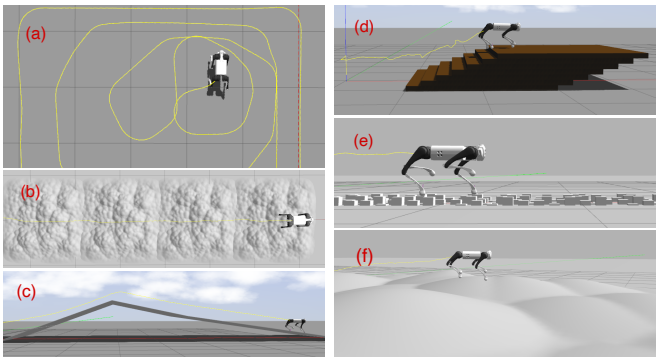

Fig. 4. **Simulation Environments:** This figure showcases six diverse simulation scenarios designed in the Gazebo simulator, each featuring unique terrain characteristics: (a) flat terrain, (b) uneven terrain, (c) slopes, (d) stairs, (e) blocks, and (f) hills.

steps) prior to the anticipated contact moment, to analyze the accumulative bias in foot positioning. Then, data collection continues for 10 time steps to monitor the initial adjustments or errors at the onset of the stance phase, as depicted in Fig. 3. Similarly, for the stance phase, sampling includes an additional 10 time steps beyond the anticipated start of the swing phase. Given the selection of $N = 30$ and the designed gait frequency, each stance phase yields two samples per leg, while the swing phase produces a single sample.

V. Simulation Results

This section presents the simulation outcomes of our investigation, beginning with the analysis of the trained RNN models in Section V-A. The models were trained on a computer equipped with an 11th Gen Intel Core i7 CPU, paired with an NVIDIA GeForce RTX 3070 GPU, and featuring 31 GB of total RAM. The proposed DLKF leg odometry is deployed on the Unitree Go1 robot. We compare the efficacy of our method against the standard KF leg odometry approach, with findings presented in Section V-B. Finally, we assess the integration of our method with a LiDAR odometer as a multi-sensor state estimation framework.

A. Neural Network Training Results

1) Platform and Data Collection Settings: The RNN models' training dataset was sourced from Gazebo simulations featuring a Unitree Go1 robot, which has four identical legs, each containing three motors with joint encoders and torque sensors. The robot is also equipped with a six-axis IMU and a 2D LiDAR, both installed on the robot body. Control over the robot was achieved through a randomly generated command velocity $[\mathbf{v}^c]_{\mathcal{B}} = (v_x^c, v_y^c, w^c)$, where v_x^c and v_y^c indicate the longitudinal and lateral velocities, w^c representing the yaw velocity in the robot's body frame. The command velocity was input from an XBox joystick.

To ensure a comprehensive dataset across diverse terrains, we designed three distinct simulation environments:

- *Flat Terrain*: Aimed at capturing data on the robot's directional movements and turns on level ground. (see Fig. 4(a)).
- *Uneven Terrain*: Features randomly generated hillocks to simulate conditions conducive to foot slippage, extending 8 meters in length with the robot traversing it three times for data collection. (see Fig. 4(b)).
- *Slope Terrain*: Challenges the robot with ascents and descents, to gather insights on contact bias under inclination changes with potential foot slippage. The terrain stretches 11 meters, with the robot completing three runs for data collection (see Fig. 4(c)).

2) RNN Model Training Results: A total of 9657 samples for the stance phase, and 4614 for the swing phase were gathered for this study. Initially, the dataset underwent a single shuffling process before being divided into training (80%), validation(10%) and testing (10%) subsets. Our methodology was developed using TensorFlow 2.13.0 [27], employing the ADAM optimizer [28] across 1000 training epochs. To reduce the risk of overfitting, the training dataset was shuffled prior to each epoch. The iteration with the minimal validation loss was chosen as the optimal model for further evaluation.

The training for both the stance and swing models converges at approximately 100 iterations. The Mean Absolute Error (MAE) observed in the test subset is 1.6 mm for the stance model and 2.4 mm for the swing model. The average computation time for one prediction request is 30.5 ms, enabling realtime operations.

B. Evaluation of Leg Odometry

To assess the efficacy of our methodology, this section compares its estimated leg odometry, including foot positioning and body positioning, against both the estimates derived from the basic KF and the ground truth.

1) Simulation Settings: To evaluate our leg odometry methodology, we conducted tests within six distinct simulation environments in the Gazebo simulator, as depicted in Fig. 4. Initially, the algorithms underwent testing in three environments—flat, uneven terrain, and slopes—which were also utilized for collecting neural network training data, as detailed in SectionV-A.1. To further examine the robustness and generalization capabilities of our approach, we introduced three additional, highly unstructured terrains:

- *Stairs*: Comprising 7 steps, each with a height of 7 cm and a width of 20 cm, this environment tests the algorithm's performance when the robot navigates

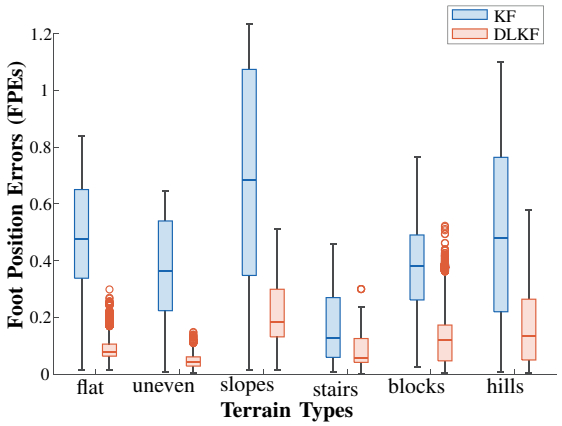

Fig. 5. **Comparative Analysis of Foot Position Errors:** Boxplot representation of FPEs for different terrain types, comparing the performance of the KF and DLKF.

TABLE I

MEAN FPES (M) COMPARISON BETWEEN KF AND DLKF ACROSS SIX SIMULATIONS, HIGHLIGHTING DISTANCES TRAVELED (M)

	flat	uneven	slopes	stairs	blocks	hills
KF	0.455	0.357	0.688	0.168	0.364	0.502
DLKF	0.086	0.045	0.214	0.080	0.131	0.183
Distance	18.65	8.49	11.36	3.75	8.02	12.83

TABLE II

COMPARATIVE EVALUATION OF RPE (M) AND MAE (M) BETWEEN KF AND DLKF ACROSS VARIOUS TERRAINS

	RPE		MAE (x/y/z)	
	KF	DLKF	KF	DLKF
flat	0.397	0.116	0.307/0.100/0.308	0.039/0.036/0.050
uneven	0.605	0.032	0.334/0.041/0.087	0.025/0.017/0.021
slopes	0.605	0.243	0.395/0.194/0.517	0.087/0.028/0.187
stairs	0.444	0.149	0.097/0.031/0.124	0.034/0.013/0.060
blocks	0.659	0.388	0.293/0.060/0.176	0.029/0.021/0.117
hills	0.498	0.256	0.059/0.148/0.478	0.053/0.095/0.143

through abrupt terrain changes. The stair configuration is shown in Fig. 4(d).

- *Blocks*: This scenario features a landscape of cubic blocks with varying heights, interspersed with gaps. No two adjacent cubes share the same height, creating a highly irregular terrain that stretches over 7 meters, as illustrated in Fig. 4(e).
- *Hills*: Characterized by a series of randomly generated hills of differing heights and gradients, this terrain extends across 10 meters, offering a challenging landscape for evaluation, as shown in Fig. 4(f).

Throughout the simulations, the Gazebo simulator provided ground truth data for both foot and body positions at a frequency of 1000 Hz. Meanwhile, the contact positions estimated by both the KF and the DLKF were outputted at a frequency of 400 Hz.

2) Foot Position Estimation: The swing and stance neural network models utilize IMU and GRF measurements to predict the contact biases for each leg. These predictions are subsequently employed to adjust the estimated foot contact positions via Eq. (18). The foot position error (FPE) is measured by calculating the three-dimensional Euclidean distance between the estimated foot positions by KF or DLKF and the ground truth positions. This calculation is performed at the frequency of the KF/DLKF output. Given the higher frequency of the ground truth data compared to that of both KF and DLKF estimates, the ground truth is resampled to the nearest corresponding timestamps of the estimation outputs for the FPE assessment. Figure 5 presents the box plots of FPE, aggregating results from all legs in each environment. The plots reveal that the FPE distributions for the DLKF are notably more compact, reflecting a narrower error range in all six environments and lower median values. This indicates a marked improvement in DLKF's performance over the traditional KF, as further evidenced by the reduced average FPEs detailed in Table I. The distance traveled in each environment is also listed in Table I.

3) Body Position Estimation: According to Eq. (13), given a relative foot position measurement, the estimation of the body position is related to the stance feet's positions. Consequently, enhancing the accuracy of contact position estimation is expected to directly improve the estimation of body position.

Figure 6 illustrates the body positions estimated by the basic KF and DLKF, compared with the actual ground truth. The simulation environments employed here are identical to those used in the contact position estimation tests discussed in the preceding section. From Fig. 6, it is evident that the DLKF achieves a more precise estimation of leg odometry than the basic KF across all six environments. This enhancement in accuracy is not limited to flat terrains; it also extends to challenging terrains where extensive foot slippage and falls were observed during the tests, albeit to a certain degree. It is important to highlight that despite the basic KF not being optimally calibrated for position estimation in the z-direction, as evidenced in the left graph of Fig. 6(a), the deep learning model has effectively addressed this limitation. This is reflected in the enhanced leg odometry results produced by the DLKF in the z-direction.

For each testing environment, the Relative Position Error (RPE) is calculated from the start to the end, reflecting the total position error over the entire distance traveled. Additionally, the Mean Absolute Errors (MAEs) with respect to the x, y and z directions are computed. The quantitative results in terms of RPE and MAE are summarized in Table II. The leg odometry estimated by the DLKF consistently reports lower RPE values, indicating its improved accuracy. Furthermore, the DLKF exhibits an average improvement of 64.93% in MAE terms over the six simulation scenarios.

4) Fusion with LiDAR Odometry: Leg odometry is crucial for multisensor state estimation in legged robots, particularly in scenarios where perception sensors such as cameras or LiDARs may be unreliable. This section demonstrates that the proposed leg odometry with learned contact bias, can be seamlessly integrated with measurements from other sensors. Specifically, the proposed leg odometry is loosely coupled with LiDAR-based odometry. The LiDAR odometry is obtained from a 2D LiDAR using the method from [29].

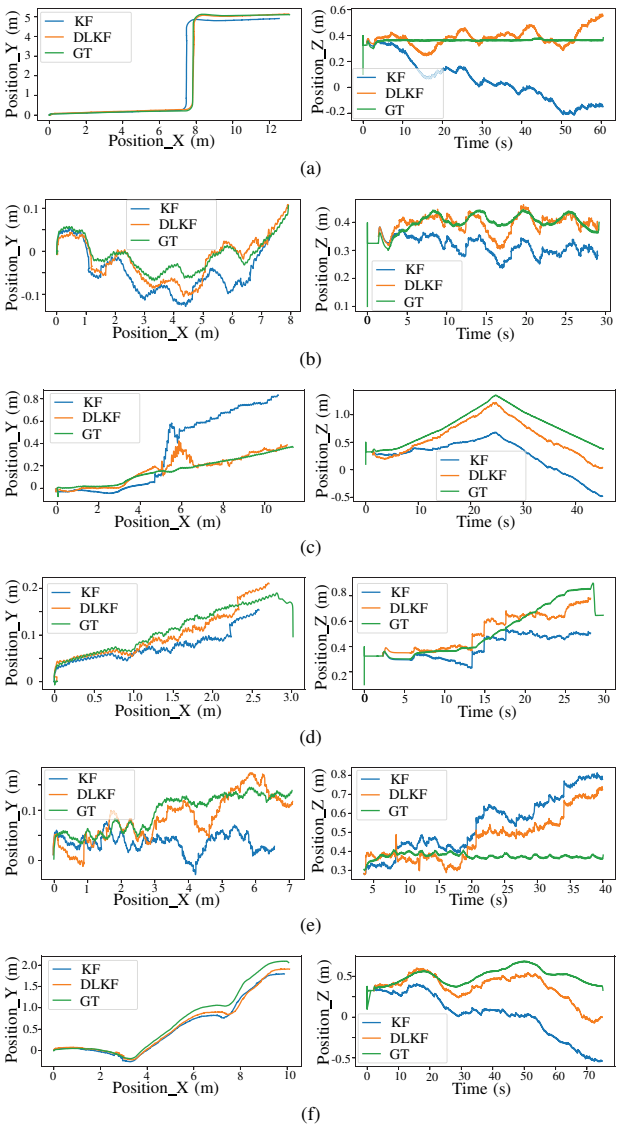

Fig. 6. **Body Position Estimation Comparison:** This figure compares the body position estimations from the KF and the DLKF with the ground truth (GT) in six simulation environments: (a) flat terrain, (b) uneven terrain, (c) slopes, (d) stairs, (e) blocks, and (f) hills. For each environment, the left graph within the subfigure illustrates the estimated (x, y) coordinates, whereas the right graph details the estimated z coordinates over time.

In environments rich in features, such as the one depicted in Fig.7(a) (left), LiDAR odometry alone can produce quite accurate position estimates. The integration of DLKF leg odometry further refines these estimates, resulting in a body pose that more closely aligns with the ground truth, as illustrated in Fig.7(a) (right). Conversely, in environments lacking distinct features, like the extended corridor shown in Fig.7(b) (left), LiDAR odometry fails entirely. Nevertheless, by combining LiDAR odometry with our DLKF leg odometry, we can still achieve high-quality state estimation, as evidenced in Fig.7(b) (right).

VI. Experiments

This section elaborates the experimental results obtained from deploying our algorithm on an actual Go1 robot, which

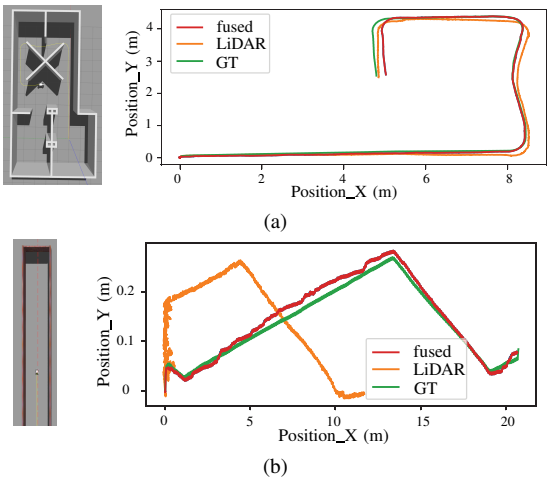

Fig. 7. **Sensor Fusion Results:** This figure presents a comparison of body position estimations obtained from a LiDAR-based odometry system and those fusing the LiDAR odometry with our DLKF leg odometry, against the ground truth (GT). The comparison is made as the robot navigates through (a) a feature-rich room and (b) a feature-sparse corridor.

has the same specifications as the simulated one.

For enhanced real-world applicability, the swing and stance RNN models were re-trained with actual experimental data. The robot's pose ground truth was obtained using a Vicon motion capture system, facilitating the computation of foot positions' ground truth through forward kinematics. The experiment was conducted exclusively on flat terrain. 11765 and 5887 data samples were collected for training the stance and swing models, respectively. Using the test dataset, the MAEs for the stance and swing models were observed to be 1.7 mm and 2.2 mm, respectively.

To evaluate our methodology's effectiveness, the robot was instructed to navigate our laboratory and a hallway. Figure 8 presents a comparative analysis, showing the robot's positions as estimated by both the conventional KF and the DLKF against their actual ground truth values. The results highlight that the DLKF's estimates align more closely with the ground truth. This improved accuracy is attributed to the integration of contact bias within the estimation framework.

VII. Conclusions

This paper presents a novel RNN-based leg odometry method for legged robots. An LSTM model is used to capture the temporal relationships between foot contact bias and the input from IMU and GRF measurements. The contact biases predicted by the RNN models provide a significant compensation to the basic KF estimation, leading to a more precise leg odometry estimation.

The efficacy of our method is closely tied to the composition of the training data. Enhanced precision in leg odometry is attainable as the training data encompasses a wider array of terrains, enabling the robot to better adapt to complex environments. Nevertheless, a notable limitation of this study is the exclusive use of flat terrain data for training in the experiments, due to the coverage constraints of our motion

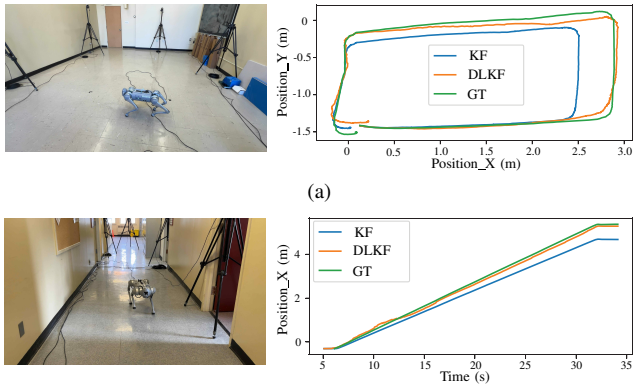

Fig. 8. **Experimental Outcomes:** Presented are the results from experiments featuring a Go1 robot navigating through: (a) an indoor laboratory measuring 7.5 by 4.8 meters; (b) a hallway that is 1.8 meters wide. The subfigures detail the respective environments and compare leg odometry estimates from basic KF and DLKF against the ground truth (GT).

capture system. This restriction undermines the model's application in varied real-world settings. Future efforts will focus on including a more diverse set of training data, reflective of varied physical environments. This expansion is expected to refine the model's robustness and ensure more reliable performance outside controlled conditions.

REFERENCES

[1] T.-Y. Lin, R. Zhang, J. Yu, and M. Ghaffari, "Legged robot state estimation using invariant Kalman filtering and learned contact events," *arXiv preprint arXiv:2106.15713*, 2021.

[2] M. Fourmy, T. Flayols, P.-A. Léziart, N. Mansard, and J. Solà, "Contact forces preintegration for estimation in legged robotics using factor graphs," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, Xi'an, China, 2021, pp. 1372–1378.

[3] S. Yang, H. Choset, and Z. Manchester, "Online kinematic calibration for legged robots," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 8178–8185, 2022.

[4] D. Wisth, M. Camurri, and M. Fallon, "VILENS: visual, inertial, lidar, and leg odometry for all-terrain legged robots," *IEEE Transactions on Robotics*, vol. 39, no. 1, pp. 309–326, 2022.

[5] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: continual prediction with LSTM," *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[6] M. Bloesch, M. Hutter, M. A. Hoepflinger, S. Leutenegger, C. Gehring, C. D. Remy, and R. Siegwart, "State estimation for legged robots: consistent fusion of leg kinematics and IMU," in *Proc. of Robotics: Science and Systems*, Jul 2012.

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

[8] 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)*, Madrid, Spain, 2018, pp. 2245–2252.

[9] T. Flayols, A. Del Prete, P. Wensing, A. Mifsud, M. Benallegue, and O. Stasse, "Experimental evaluation of simple estimators for humanoid robots," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, Birmingham, UK, 2017, pp. 889–895.

[10] 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*, Tokyo, Japan, 2013, pp. 6058–6064.

[11] 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.

[12] 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)*, Paris, France, 2020, pp. 392–398.

[13] 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)*, Brisbane, QLD, Australia, 2018, pp. 4422–4429.

[14] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry," *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, 2016.

[15] M. Bloesch, M. Hutter, C. Gehring, M. A. Hoepflinger, and R. Siegwart, "Kinematic batch calibration for legged robots," in *2013 IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, 2013, pp. 2542–2547.

[16] M. Bloesch, M. Burri, H. Sommer, R. Siegwart, and M. Hutter, "The two-state implicit filter recursive estimation for mobile robots," *IEEE Robotics and Automation Letters*, vol. 3, no. 1, pp. 573–580, 2017.

[17] 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.

[18] 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)*, Daejeon, Korea (South), 2016, pp. 3872–3878.

[19] G. Bledt, P. M. Wensing, S. Ingersoll, and S. Kim, "Contact model fusion for event-based locomotion in unstructured terrains," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, QLD, Australia, 2018, pp. 4399–4406.

[20] B. Wagstaff and J. Kelly, "LSTM-based zero-velocity detection for robust inertial navigation," in *2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Nantes, France, 2018, pp. 1–8.

[21] M. K. Al-Sharman, Y. Zweiri, M. A. K. Jaradat, R. Al-Husari, D. Gan, and L. D. Seneviratne, "Deep-learning-based neural network training for state estimation enhancement: Application to attitude estimation," *IEEE Transactions on Instrumentation and Measurement*, vol. 69, no. 1, pp. 24–34, 2019.

[22] M. Zhang, M. Zhang, Y. Chen, and M. Li, "IMU data processing for inertial aided navigation: a recurrent neural network based approach," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, Xi'an, China, 2021, pp. 3992–3998.

[23] C. Chen, X. Lu, A. Markham, and N. Trigoni, "IONet: learning to cure the curse of drift in inertial odometry," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

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

[25] 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.

[26] R. M. Murray, Z. Li, and S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation (1st Ed.)*. Boca Raton: CRC press, 1994.

[27] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: a system for large-scale machine learning," in *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, Savannah, GA, USA, 2016, pp. 265–283.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[29] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf, "A flexible and scalable SLAM system with full 3D motion estimation," in *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*, Kyoto, Japan, November 2011.

[30] Y. Gal and Z. Ghahramani, "Dropout as a Bayesian approximation: representing model uncertainty in deep learning," in *Proceedings of the* 33^{rd} *International Conference on Machine Learning*, vol. 48. PMLR, 2016, pp. 1050–1059.