

# Legged Robot State Estimation Using Invariant Neural-Augmented Kalman Filter with a Neural Compensator

Seokju Lee, Hyun-Bin Kim, and Kyung-Soo Kim

**Abstract**—This paper presents an algorithm to improve state estimation for legged robots. Among existing model-based state estimation methods for legged robots, the contact-aided invariant extended Kalman filter defines the state on a Lie group to preserve invariance, thereby significantly accelerating convergence. It achieves more accurate state estimation by leveraging contact information as measurements for the update step. However, when the model exhibits strong nonlinearity, the estimation accuracy decreases. Such nonlinearities can cause initial errors to accumulate and lead to large drifts over time. To address this issue, we propose compensating for errors by augmenting the Kalman filter with an artificial neural network serving as a nonlinear function approximator. Furthermore, we design this neural network to respect the Lie group structure to ensure invariance, resulting in our proposed Invariant Neural-Augmented Kalman Filter (InNKF). The proposed algorithm offers improved state estimation performance by combining the strengths of model-based and learning-based approaches. Supplementary Video: <https://youtu.be/k1ZVb6Xj8D8>

## I. INTRODUCTION

Significant research has focused on legged robots capable of traversing challenging terrains such as stairs, slopes, and slippery surfaces [1], [2], [3]. These robots are also used for terrain mapping, where SLAM algorithms rely heavily on accurate state estimation. Errors in state estimation can cause localization drift, leading to mapping failure [4]. This issue is especially critical when exteroceptive sensors (e.g., LiDAR, cameras) are unavailable, requiring state estimation using only proprioceptive sensors like IMUs and encoders.

State estimation methods have consisted of model-based and learning-based approaches. Among model-based methods, the state-of-the-art Invariant Extended Kalman Filter (InEKF) defines the state on a Lie group to improve convergence speed. It estimates states by incorporating contact foot kinematics as measurements [5]. However, InEKF assumes static contact, leading to significant errors when a slip occurs. To address this, methods like slip rejection [6] and Invariant Smoothers (IS) [7] have been proposed.

InEKF updates states using a first-order approximation for computing the matrix  $\mathbf{A}_t$ , which fails to account for nonlinearities in error dynamics, leading to estimation drift:

$$\frac{d}{dt} \xi_t = \mathbf{A}_t \xi_t + \text{Ad}_{\bar{\mathbf{X}}_t} \mathbf{w}_t. \quad (1)$$

$$\eta_t^r = \exp(\xi_t) \approx \mathbf{I}_d + \xi_t^\wedge. \quad (2)$$

\*This work was not supported by any organization. (Corresponding author: Kyung-Soo Kim)

The authors are with the Mechatronics, Systems and Control Lab (MSC Lab), Department of Mechanical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Yuseong-gu, Daejeon 34141, Republic of Korea (e-mail: {dljrwn0322, youfree22, kyungsokim}@kaist.ac.kr)

Meanwhile, learning-based approaches like KalmanNet [8], Neural Measurement Networks (NMN) [9], and methods that concurrent train state estimators and controllers [10] have been introduced. However, these methods still suffer from estimation errors, highlighting the need for further legged robot state estimation improvements.

In this paper, we first adopt the InEKF and propose the Invariant Neural-Augmented Kalman Filter (InNKF), which augments the InEKF with a neural network to compensate for errors in the InEKF-estimated state. Since errors in the states estimated by the InEKF arise because the error dynamics exhibit significant nonlinearity due to various factors (such as slip, model uncertainty, and sensor noise), a compensation step is added using a neural network as a nonlinear function approximator. A neural compensator capable of producing elements of the  $\text{SE}_2(3)$  Group is introduced to ensure that the neural network also preserves invariance. This neural compensator outputs the error  $\hat{\mathbf{E}}_t^{r'} = \bar{\mathbf{X}}_t^+ \bar{\mathbf{X}}_t^{-1} \in \text{SE}_2(3)$  as its result. Ground truth data is obtained from simulations, and the error between the InEKF estimates and the ground truth is calculated. This error is then used as a data label for training. At every time-step  $t$ , this value compensates the updated estimate  $\bar{\mathbf{X}}_t^+$ , resulting in  $\bar{\mathbf{X}}_t^{++} = \hat{\mathbf{E}}_t^{r'} \bar{\mathbf{X}}_t^+ \in \text{SE}_2(3)$ . Our main contributions are as follows:

- We propose a state estimator that combines the strengths of model-based and learning-based approaches, introducing an additional compensation step via a neural compensator to compensate errors caused by nonlinearities in the model-based estimated state.
- During the state estimation process, all states were defined on a Lie group to enhance the convergence speed of the model. To achieve this, the neural network was designed to output elements of the Lie group.
- Training data was collected from scenarios in the simulation where state estimation errors were likely to occur. The proposed method exhibited high state estimation performance on terrains where nonlinearities caused significant errors.

This paper is organized as follows. Section II introduces general approaches for contact estimators, assuming that no foot-mounted sensors are used, and provides an overview of legged robot state estimators. Section III explains the proposed method in detail, while Section IV presents the experimental setup and results. Finally, Section V concludes the paper and discusses future work.

## II. RELATED WORK

### A. Contact Estimator

Legged robots have contact between their feet and the ground during locomotion. This contact can introduce errors in state estimation, which can be addressed by utilizing contact information as measurements during the update step of the InEKF [5], as shown in (3).

$$\mathbf{Y}_t^\top = [h_p^\top(\tilde{\mathbf{q}}_t) \quad 0 \quad 1 \quad -1] \quad (3)$$

where  $h_p(\tilde{\mathbf{q}}_t)$  represents the forward kinematics of contact position measurements.

Estimating the contact state is a critical issue in legged robot state estimation. While the actual contact state can be measured using force/torque or pressure sensors, many legged robots do not utilize such sensors. Consequently, it is necessary to estimate the contact state using only IMUs and encoders. Various contact estimators for legged robots have been studied, including methods based on Convolutional Neural Networks (CNN) [11] and probabilistic estimation using Hidden Markov Models (HMM) [12].

Among these approaches, Marco Camurri [13] proposed a method where, for each contact point  $i$ , the contact force  $\mathbf{f}_i$  can be estimated based on robot dynamics and joint torques, as given in (4). The normal force component, perpendicular to the contact surface, is then computed via the inner product, as expressed in (5). Using this value, the contact probability  $P_i$  is determined through logistic regression, as shown in (6).

To use this information as measurements for InEKF, it is necessary to compute the covariance. The covariance is derived based on the variation in the estimated normal force from the previous time step, as formulated in (7). Finally, based on the computed contact probability, if the probability exceeds a predefined threshold  $\theta$ , the contact state is set to true, as defined in (8).

In this study, we use this contact estimator to design a legged robot state estimator.

$$\mathbf{f}_i = -\mathbf{J}_i^\top (\mathbf{J}_i \mathbf{J}_i^\top)^{-1} (\boldsymbol{\tau}_i - \mathbf{g}_i) \quad (4)$$

where  $\mathbf{J}_i$  is the contact Jacobian matrix at contact point  $i$ , defined at the joint positions,  $\boldsymbol{\tau}_i$  is the joint torque applied at contact point  $i$ , and  $\mathbf{g}_i$  is the force computed at contact point  $i$  using inverse dynamics.

$$f_{i,\text{normal}} = \mathbf{f}_i \cdot \mathbf{n}_i \quad (5)$$

where  $\mathbf{n}_i$  represents the normal vector of the contact surface at contact point  $i$ .

$$P_i = \frac{1}{1 + \exp(-\beta_1[i] f_{i,\text{normal}} - \beta_0[i])} \quad (6)$$

where  $\beta_0[i], \beta_1[i]$  are the regression coefficients.

$$\Sigma_i = k \cdot (f_{i,\text{normal}} - f_{i,\text{normal, prev}})^2 \quad (7)$$

$$S_i = \begin{cases} \text{true}, & \text{if } P_i \geq \theta \\ \text{false}, & \text{otherwise} \end{cases} \quad (8)$$

where  $S_i$  represents the contact state at contact point  $i$ .

### B. Legged Robot State Estimator

Legged robots can perform state estimation using either model-based or learning-based approaches. Model-based methods typically rely on Kalman filters, and due to the nonlinear dynamics of legged robot systems, the extended Kalman filter (EKF) is commonly employed. By leveraging leg kinematics and IMU data, EKF variants have been developed to address specific challenges. For instance, the quaternion-based EKF (QEKF) [14] resolves singularity issues and improves numerical stability using a quaternion-based mathematical model. To enhance the convergence speed of QEKF, the Invariant EKF (InEKF) [5] defines states on a Lie group to ensure invariance. This study adopts the contact-aided InEKF, which is widely used in legged robot state estimators.

For learning-based approaches, neural networks have been incorporated into the Kalman filter framework to compute covariances, as in KalmanNet [8]. Other methods include state estimators that learn contact events across various terrains for integration into InEKF [15], approaches that simultaneously train legged robot control policies and state estimators [10], and Pronto [16], which utilizes learned displacement measurements [17]. Additionally, the Neural Measurement Network (NMN) [9] employs neural networks to estimate measurement values for use in InEKF.

This study also employs neural networks; however, it proposes a state estimator by augmenting a model-based approach with neural networks.

## III. METHOD

In this study, the Neural Compensator (NC) is augmented to the InEKF, enabling an additional compensation of the error in the model-based updated estimated state to obtain the final estimated state  $\hat{\mathbf{X}}_t^{++}$ . During this process, the NC outputs an element of the  $SE_2(3)$  Group, ensuring that invariance is consistently maintained in the estimated states.

### A. $SE_2(3)$ Group Generation Network

The Neural Compensator is a neural network augmented to the InEKF, designed to consider invariance by outputting elements on the Lie group. To achieve this, the focus was placed on values defined in the Lie algebra, i.e., the tangent space, when designing the neural network. First, the state  $\mathbf{X}_t$  is defined in (9) as an element of  $\mathbb{R}^{5 \times 5}$ . If the generators of the  $SE_2(3)$  group's Lie algebra ( $\mathfrak{se}_2(3)$ ) are expressed as  $\mathbf{G}_i (i \in \{1, 2, \dots, 9\})$ , then the elements of  $\mathfrak{se}_2(3)$  can be represented as the linear combination of the generators, as shown in (10).

$$\mathbf{X}_t = \begin{bmatrix} \mathbf{R}_t & v_t & p_t \\ \mathbf{0}_{3,1} & 1 & 0 \\ \mathbf{0}_{3,1} & 0 & 1 \end{bmatrix} \in SE_2(3) \quad (9)$$

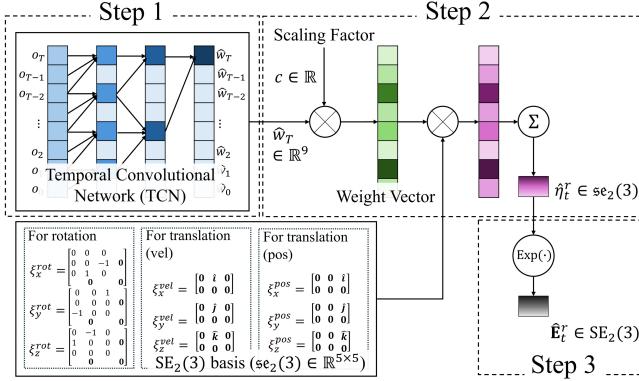


Fig. 1. The SE<sub>2</sub>(3) Group Generation Network (SEGGN) consists of three steps: (1) generating the weights for the elements of  $\mathfrak{se}_2(3)$ , (2) performing a linear combination of the TCN outputs with the elements of  $\mathfrak{se}_2(3)$ , and (3) applying exponential mapping to the results from step (2).

$$\xi = (\omega, v, p)^\top \in \mathbb{R}^9$$

$$\sum_{i=1}^9 \xi_i G_i \in \mathfrak{se}(3), \quad \text{where } \xi_i = \begin{cases} \omega_i & \text{if } i \leq 3, \\ v_{i-3} & \text{if } 4 \leq i \leq 6, \\ p_{i-6} & \text{if } i > 6. \end{cases} \quad (10)$$

Using the linear combination of the generators, the neural compensator, as shown in Figure 1, consists of three main steps. First, a temporal convolutional network (TCN) [18] takes as input a time-series sequence that includes linear acceleration, angular velocity, joint position, joint velocity, foot position, foot velocity, contact state, and the estimated state from the InEKF over the previous 49 time-steps as well as the current time step, and it produces 9 outputs. Second, these outputs are combined linearly with the generators of  $\mathfrak{se}_2(3)$  to obtain an element of  $\mathfrak{se}_2(3)$ , denoted as  $\hat{\eta}_t^r$ . Finally, by applying the exponential mapping described in (11), the result is mapped to an element of the SE<sub>2</sub>(3) group,  $\hat{\mathbf{E}}_t^r$ , which represents the estimated error. This value acts as a compensation term for the estimated state in the InEKF.

$$\begin{aligned} \exp(\cdot) : \hat{\eta}_t^r &\mapsto \hat{\mathbf{E}}_t^r \in \text{SE}_2(3) \\ \text{Exp}(\cdot) : \hat{\eta}_t^r &\mapsto \hat{\mathbf{E}}_t^r \in \text{SE}_2(3) \end{aligned} \quad (11)$$

The hidden layers of the TCN structure were configured as [128, 128, 128, 256, 256], with a kernel size of 2, a dropout rate of 0.5, and the ReLU activation function. The supervised learning process of the TCN was conducted as follows. The labeled dataset, which consists of ground truth error, was collected from simulation data. Specifically, ground truth values and estimated values from the InEKF were saved during the simulation. The target value for computing the loss was defined as  $\mathbf{E}_t^r = \bar{\mathbf{X}}_t^+ \mathbf{X}_t^{-1}$ , and training was performed using this target value. Based on the tangent space values obtained through the neural network, exponential mapping was applied to map them to the SE<sub>2</sub>(3) group. The loss function was calculated accordingly since the target value was also defined on the SE<sub>2</sub>(3) group.

The loss function was divided into a rotation part and a translation part. The rotation part corresponds to the top 3x3 matrix of the SE<sub>2</sub>(3) matrix, and the rotation loss was computed using the Frobenius norm and the geodesic distance. The translation part, which consists of velocity and position (i.e., the vectors in the 4th and 5th columns of the SE<sub>2</sub>(3) matrix), was used to compute the translation loss via Mean Squared Error. The total loss was calculated by weighting the rotation and translation losses, and training was performed. The Adam optimizer was used for training, with a learning rate 5e-4.

The InEKF operated at a frequency of 500 Hz, and inputs with the same frequency were provided for training. The TCN window time was set to 0.1 seconds, corresponding to a window size of 50, allowing the model to receive sequences of 50 time-steps as input. The window shift size was set to 1, resulting in a shift of 0.002 seconds for training.

### B. Invariant Neural-Augmented Kalman Filter

The Neural Compensator introduced in Section III-A is augmented into the InEKF [5] to form the Invariant Neural-Augmented Kalman Filter (InNKF). The structure of InNKF is shown in Figure 2, where the state estimation is first performed using a model-based approach. In this process, two types of states are defined. The state of the InEKF is expressed as  $\mathbf{Z}_t \in \text{SE}_{N+2}(3)$ , as shown in (12), while the state of the InNKF is expressed as  $\mathbf{X}_t \in \text{SE}_2(3)$ . Here,  $\mathbf{X}_t$  is a  $5 \times 5$  submatrix in the top-left corner of  $\mathbf{Z}_t$ .

$$\mathbf{Z}_t \triangleq \begin{bmatrix} \mathbf{X}_t & {}^w\mathbf{p}_{WC_1}(t) & {}^w\mathbf{p}_{WC_2}(t) & \cdots & {}^w\mathbf{p}_{WC_N}(t) \\ \mathbf{0}_{1,5} & 1 & 0 & \cdots & 0 \\ \mathbf{0}_{1,5} & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{1,5} & 0 & 0 & \cdots & 1 \end{bmatrix} \quad (12)$$

where  $\mathbf{X}_t$  represents the (9), and  ${}^w\mathbf{p}_{WC_i}(t)$  represents the each contact point for measurements of InEKF.

The IMU measurements used in the InEKF can be modeled using additive white Gaussian noise (AWGN), as expressed in (13).

$$\begin{aligned} \tilde{\mathbf{a}}_t &= \mathbf{a}_t + \mathbf{w}_t^g, \quad \mathbf{w}_t^g \sim \mathcal{N}(\mathbf{0}_{3 \times 1}, \Sigma^g \delta(t-t')) \\ \tilde{\mathbf{v}}_t &= \mathbf{v}_t + \mathbf{w}_t^a, \quad \mathbf{w}_t^a \sim \mathcal{N}(\mathbf{0}_{3 \times 1}, \Sigma^a \delta(t-t')) \end{aligned} \quad (13)$$

where  $\mathcal{N}$  is a Gaussian distribution and  $\delta(t-t')$  represents the Dirac delta function. Additionally, in the contact measurements, the velocity of the foot can also be modeled by considering potential slip and additive white Gaussian noise (AWGN). The measured foot velocity can thus be expressed as shown in (14).

$${}^w\tilde{\mathbf{v}}_C(t) = {}^w\mathbf{v}_C(t) + \mathbf{w}_t^v, \quad \mathbf{w}_t^v \sim \mathcal{N}(\mathbf{0}_{3 \times 1}, \Sigma^v \delta(t-t')) \quad (14)$$

The system dynamics derived from the IMU and contact

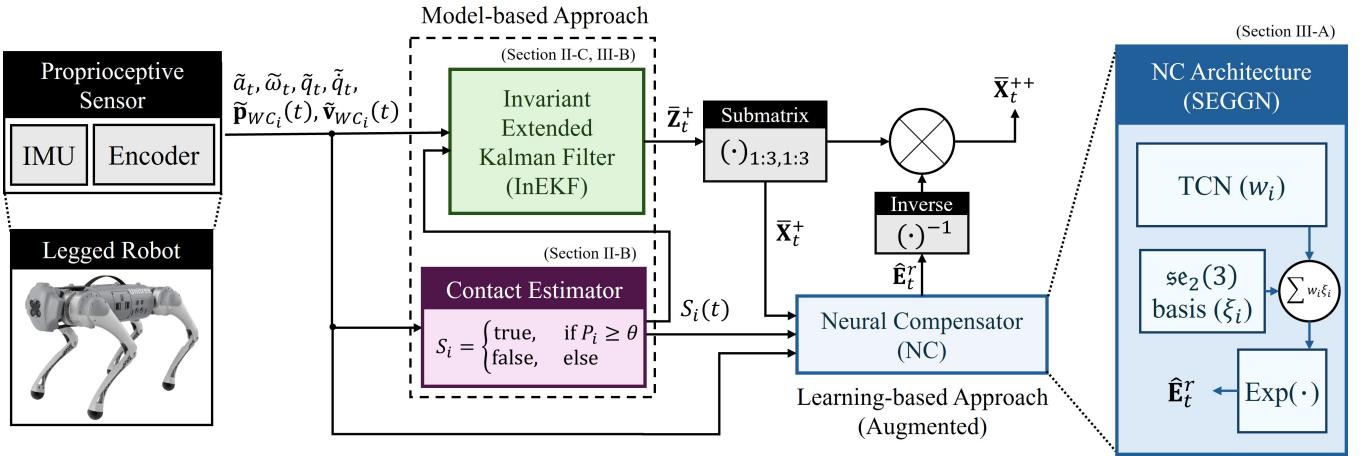


Fig. 2. The overall architecture of the Invariant Neural-Augmented Kalman Filter (InNKF). InNKF involves the predict and update step of InEKF and Neural Compensator (NC) that reduces error values obtained which is designed to  $se_2(3)$  Group Generation Network (SEGGN).

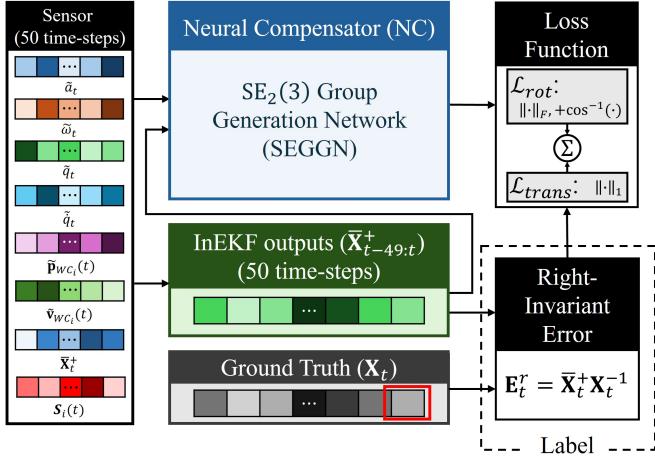


Fig. 3. Training process of the Neural Compensator (NC): The dataset is collected in 50 time-step sequences, where state estimates are obtained at each time step using the InEKF. The right-invariant error is computed and labeled only for the final time step. Based on this labeled error, the SEGGN is then trained using the output and a loss function.

measurements are expressed as shown in (15):

$$\begin{aligned} \frac{d}{dt} \mathbf{R}_t &= \mathbf{R}_t (\tilde{\omega}_t - \mathbf{w}_t^g) \\ \frac{d}{dt} \mathbf{v}_t &= \mathbf{R}_t (\tilde{a}_t - \mathbf{w}_t^a) + \mathbf{g} \\ \frac{d}{dt} \mathbf{p}_t &= \mathbf{v}_t \\ \frac{d}{dt} \mathbf{w}^w \mathbf{v}_C(t) &= \mathbf{R}_t h_R(\tilde{q}_t)(-\mathbf{w}_t^v) \end{aligned} \quad (15)$$

where  $h_R(\tilde{q}_t)$  denotes the orientation measurements of the contact frame in IMU frame by forward kinematics using encoder measurements,  $\tilde{q}_t$ . This equation can be represented in matrix form as shown in (16), which allows the compu-

tation of the predict step.

$$\begin{aligned} \frac{d}{dt} \bar{\mathbf{Z}}_t &= f_{u_t}(\bar{\mathbf{Z}}_t) \\ \frac{d}{dt} \mathbf{P}_t &= \mathbf{A}_t \mathbf{P}_t + \mathbf{P}_t \mathbf{A}_t^\top + \bar{\mathbf{Q}}_t \end{aligned} \quad (16)$$

where  $f_{u_t}(\cdot)$  represents the matrix form of (15),  $\mathbf{P}_t$  is the covariance matrix, and  $\mathbf{A}_t$  and  $\bar{\mathbf{Q}}_t$  are obtained from the linearization of the invariant error dynamics as shown in (17):

$$\begin{aligned} \mathbf{A}_t &= \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{g}^\times & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \\ \bar{\mathbf{Q}}_t &= \text{Ad}_{\bar{\mathbf{Z}}_t} \text{Cov}(\mathbf{w}_t) \text{Ad}_{\bar{\mathbf{Z}}_t}^\top. \end{aligned} \quad (17)$$

with  $\mathbf{w}_t = \text{vec}(\mathbf{w}_t^g, \mathbf{w}_t^a, \mathbf{0}_{3,1}, h_R(\tilde{q}_t)\mathbf{w}_t^v)$ .

The predicted state,  $\bar{\mathbf{Z}}_t$ , is updated using the measurements,  $\mathbf{Y}_t$ , from (3), along with the covariance, as shown in (18).

$$\begin{aligned} \bar{\mathbf{Z}}_t^+ &= \exp(\mathbf{K}_t \Pi_t \bar{\mathbf{Z}}_t \mathbf{Y}_t) \bar{\mathbf{Z}}_t \\ \mathbf{P}_t^+ &= (\mathbf{I} - \mathbf{K}_t \mathbf{H}_t) \mathbf{P}_t (\mathbf{I} - \mathbf{K}_t \mathbf{H}_t)^\top + \mathbf{K}_t \bar{\mathbf{N}}_t \mathbf{K}_t^\top \end{aligned} \quad (18)$$

where the  $\Pi_t$  defines the auxiliary selection matrix, which is the  $[\mathbf{I} \quad \mathbf{0}_{3,3}]$ , and the gain  $\mathbf{K}_t$  is obtained from (19):

$$\mathbf{K}_t = \mathbf{P}_t \mathbf{H}_t^\top (\mathbf{H}_t \mathbf{P}_t \mathbf{H}_t^\top + \bar{\mathbf{N}}_t)^{-1}, \quad (19)$$

$\mathbf{H}_t$  is defined to  $[\mathbf{0}_{3,3} \quad \mathbf{0}_{3,3} \quad -\mathbf{I} \quad \mathbf{I}]$ , and  $\bar{\mathbf{N}}_t$  is computed by  $\bar{\mathbf{R}}_t \mathbf{J}_p(\tilde{q}_t) \text{Cov}(\mathbf{w}_t^g) \mathbf{J}_p^\top(\tilde{q}_t) \bar{\mathbf{R}}_t^\top$ .

Now, the state  $\bar{\mathbf{Z}}_t^+$ , estimated through the InEKF, is used to extract a  $5 \times 5$  submatrix from the top-left corner, which represents only the robot base's state. The remaining values correspond to the contact positions of the robot's foot and are unnecessary for compensation; therefore, the submatrix is extracted. This extracted submatrix becomes  $\bar{\mathbf{X}}_t^+$ , which undergoes additional compensation to obtain the final estimated state.

Using the  $se_2(3)$  Group Generation Network (SEGGN) designed in Section III-A, a Neural Compensator (NC) is

added. The NC is trained on a dataset obtained from a simulation in which a single trajectory traverses four different terrains (stairs, slope, random uniform, discrete obstacle) for 100 seconds. The dataset included the ground truth state  $\mathbf{X}_t$ , sensor data such as linear acceleration ( $\ddot{\mathbf{a}}_t$ ), angular velocity ( $\tilde{\omega}_t$ ), joint position and velocity ( $\tilde{q}_t, \dot{\tilde{q}}_t$ ), foot position and velocity ( $\tilde{\mathbf{p}}_{WC_i}(t), \tilde{\mathbf{v}}_{WC_i}(t)$ ), contact state ( $S_t$ ), and the estimated state  $\bar{\mathbf{X}}_t^+$  from the InEKF. This dataset was used to train the SEGGN through the supervised learning process shown in Figure 3.

During the training process, the ground truth error, defined as  $\mathbf{E}_t^r = \bar{\mathbf{X}}_t^+ \bar{\mathbf{X}}_t^{-1} \in \text{SE}_2(3)$ , is used as the label. The loss is calculated between this value and the output of the SEGGN.

The training process employed two loss functions to separately handle rotation and translation. For the rotation part, the loss function in (22) was formulated using the geodesic distance and Frobenius norm. The Frobenius norm captures element-wise differences between the ground truth and network output, while the geodesic distance ensures proper evaluation of rotational discrepancies on  $\text{SO}(3)$ . This combination makes (22) effective for training. The Frobenius norm is defined in (20).

$$\mathcal{L}_{\text{fro}} = \frac{1}{2} \|\mathbf{R}_1 - \mathbf{R}_2\|_F^2, \quad (20)$$

where  $\|\mathbf{R}_1 - \mathbf{R}_2\|_F = \sqrt{\sum_{i,j} (\mathbf{R}_1[i, j] - \mathbf{R}_2[i, j])^2}$  denotes the element-wise difference between the matrices. This metric ensures computational efficiency while providing a measure of proximity in Euclidean space.

On the other hand, the geodesic distance represents the minimal angular displacement required to align  $\mathbf{R}_1$  with  $\mathbf{R}_2$  on the rotation group  $\text{SO}(3)$ . It is computed as (21):

$$\mathcal{L}_{\text{geo}} = \arccos \left( \frac{\text{trace}(\mathbf{R}_1^T \mathbf{R}_2) - 1}{2} \right), \quad (21)$$

where  $\text{trace}(\cdot)$  is the sum of the diagonal elements of the matrix. This measure captures the intrinsic rotational difference and is particularly meaningful in applications sensitive to angular deviations.

$$\mathcal{L}_{\text{rot}}(\mathbf{R}_1, \mathbf{R}_2) = \alpha \mathcal{L}_{\text{fro}} + \beta \mathcal{L}_{\text{geo}} \quad (22)$$

where  $\alpha$  and  $\beta$  are weighting factors that balance the contributions of the Frobenius norm and the geodesic distance. This formulation ensures both computational simplicity and rotational accuracy.

The loss function for the translation part is formulated using the L1 loss, as shown in (23).

$$\mathcal{L}_{\text{trans}} = \|\mathbf{v}_1 - \mathbf{v}_2\|_1 + \|\mathbf{p}_1 - \mathbf{p}_2\|_1, \quad (23)$$

The final loss function is constructed using (22) and (23), as expressed in (24), and is used for training.

$$\mathcal{L} = c_1 \mathcal{L}_{\text{rot}} + c_2 \mathcal{L}_{\text{trans}} \quad (24)$$

where  $c_1$  and  $c_2$  denote the coefficients of the loss functions.

The SEGGN trained using (24) is employed as the NC and augmented into the InEKF. The label used for training

the SEGGN is given by  $\mathbf{E}_t^r = \bar{\mathbf{X}}_t^+ \bar{\mathbf{X}}_t^{-1}$ , and the output of the SEGGN can therefore be expressed as  $\hat{\mathbf{E}}_t^r$ . Since the SEGGN's output represents the error that must be corrected in the state estimated by the InEKF, it can be formulated as shown in (25).

$$\bar{\mathbf{X}}_t^{++} = \hat{\mathbf{E}}_t^{r^{-1}} \bar{\mathbf{X}}_t^+ \quad (25)$$

where  $\bar{\mathbf{X}}_t^{++}$  represents the compensated state derived from the estimated state by the InEKF. This compensated state serves as the final state estimation of the InEKF, with the InEKF and NC operating in parallel. The InEKF continuously computes estimates based on proprioceptive sensor data, while the NC uses the estimates from the InEKF and proprioceptive sensor values as inputs to predict the estimated error. The NC operates with a delay of one time step compared to the InEKF and corrects the InEKF's estimated states using its estimated error. The compensated state is not reused in the InEKF but is solely used as the final state estimate.

#### IV. EXPERIMENTS

This study aims to enhance the state estimation performance of legged robots by evaluating rotation, position, and velocity estimation. Simulations provide ground truth data for direct performance assessment, but testing on a single simulator may lead to overfitting. To ensure generalization, the estimator was further evaluated in a different dynamic simulator. To introduce significant nonlinearities, trajectories were designed on terrains with sudden rotational changes. Additionally, real-world performance was tested in an uneven terrain environment.

##### A. Performance Analysis of $\text{SE}_2(3)$ Group Generation

The neural network used in this study does not perform operations directly on the Lie group but operates in the Lie algebra, followed by exponential mapping. Since the Lie group is defined on a smooth manifold, performing group operations requires matrix multiplication. In contrast, the Lie algebra, defined in the tangent space, only requires matrix addition for operations. Consequently, obtaining the output in the Lie algebra and applying exponential mapping significantly improves training efficiency. Table I and Figure 4 show the computation time according to batch size. While matrix addition has a time complexity of  $O(n^2)$ , matrix multiplication has a time complexity of  $O(n^3)$ . This highlights the advantage of obtaining outputs in the tangent space, where only matrix addition is required, to enhance training efficiency. A difference of approximately 0.1 times in computation time was observed, confirming significantly faster performance.

Furthermore, due to the structure of the SEGGN, which utilizes the bases of  $\mathfrak{se}_2(3)$  followed by an exponential mapping process, it inevitably produces outputs within  $\text{SE}_2(3)$ . The corresponding results are presented in Figure 5. The first row of Figure 5 illustrates the transformation of the original coordinates based on the output values obtained from SEGGN. Visually, it can be observed that the transformed

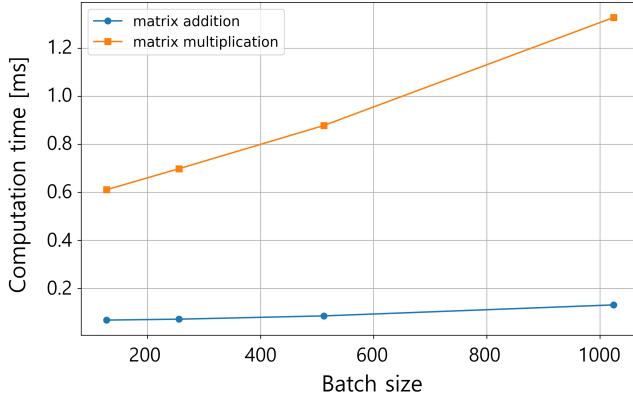


Fig. 4. Comparison of addition and multiplication operation time depending on the batch size. This graph represents the efficiency of computation in lie algebra.

Batch Size	Computation Time [ms]		Ratio (Add/Mul)
	Addition	Multiplication	
128	0.0654	0.6140	0.1066
256	0.0703	0.6978	0.1008
512	0.0826	0.8612	0.0960
1024	0.1033	1.2170	0.0849

TABLE I

COMPUTATION TIME OF ADDITION AND MULTIPLICATION DEPENDING ON BATCH SIZE AND THEIR RATIO.

coordinates maintain orthonormality, ensuring the structural integrity of the transformation. A numerical analysis of these results is presented in the histograms below. The Frobenius norm error distribution shows that nearly all samples have errors close to zero, indicating minimal deviation from the expected transformation properties. Additionally, the determinant values are consistently close to one, confirming that the rotational component of the output matrix adheres well to the properties of the SO(3) group (26).

$$\text{SO}(3) = \{\mathbf{R} \in \mathbb{R}^{3 \times 3} | \mathbf{R}\mathbf{R}^\top = \mathbf{I}, \det(\mathbf{R}) = 1\} \quad (26)$$

These results demonstrate that SEGGN successfully generates outputs that satisfy the rotational and translational properties of the SE<sub>2</sub>(3) group.

### B. Implementation Details

To reduce the sim-to-real gap, sensor noise and IMU bias were introduced to better reflect real-world conditions. These noise factors were applied during both locomotion controller training and dataset collection. The sensor noise was assumed to follow a Gaussian distribution. Additionally, to address uncertainties in the measurement model, a covariance matrix was configured. Since InEKF measurements primarily depend on kinematic information, the covariance matrix was constructed based on the standard deviation of the noise applied to joint position and velocity.

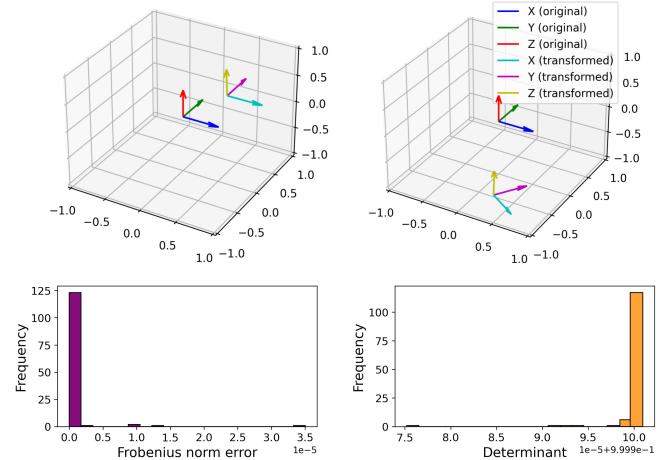


Fig. 5. The visual and numerical results of the trained SEGGN. The first row illustrates the transformed coordinates based on the SEGGN output. The second row presents numerical evaluations, including the Frobenius norm error, calculated as  $\|\mathbf{R}^\top \mathbf{R} - \mathbf{I}\|_F$ , and the determinant of  $\mathbf{R}$ , both shown as histograms.

### C. Experimental Results

Two baselines were compared to analyze the experimental results. The first was the model-based (MB) estimator, represented by the Invariant Extended Kalman Filter (InEKF). The second was the learning-based (LB) estimator, represented by a Neural Network (NN).

The training and test datasets were collected in the Isaac Gym simulator [19], using terrains shown in Figure 7. The quadruped robot, Unitree Go1, was the model used, controlled by a blind locomotion controller trained via reinforcement learning [20]. To evaluate whether the state estimator overfits the training conditions, additional experiments were conducted in Raisim [21]. The results from Isaac Gym are presented in Figure 6 and Table II, while the results from Raisim are shown in Figure 8 and Figure 9. Furthermore, to assess real-world applicability, the proposed method was evaluated in environments depicted in Figure 10, with the corresponding results summarized in Table III.

In Figure 6, the proposed method, InNKF, demonstrates a significant reduction in y-axis position error compared to InEKF. In particular, for the discrete obstacle terrain, the z-axis position error of InEKF is 0.3472m at 140s, whereas InNKF maintains an error of 0.1420m, confirming its improved performance. Table II further evaluates the performance using Absolute Trajectory Error (ATE), including NN-based state estimation as a baseline. The results indicate that InNKF consistently reduces rotation ATE across all terrains. Specifically, velocity ATE is reduced by 40.1% in the slope terrain, while position ATE is reduced by 87.0% in the discrete obstacle terrain. These results demonstrate that NN-based state estimation alone does not provide reliable performance, whereas the proposed InNKF achieves superior accuracy in most cases.

To validate its performance in a different dynamic simulator, position error was evaluated in Raisim, with addi-

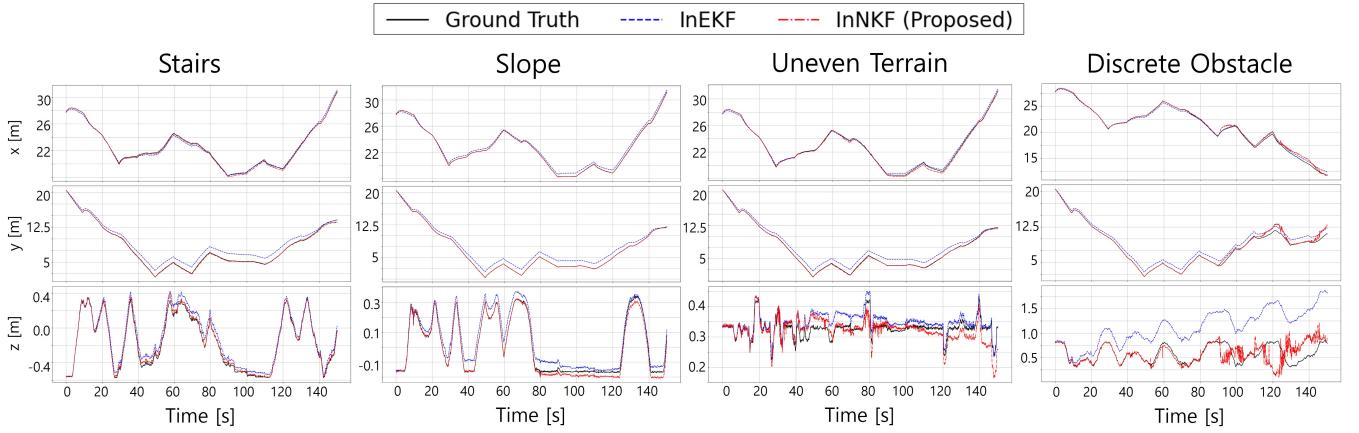


Fig. 6. Position estimation results across four terrains (stairs, slope, uneven terrain, discrete obstacle) over a time interval of  $0 \sim 140$  s. The solid black line represents the ground truth, the blue dashed line represents InEKF, and the red dashed line represents the InNKF (proposed method).

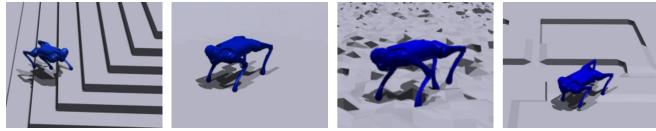


Fig. 7. The four terrains (from left: stairs, slope, uneven terrain, and discrete obstacle) in the Isaac Gym are used for collecting training and test datasets.

Terrain	Eval. metric	Algorithm		
		InEKF (MB-only)	NN (LB-only)	InNKF (Proposed)
Stairs	ATE <sub>R</sub> [rad]	0.1247	0.4695	<b>0.0986</b>
	ATE <sub>v</sub> [m/s]	<b>0.1441</b>	0.2712	0.2056
	ATE <sub>p</sub> [m]	<b>0.6224</b>	1.9160	0.8697
Slope	ATE <sub>R</sub> [rad]	0.0454	0.1212	<b>0.0231</b>
	ATE <sub>v</sub> [m/s]	0.0930	0.0648	<b>0.0373</b>
	ATE <sub>p</sub> [m]	1.0439	0.2363	<b>0.1455</b>
Uneven Terrain	ATE <sub>R</sub> [rad]	0.0388	0.3114	<b>0.0341</b>
	ATE <sub>v</sub> [m/s]	<b>0.0930</b>	0.1744	0.1186
	ATE <sub>p</sub> [m]	1.0401	1.4215	<b>0.4582</b>
Discrete Obstacle	ATE <sub>R</sub> [rad]	0.0500	0.1110	<b>0.0215</b>
	ATE <sub>v</sub> [m/s]	0.0958	0.0631	<b>0.0462</b>
	ATE <sub>p</sub> [m]	1.0734	0.2036	<b>0.1389</b>

TABLE II

COMPARISON OF ABSOLUTE TRAJECTORY ERROR (ATE) IN STATE ESTIMATION AMONG THE TWO BASELINES AND INNKF (PROPOSED METHOD) ACROSS DIFFERENT TERRAINS.

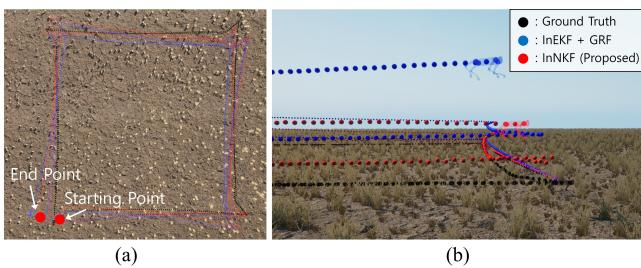


Fig. 8. State estimation results in Raisim for two laps around a  $20 \text{ m} \times 20 \text{ m}$  square path on field terrain. The black line represents the ground truth, the blue line represents InEKF, and the red line represents InNKF (proposed). (a): Bird's-eye view, (b): Side view.

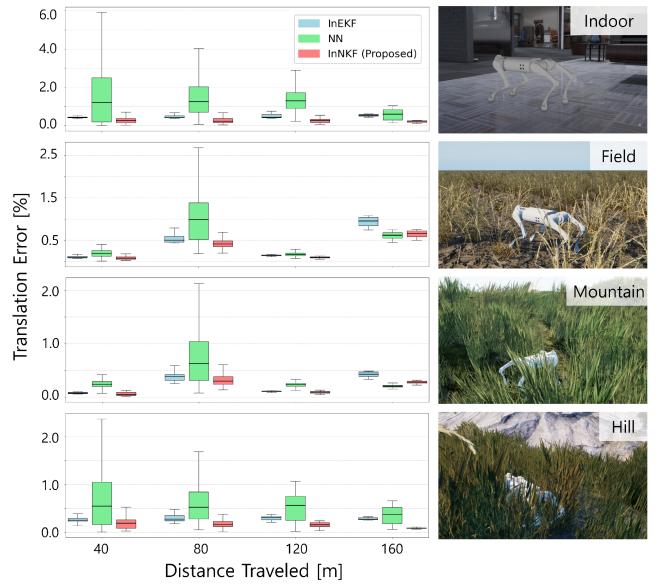


Fig. 9. Relative Errors (RE) in Raisim across different environments (indoor, field, mountain, hill). In each box, blue represents InEKF, green represents NN, and red represents InNKF (proposed).

tional data collected for fine-tuning. As shown in Figure 8, when the robot completed two laps along a  $20\text{m} \times 20\text{m}$  square trajectory, the x and y-axis errors between the starting and ending points remained minimal. Additionally, InNKF demonstrated significantly improved accuracy along the z-axis compared to InEKF, staying closer to the ground truth. Figure 9 compares the relative error (RE) across four environments—indoor, field, mountain, and hill—over a total walking distance of 160m. NN-based estimation exhibited high variance, reaching a maximum of 4.6298 in the indoor environment, whereas InNKF maintained a significantly lower maximum variance of 0.0466, highlighting the unreliability of NN-based estimation. As the distance traveled increased, InNKF consistently maintained a lower RE than InEKF, achieving up to 68.32% performance improvement.

To verify the applicability of the proposed method in



Fig. 10. Real-world experimental setup with gravel, wooden planks, and a cart to create unstructured terrain.

Eval. metric	Algorithm		
	InEKF (MB-only)	NN (LB-only)	InNKF (Proposed)
ATE <sub>R</sub> [rad]	0.2564	0.1453	<b>0.0330</b>
ATE <sub>p</sub> [m]	0.2042	0.1291	<b>0.0726</b>
RE <sub>R</sub> [rad/m]	0.0210 (0.0021)	0.0220 (0.0114)	<b>0.0209 (0.0021)</b>
RE <sub>p</sub> [%]	<b>1.0019</b> <b>(0.1727)</b>	3.3852 (2.0124)	1.1690 (0.4459)

TABLE III

ATE AND 10 SECONDS RE (MEAN AND STANDARD DEVIATION) OF STATE ESTIMATION IN THE REAL WORLD.

real-world environments, experiments were conducted in the setting shown in Figure 10. Since a motion capture system was unavailable, pseudo ground truth was obtained using LIO-SAM [22] with VectorNav VN-100 IMU and Ouster OS1-64-U LiDAR. Based on this, the state estimation performance was evaluated by comparing ATE and Relative Error (RE), and the results are summarized in Table III. The proposed InNKF demonstrated improvements of up to 77% in ATE, confirming its feasibility for real-world applications.

## V. CONCLUSIONS

This study proposed the Invariant Neural-Augmented Kalman Filter (InNKF), a novel state estimator that integrates a neural network-based compensation step into a model-based framework. Using a neural network as a nonlinear function approximator, InNKF corrects linearization errors at each time step, enhancing state estimation performance. However, as it relies on dataset-based learning, retraining is needed for different robot models, and the lack of a motion capture system in real-world experiments limited precise evaluation. SEGGN operates at 660 Hz, suggesting that its integration with InEKF (up to 2000 Hz) ensures real-time capability. Future work will refine the Neural Compensator structure, address key nonlinear factors, and extend its applicability to various legged robots.

## REFERENCES

- [1] R. Grandia, F. Jenelten, S. Yang, F. Farshidian, and M. Hutter, “Perceptive locomotion through nonlinear model-predictive control,” *IEEE Transactions on Robotics*, vol. 39, no. 5, pp. 3402–3421, 2023.
- [2] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, “Learning quadrupedal locomotion over challenging terrain,” *Science robotics*, vol. 5, no. 47, p. eabc5986, 2020.
- [3] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, “Learning robust perceptive locomotion for quadrupedal robots in the wild,” *Science robotics*, vol. 7, no. 62, p. eabk2822, 2022.
- [4] D. Wisth, M. Camurri, and M. Fallon, “Robust legged robot state estimation using factor graph optimization,” *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4507–4514, 2019.
- [5] 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.
- [6] 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.
- [7] Z. Yoon, J.-H. Kim, and H.-W. Park, “Invariant smoother for legged robot state estimation with dynamic contact event information,” *IEEE Transactions on Robotics*, 2023.
- [8] G. Revach, N. Shlezinger, X. Ni, A. L. Escoriza, R. J. Van Sloun, and Y. C. Eldar, “Kalmannet: Neural network aided kalman filtering for partially known dynamics,” *IEEE Transactions on Signal Processing*, vol. 70, pp. 1532–1547, 2022.
- [9] D. Youm, H. Oh, S. Choi, H. Kim, and J. Hwangbo, “Legged robot state estimation with invariant extended kalman filter using neural measurement network,” *arXiv preprint arXiv:2402.00366*, 2024.
- [10] G. Ji, J. Mun, H. Kim, and J. Hwangbo, “Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion,” *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4630–4637, 2022.
- [11] T.-Y. Lin, R. Zhang, J. Yu, and M. Ghaffari, “Legged robot state estimation using invariant kalman filtering and learned contact events,” in *Proceedings of the 5th Conference on Robot Learning*, pp. 1057–1066, PMLR, 2022.
- [12] 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)*, pp. 3872–3878, IEEE, 2016.
- [13] 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.
- [14] 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,” 2013.
- [15] 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.
- [16] M. Camurri, M. Ramezani, S. Nobili, and M. Fallon, “Pronto: A multi-sensor state estimator for legged robots in real-world scenarios,” *Frontiers in Robotics and AI*, vol. 7, p. 68, 2020.
- [17] R. Buchanan, M. Camurri, F. Dellaert, and M. Fallon, “Learning inertial odometry for dynamic legged robot state estimation,” in *Conference on robot learning*, pp. 1575–1584, PMLR, 2022.
- [18] S. Bai, J. Z. Kolter, and V. Koltun, “An empirical evaluation of generic convolutional and recurrent networks for sequence modeling,” *arXiv preprint arXiv:1803.01271*, 2018.
- [19] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, *et al.*, “Isaac gym: High performance gpu-based physics simulation for robot learning,” *arXiv preprint arXiv:2108.10470*, 2021.
- [20] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, “Learning to walk in minutes using massively parallel deep reinforcement learning,” in *Conference on Robot Learning*, pp. 91–100, PMLR, 2022.
- [21] J. Hwangbo, J. Lee, and M. Hutter, “Per-contact iteration method for solving contact dynamics,” *IEEE Robotics and Automation Letters*, vol. 3, no. 2, pp. 895–902, 2018.
- [22] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, “Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping,” in *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 5135–5142, IEEE, 2020.