IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

# Enhancing Robustness of Locomotion Policy for Quadrupedal Robot with Deep Disturbance Observer

Fikih Muhamad, Anak Agung Krisna Ananda Kusuma, Jae-Han Park, and Jung-Su Kim*

***Abstract*—This letter proposes a control framework to enhance the robustness of a locomotion policy against uncertainties by integrating it with a deep disturbance observer (DOB) network and a deep state estimator network. The deep DOB approximates the inverse model of a quadrupedal robot. The locomotion policy is trained to produce optimal actions, with the deep DOB estimating the overall uncertainties of the robot, and the deep state estimator estimates the body's linear velocities. All networks are trained under nominal conditions in IsaacGym. Subsequently, all the trained networks are transferred to Gazebo and a real robot with ROS2 are used to validate their robustness under uncertain conditions without additional tuning. Furthermore, validation results show that the proposed control framework performs best in velocity tracking compared to the baseline method in terms of lowest estimation errors. This emphasizes the effectiveness of the proposed control framework in improving robustness of the locomotion policy. Videos on IsaacGym and Gazebo simulation, and real robot experiment are available at Project page: bit.ly/3CF3OTQ.**

***Index Terms*—Legged robot, robust locomotion policy, deep learning, disturbance observer.**

## I. Introduction

DEEP reinforcement learning (RL) has significantly advanced in developing locomotion policies for legged robots. Famous locomotion policies employing neural networks have demonstrated high performances in various scenarios. A locomotion policy using deep RL was designed for high-speed legged locomotion with the Anymal robot [1], [2]. DeepGait was trained to produce an optimal policy considering a height map as input [3]. Furthermore, Rudin et al. introduced an approach utilizing parallelized online training in GPU-based simulations, enabling the simultaneous training of multiple robots under nominal conditions [4]. Gangapurwala et al. [5] propose a reinforcement learning-based approach that uses sensor feedback and velocity commands for footstep planning on uneven terrain, while Bellegarda et al. [6] develop a framework for fast and robust bounding by directly controlling task-space actions under noise, and Wang et al. [7] introduce a teacher-student RL architecture for efficient locomotion over uneven terrain.

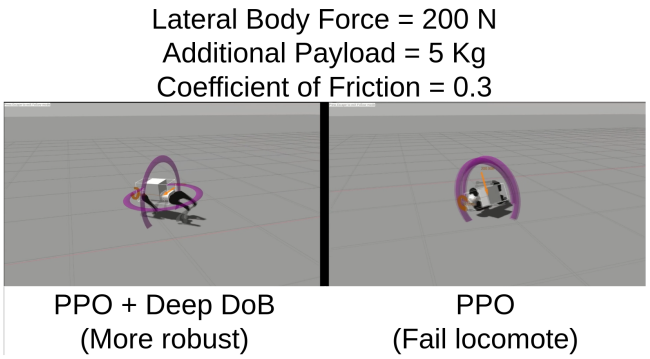



Fig. 1. The Unitree Go1 Edu equipped with PPO and deep DOB can locomote effectively under uncertain conditions in Gazebo with ROS2. It can handle a 200 N lateral body force, an additional 5 kg payload, and a 0.3 coefficient of friction. This performance surpasses PPO without deep DOB which fails during locomotion when faced with these uncertainties.

These locomotion policies can perform optimally under nominal conditions without any uncertainties. However, in real-world scenarios, robots often encounter uncertain conditions such as external disturbances and sensor noise. External disturbances may include navigating over slippery surfaces, carrying additional payloads, or experiencing lateral body forces. Moreover, noise can also arise from inaccurate state measurements due to sensor limitation and external disturbance. The robot's lack of familiarity with these uncertain conditions during training can significantly impact its performance and lead to failed locomotion.

Additionally, several states cannot be directly measured by onboard sensors in real-world applications. These states are known as privileged states. Although privileged states can be easily evaluated during simulation, it is challenging to acquire their information in real-world scenarios. Existing locomotion policies often require these privileged states to generate optimal actions. As a result, it is essential to devise a state estimator for those privileged states and achieve optimal locomotion performance.

### A. Related Work

Domain randomization is a common strategy to bridge the simulation-to-reality gap by introducing noise and disturbances during training [8]. Rudin et al. applied this method

Manuscript received: February 11, 2025; Revised: May 13, 2025; Accepted: July 12, 2025. This paper was recommended for publication by Editor Abderrahmane Kheddar upon evaluation of the Associate Editor and Reviewers' comments.

*Corresponding author: J.-S. Kim (e-mail: jungsu@seoultech.ac.kr; Tel.: +82-2-970-6547).

Fikih Muhamad, Anak Agung Krisna Ananda Kusuma, and Jung-Su Kim* are with the Control and Intelligent Robotics Lab, Department of Electrical and Information Engineering, Seoul National University of Science and Technology, Seoul 01811, Korea.

Jae-Han Park is with the AI Robot R&D Department, Korea Institute of Industrial Technology (KITECH), Ansan 15588, Korea.

to improve the smoothness of locomotion policies [4], while others have randomized payloads to enhance robustness to external forces [6]. Despite its benefits, domain randomization alone is insufficient for effectively identifying or adapting to real-world disturbances. Moreover, relying on pre-defined noise distributions can hinder generalization to unseen scenarios. In legged robots, the high-dimensional state space makes selecting effective randomization parameters difficult, increasing vulnerability to the curse of dimensionality.

Model-based disturbance observers offer robust control without extensive parameter tuning [9]. They have shown effectiveness in handling mismatched uncertainties, such as in miniature helicopters using sliding mode control [10], and in fixed-wing UAVs dealing with wind disturbances [11]. Recent work also combines reinforcement learning with model-based signum of the error (RISE) control for robust quadcopter performance under real-world uncertainties [12]. However, these approaches rely on accurate dynamic models, which are challenging to construct for legged robots due to their complex, nonlinear dynamics and high-dimensional state space.

Another approach uses robust tube Model Predictive Control (MPC) for setpoint tracking in manipulators, followed by training a neural network to imitate the MPC controller, reducing computation time compared to online optimization [13]. This idea has been extended by using deep reinforcement learning to mimic trajectories from finite-horizon robust MPC as expert demonstrations [14], [15]. The resulting policies exhibit strong robustness to external disturbances, and expert-generated motions improve sampling efficiency in training. However, this method requires multiple steps: formulating the MPC, collecting data, and training the deep RL model, making the process complex and resource-intensive.

Recent studies have applied distributional reinforcement learning (RL) to improve reward optimization under large disturbances, enhancing the robustness of locomotion policies in real-world conditions [16], [17]. While effective, identifying critical disturbances in locomotion remains difficult. To address this, Shi et al. proposed an RL-based method to uncover problematic observation, command, and perturbation spaces in locomotion policies [18]. These insights allow for policy refinement by adjusting domain randomization and retraining. However, this approach involves multiple stages which makes the process time-consuming and computationally intensive.

Existing model-based state estimators often fail in challenging conditions like slippery terrain [19], [20]. Neural network-based estimators have shown greater robustness [21]. For example, Lee et al. used privileged information to train locomotion policies in the real world [22], and Miki et al. proposed a belief encoder-decoder for estimating exteroceptive states under sensor uncertainty [23]. Ji et al. further introduced concurrent training of policies and estimators for robust locomotion on slippery surfaces [24]. However, these methods typically rely on randomized initial state noise during training, which limits generalization to unseen scenarios with greater uncertainties.

### B. Contributions

This study proposes a control framework to address robust locomotion policy against uncertain operating conditions. The deep DOB is designed to enhance the robustness of locomotion policies against external disturbances under uncertain conditions. Additionally, a deep state estimator is devised to accurately estimate privileged states, such as body linear velocity in uncertain conditions. The deep DOB and the deep state estimator use a combination of Long Short-Term Memory (LSTM) networks and Multi-Layer Perceptrons (MLP). In contrast to existing approaches, this control framework is specifically designed to improve the robustness of locomotion policies under uncertain conditions that are never experienced during training.

The most important contribution of this paper is to propose a way to deal with levels of uncertainty that domain randomization cannot handle [18], using a key idea from disturbance observer theory: the use of inverse models. Therefore, the proposed method approximates the inverse model using a neural network.

Furthermore, accurately identifying an inverse nominal model for a disturbance observer [9] is highly challenging due to the complex, nonlinear, and high-dimensional nature of quadruped robot dynamics, as described in [25], [26]. Especially, when multiple components are lumped into a single disturbance term, modeling errors can significantly degrade the performance of observers and controllers. Therefore, constructing an inverse dynamics model in this disturbance observer is particularly challenging and often impractical.

To address these issues, we adopt a learning-based approach to replace selected model-based components within the locomotion controller, disturbance observer, and state estimator. This model-free, deep learning-based method captures the nonlinear and high-dimensional characteristics of quadruped dynamics more effectively while reducing reliance on precise modeling. As a result, the proposed locomotion policy can concentrate more directly on rejecting noise and external disturbances which can not be dealt with domain randomization only.

The contribution of the letter is in order:

- The proposed framework integrates the locomotion policy network with a deep DOB network and a deep state estimator network. The training process is conducted by training all networks in IsaacGym under nominal conditions.
- The optimal networks are transferred to Gazebo and the real Unitree Go1 Edu robot with ROS2 to validate their robustness under uncertain conditions. Additionally, a performance comparison of the proposed framework with baseline methods is presented.
- The entire framework utilizes a model-free approach to eliminate the need for formulating a mathematical model of a legged robot. Furthermore, comparisons between different types of deep DOB and deep state estimator architectures such as Recurrent Neural Networks (RNN) are conducted.

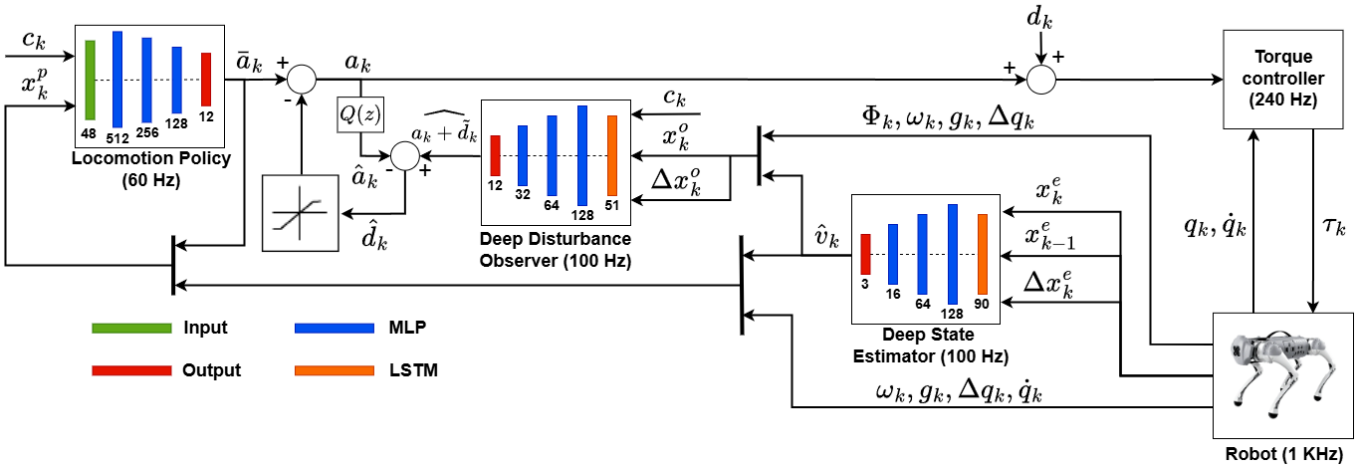



Fig. 2. Illustration of the proposed framework to enhance the robustness of the locomotion policy. The locomotion policy generates action $\bar{a}_k$ based on the given command $c_k$ and policy state $x_k^p$. Furthermore, the proposed deep disturbance observer (DOB) based on the inverse model estimates $\widehat{a_k + d_k}$. Subsequentially, the estimated disturbance $\hat{d}_k$ is calculated by $\hat{d}_k \approx a_k + \tilde{d}_k - \hat{a}_k$. Additionally, the deep state estimator estimates the body linear velocity $\hat{v}_k$ based on the estimator state $x_k^e$, previous estimator state $x_{k-1}^e$, and transition of estimator state $\Delta x_k^e$. The deep DOB output (100 Hz) estimates action space disturbances instead of torque space, as the torque controller runs faster (240 Hz) than the locomotion policy (60 Hz). Although torque space disturbances are effective in simulation, prior real world experiment shows that deep DOB struggles with high rates on real robots with limited resources.

- The proposed framework is validated using both Gazebo simulation and experiments with Unitree Go1 Edu.

## II. Methodology

This section provides a detailed explanation of the architecture and calculations involved in the proposed framework. An overview of the proposed framework is presented in Fig. 2.

### A. Locomotion Policy

Defining the parameters: $v_k \in R^3$ represents the body linear velocity, $\omega_k \in R^3$ the body angular velocity, $g_k \in R^3$ the gravity projection, and $\Phi_k \in R^3$ the Euler angles of the body of the robot. Additionally, the difference between the joint positions $q_k \in R^{12}$ and the arbitrary default joint positions $q_d \in R^{12}$ is denoted as $\Delta q_k \in R^{12}$, so $\Delta q_k = q_k - q_d$. Lastly, $\dot{q}_k \in R^{12}$ denotes the joint velocities.

The locomotion policy aims to produce an action $\bar{a}_k \in R^{12}$ to track a velocity command $c_k \in R^3$ with 60 Hz sampling rate. The action represents the relative position targets offset by $q_d$ of the robot while the command consists of body velocities target $c_k = \begin{bmatrix} v_{x,k}^c & v_{y,k}^c & \omega_{z,k}^c \end{bmatrix}$. To track the command, the policy state $x_k^p$ is constructed as:

$$x_k^p = \begin{bmatrix} v_k & \omega_k & g_k & \Delta q_k & \dot{q}_k & \bar{a}_{k-1} \end{bmatrix} \in \mathbb{R}^{45}. \quad (1)$$

This study employs the stochastic policy for the locomotion policy. It can be expressed using MLP $\pi$ with weight $\theta$. The locomotion policy predicts the mean $\mu_k \in R^{12}$ and variance $\sigma_k \in R^{12}$ of the action based on the policy state $x_k^p$ and command $c_k$. Then, the action $\bar{a}_k$ is generated from the distribution of the locomotion policy output. The calculation of the locomotion policy is represented as follows:

$$\begin{bmatrix} \mu_k & \sigma_k \end{bmatrix} = \pi(c_k, x_k^p | \theta), \quad (2)$$

$$\bar{a}_k \sim \mathcal{N}(\mu_k, \sigma_k). \quad (3)$$

The deep RL approach is used to obtain the optimal locomotion policy. It trains the locomotion policy to maximize the expected rewards over a horizon $\mathrm{E}[\sum_{k=0}^{\infty} \gamma^k r_{k+1}]$, where $\gamma$ is a discount factor and $r_k$ denotes the reward functions for achieving the control objective of the locomotion policy. The problem of training this locomotion policy using RL can be formulated as a Markov Decision Process (MDP). The MDP formulation consists of $(x_k^p, \bar{a}_k, r_k, x_{k+1}^p, \bar{a}_{k+1})$.

To maximize the expected reward and obtain the optimal weight $\theta^*$, Proximal Policy Optimization (PPO) with Kullback-Leibler (KL) divergence is used [27].

The action $\bar{a}_k$ from the locomotion policy then is used to calculate the torque $\tau_k \in R^{12}$ for each joint of the robot in the torque controller. It is expressed as follows:

$$\tau_k = K_p \times (\bar{a}_k + q_d - q_k) - K_d \times \dot{q}_k \in \mathcal{R}^{12}, \quad (4)$$

where $K_p$ and $K_d$ are tuning parameters for the torque controller.

### B. Deep Disturbance Observer

The nominal model of the legged robot is expressed as:

$$x_{k+1}^p = \bar{\mathcal{F}}(\bar{a}_k, x_k^p | \bar{A}), \quad (5)$$

where $\bar{a}_k$ denotes the action for the nominal model, $\bar{A}$ is a known nominal parameters and $\bar{\mathcal{F}}(\cdot)$ is the nominal mathematical model or simulation model of the legged robot such as IsaacGym. Since there are no uncertainties, it is assumed that the nominal control $\bar{a}_k$ is designed to guarantee acceptable performance of the nominal closed-loop system (5) [9]. On the other hand, the real legged robot can be expressed as

$$x_{k+1}^p = \mathcal{F}(a_k + d_k, x_k^p | A), \quad (6)$$

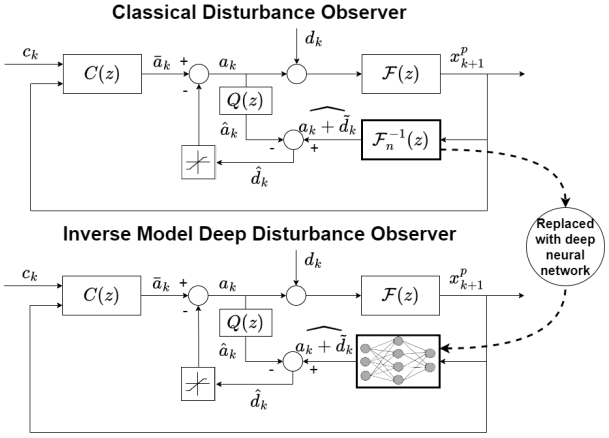



Fig. 3. Deep DOB: approximation of the inverse model using deep network.

where $\mathcal{F}(\cdot)$ denotes the dynamics of the real legged robot, $a_k$ is the action for the real legged robot, $A$ denotes uncertain parameters, and $d_k$ is unknown external disturbance.

Following the control design setup of the disturbance observer theory, and considering the two expressions of the nominal and real legged models, it is assumed that there exists a lumped disturbance $\tilde{d}_k$ such that

$$x^p_{k+1} = \bar{\mathcal{F}}(a_k + \tilde{d}_k, x^p_k | \bar{A}). \tag{7}$$

Note that the lumped disturbance $\tilde{d}_k$ is assumed to include entire uncertainties such as modeling error, parameter error, and external disturbance. Hence, it is possible to think that there exists the inverse dynamics of (7) as follows.

$$a_k + \tilde{d}_k = \bar{\mathcal{F}}^{-1}(x^p_{k+1}, x^p_k | \bar{A}^{-1}), \tag{8}$$

where $a_k + \tilde{d}_k$ implies the injected input to the real robot, i.e., the sum of the computed control input and lumped matched disturbance. Then, the lumped disturbance estimate $\hat{d}_k$ can be approximated as $\hat{d}_k \approx a_k + \tilde{d}_k - \hat{a}_k$ where $\hat{a}_k$ is a filtered $a_k$ using a stable filter with Exponential Moving Average (EMA) [28]. It can be expressed as $\hat{a}_k = EMA(a_k|\alpha)$ where $\alpha$ is a smoothing factor. See Fig. 3. To avoid large error in the disturbance estimate during the transient period, a saturation function is used as

$$\hat{d}_k = clip(\hat{d}_k, d_{max}, d_{min}). \tag{9}$$

It is well known that the DOB using the inverse model can estimate the disturbance very conservatively during the transient period, i.e., very large estimate error and it can put harmful effect on the system [9]. This is known as a peaking phenomenon in nonlinear observer theory [29]. To overcome this, a saturation function is used at the output of the DOB. Then, the DOB-based locomotion control is written as

$$a_k = \bar{a}_k - \hat{d}_k. \tag{10}$$

As a result, the closed-loop system becomes

$$\begin{aligned} x^p_{k+1} &= \bar{\mathcal{F}}(\bar{a}_k - \hat{d}_k + \tilde{d}_k, x^p_k | \bar{A}) \\ &\approx \bar{\mathcal{F}}(\bar{a}_k, x^p_k | \bar{A}). \end{aligned} \tag{11}$$

Therefore, the resulting closed-loop system consisting of the uncertain legged robot with the DOB-based control becomes the same as the closed-loop (5) of the nominal model with the nominal control $\bar{a}_k$. This property is called nominal performance recovery in [9]. This is how the DOB-based locomotion control robustly stabilizes the uncertain legged robot.

However, due to the complexity, non-linearity, and high-dimensional state of the dynamics of a legged robot as in [25], [26], it is very difficult to find the inverse nominal model $\bar{\mathcal{F}}^{-1}$. To address this issue, this paper proposes to design a deep neural network to approximate the inverse nominal model $\bar{\mathcal{F}}^{-1}$. It is called as deep DOB in this letter. The schematic is illustrated in Fig. 3.

The observer state $x^o_k$ is introduced as a subset of the policy state $x^o_k \subset x^p_k$. Due to this, sample efficiency is maintained since both deep DOB and locomotion policy are trained using the same dataset. Additionally, to ensure that the deep DOB is less sensitive to the sampling time $T$, the transition of observer states $\Delta x^o_k$ is also introduced. $x^o_k$ and $\Delta x^o_k$ are defined as

$$x^o_k = \begin{bmatrix} \Phi_k & v_k & \omega_k & g_k & \Delta q_k \end{bmatrix} \in \mathbb{R}^{24}, \tag{12}$$

$$\Delta x^o_k = \frac{x^o_{k+1} - x^o_k}{T} \in \mathbb{R}^{24}. \tag{13}$$

The proposed deep DOB denoted by $\hat{\mathcal{F}}^{-1}_\lambda$is constructed using a combination of LSTM and MLP with $\lambda$ representing the weight of the network.

An MLP can model complex static nonlinearities but lacks temporal memory, making it insufficient when disturbances depend on past states, inputs, or actions. LSTMs, on the other hand, capture temporal dependencies but may struggle with high-dimensional nonlinear mappings. By combining LSTM and MLP, we exploit both temporal awareness and nonlinear approximation: the LSTM extracts sequential features, and the MLP maps them to accurate disturbance estimates. This hybrid improves deep DOB performance compared to using either model alone.

After that, with the command $c_k$, the sequence of observer states $\begin{bmatrix} c_r & x^o_r & \Delta x^o_r \end{bmatrix}$ for the deep DOB input is constructed over a horizon $s$.

$$\begin{bmatrix} c_r \\ x^o_r \\ \Delta x^o_r \end{bmatrix} = \begin{bmatrix} c_k & c_{k-1} & \cdots & c_{k-s} \\ x^o_k & x^o_{k-1} & \cdots & x^o_{k-s} \\ \Delta x^o_k & \Delta x^o_{k-1} & \cdots & \Delta x^o_{k-s} \end{bmatrix} \in \mathbb{R}^{51 \times s}. \tag{14}$$

The deep DOB uses the sequence of observer states as input to predict $a_k + \tilde{d}_k$.

$$\widehat{a_k + \tilde{d}_k} = \hat{\mathcal{F}}^{-1}(c_r, x^o_r, \Delta x^o_r | \lambda), \tag{15}$$

where $\widehat{a_k + \tilde{d}_k}$ denotes the estimate of $a_k + \tilde{d}_k$ generated by the deep DOB.

A supervised learning is employed to train the deep DOB under nominal conditions. In this letter, $x^o_i$ and resulting $\bar{a}_i$ are collected for training data. The objective of the training is to minimize the observation loss $\mathcal{L}_{obs}$ until obtaining the optimal weight $\lambda^*$.

$$\lambda^* = \underset{\lambda}{arg\,min}\, \mathcal{L}_{obs} = \underset{\lambda}{arg\,min}\, \frac{1}{N} \sum_{i=1}^{N} \left(\bar{a}_i - \hat{\bar{a}}_i\right)^2, \tag{16}$$

where $\hat{\bar{a}}_i$ is the output of the deep network approximating the inverse model when $x_i^o$ is injected to the deep network. After obtaining the optimal deep DOB, the equation (8) can be replaced by equation (15).

This study presents the output from the deep DOB (100 Hz) from the disturbances in the action and not the torque. This approach is motivated by the fact that the torque controller operates at a higher sampling rate (240 Hz) than the locomotion policy (60 Hz). It is difficult for a neural network to handle it effectively at such a high sampling rate. Prior experiments show that the deep DOB performs better in torque control than in locomotion policy on simulations. However, when applied to a real robot with limited resources, the deep DOB struggles to handle such a very high sampling rate.

### C. Deep State Estimator

The objective of the deep state estimator is to estimate the privilege state such as the body linear velocity with 100 Hz sampling rate. It utilizes estimator state $x_k^e$ which is the subset of policy state $x_k^e \subset x_k^p$. Similar to the deep DOB, sample efficiency is maintained. This state can be expressed as:

$$x_k^e = \begin{bmatrix} \omega_k & g_k & \Delta q_k & \dot{q}_k \end{bmatrix} \in \mathbb{R}^{30}, \tag{17}$$

$$\Delta x_k^e = \frac{x_k^e - x_{k-1}^e}{T}. \tag{18}$$

The deep state estimator is constructed by integrating LSTM and MLP denoted by $\Gamma$ with a weight $\delta$. The motivation for using combination of the LSTM and MLP in the deep state estimator is similar to that in the deep DOB. The input to the deep state estimator $\Gamma$ is a sequence of the estimator states as follows $\begin{bmatrix} x_r^e & x_{r-1}^e & \Delta x_r^e \end{bmatrix}$ where $x_r^e$ is the estimator state, $x_{r-1}^e$ the previous estimator state, and $\Delta x_r^e$ the transition of estimator state.

$$\begin{bmatrix} x_r^e \\ x_{r-1}^e \\ \Delta x_r^e \end{bmatrix} = \begin{bmatrix} x_k^e & x_{k-1}^e & \cdots & x_{k-s}^e \\ x_{k-1}^e & x_{k-2}^e & \cdots & x_{k-s-1}^e \\ \Delta x_k^e & \Delta x_{k-1}^e & \cdots & \Delta x_{k-s}^e \end{bmatrix} \in \mathbb{R}^{90 \times s}. \tag{19}$$

A supervised learning approach is used to train the deep state estimator $\Gamma$ under nominal conditions. The objective of the training is to minimize the estimation loss $\mathcal{L}_{est}$ until obtaining the optimal weight $\delta^*$.

$$\delta^* = \underset{\delta}{arg\,min}\, \mathcal{L}_{est} = \underset{\delta}{arg\,min}\, \frac{1}{N} \sum_{i=1}^{N} \left(v_i - \hat{v}_i\right)^2, \tag{20}$$

where the body linear velocity $v_k$ is measured from the simulation.

The trained deep state estimator $\Gamma$ can estimate the body linear velocity when the sequence of estimator states is given by

$$\hat{v}_k = \Gamma(x_r^e, x_{r-1}^e, \Delta x_r^e | \delta). \tag{21}$$

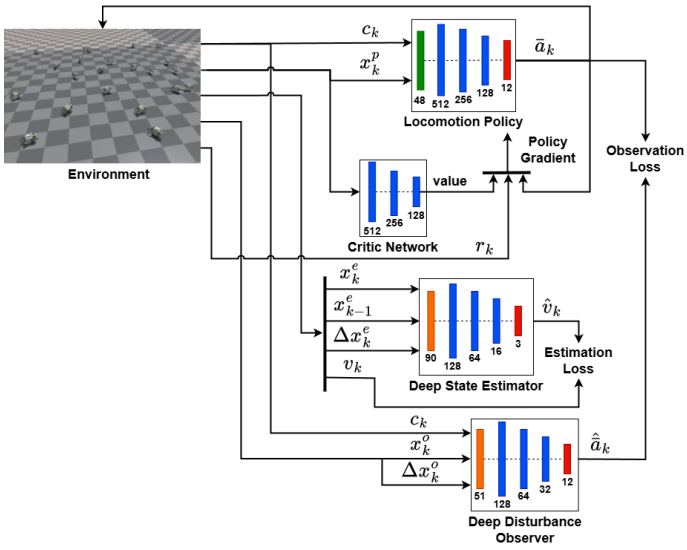



Fig. 4. An illustration of the training process for the proposed framework is shown. The locomotion policy is trained using the policy gradient method PPO to maximize the reward. Meanwhile, the deep DOB is trained to minimize the observation loss, and the deep state estimator is trained to minimize the estimation loss.

TABLE I
PARAMETERS OF NOMINAL AND UNCERTAIN CONDITIONS.

| | nominal | uncertain |
|---|---|---|
| additional body mass | [-1.0, 1.0] Kg | 5.0 Kg |
| coefficient of friction | [0.5, 1.25] | [0.2, 0.4] |
| lateral body force | 75 N | 200 N |
| joint position noise | ± 0.1 rad | ± 0.15 rad |
| joint velocity noise | ± 1.5 rad/s | ± 2.25 rad/s |
| linear velocity noise | ± 0.1 m/s | ± 0.15 m/s |
| angular velocity noise | ± 0.2 rad/s | ± 0.3 rad/s |
| gravity noise | ± 0.05 m/$s^2$ | ± 0.075 m/$s^2$ |

## III. RESULTS

This section presents the training and validation process of the proposed control framework. All networks are trained on flat terrain under nominal conditions using multiple robots in the IsaacGym simulation environment with PyTorch [30] and two NVIDIA®Titan V GPUs. The training time for the locomotion policy is 1.5 hours, while the deep disturbance observer and state estimator are trained together over 4 hours. An overview of the training process is shown in Fig. 4. After training, the networks are deployed on Gazebo and a real Unitree Go1 Edu without further tuning. Running on a single Jetson Xavier NX with ROS2, the system shows robustness under uncertainty. Table I details the conditions, and various architectures for the deep DOB and state estimator are evaluated.

### A. Training Proposed Framework

The training process of the locomotion policy using a RL approach in nominal conditions requires reward functions $r_k$ as control objectives. In this paper, the locomotion policy, reward function, and curriculum for training the locomotion policy are inspired by [4]. The reward functions used for training the locomotion policy are listed in Table II. Additionally,

TABLE II
REWARD FUNCTION.

| Reward term | Reward equation | Weight |
|---|---|---|
| $r_{\text{action rate}}$ | $\|a_{k-1} - a_k\|^2$ | -0.01 |
| $r_{\text{ang vel xy}}$ | $\|\omega_{xy}\|^2$ | -0.05 |
| $r_{\text{base height}}$ | $\|p_z - p_z^*\|^2$ | -10.0 |
| $r_{\text{collision}}$ | $n_{collisions}$ | -1.0 |
| $r_{\text{joint acc}}$ | $\|\ddot{q}\|^2$ | -2.5e-7 |
| $r_{\text{joint pos limits}}$ | $min((q-q_l),0) + max((q-q_u),0)$ | -10.0 |
| $r_{\text{lin vel z}}$ | $v_z^2$ | -2.0 |
| $r_{\text{torques}}$ | $\|\tau\|^2$ | -0.0002 |
| $r_{\text{feet air time}}$ | $\sum_{f=0}^{4}(t_{air,f} - 0.5)$ | 1.0 |
| $r_{\text{tracking ang vel}}$ | $\exp(\frac{-\|c_z^\omega - \omega_z\|^2}{0.25})$ | 0.5 |
| $r_{\text{tracking lin vel}}$ | $\exp(\frac{-\|c_{xy}^v - v_{xy}\|^2}{0.25})$ | 1.0 |

TABLE III
HYPERPARAMETERS OF TRAINING.

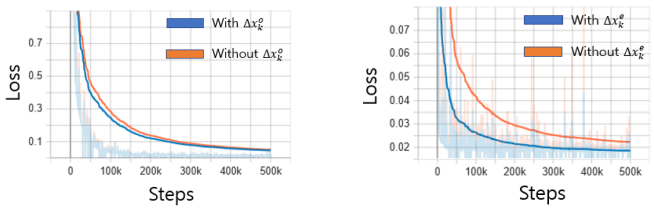
| | policy | observer | estimator |
|---|---|---|---|
| number of environments | 1000 | | |
| step | 500k | | |
| T | 0.0167 | 0.01 | |
| learning rate | $1e^{-3}$ | $1e^{-5}$ | $1e^{-4}$ |
| s | - | 5 | 5 |
| kp | 25.0 | - | - |
| kd | 0.5 | - | - |
| $d_{max}$ | - | 0.1 | - |
| $d_{min}$ | - | -0.1 | - |
| $\alpha$ | - | 0.1 | - |

the robot receives random commands within the range of $[-1\,m/s,\,1\,m/s]$ for $v_x^c$ and $v_y^c$, and within the range of $[-1.0\,rad/s,\,1.0\,rad/s]$ for $\omega_k^c$ with the purpose of sampling various data for the training process.

In the training of locomotion policy under nominal conditions, domain randomization is used to introduce small disturbances and noise to the robot [4]. This makes the locomotion policy a PPO with domain randomization. Existing results and prior experiments have demonstrated that training without domain randomization can lead to stiff and shaky movements [24]. Additionally, the arbitrary default joint position for the Unitree Go1 Edu robot is configured as:

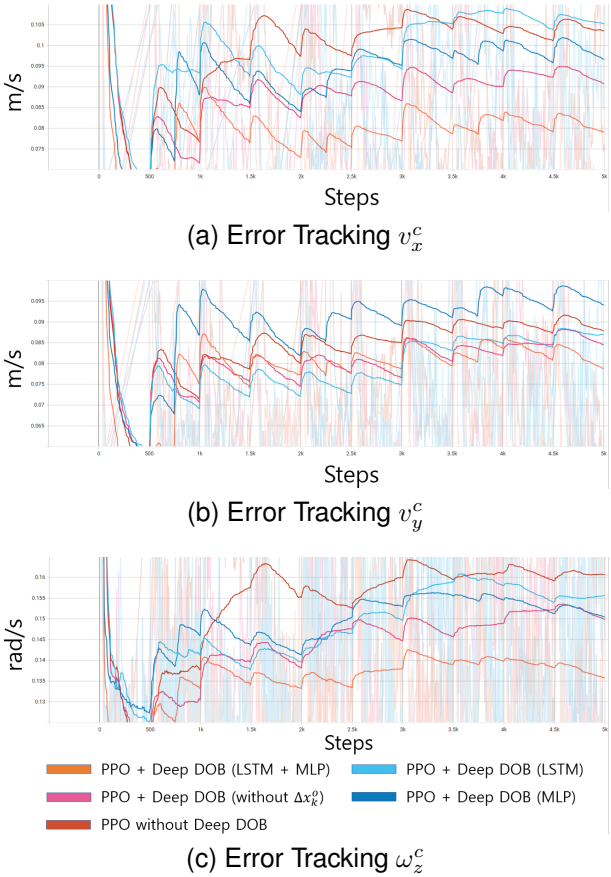
$$q_d = \begin{bmatrix} 0.0 & 0.0 & 0.0 & 0.0 \\ 0.8 & 1.0 & 0.8 & 1.0 \\ -1.5 & -1.5 & -1.5 & -1.5 \end{bmatrix} \tag{22}$$

The optimal deep DOB and deep state estimator are achieved by training them to minimize the observation loss and estimation loss. A supervised learning approach is utilized




(a) Observation loss (b) Estimation loss

Fig. 5. Losses in the training process.




(a) Error Tracking $v_x^c$

(b) Error Tracking $v_y^c$

(c) Error Tracking $\omega_z^c$

Fig. 6. Performance comparison of different types of deep DOB networks.

until the optimal weights for each network are obtained. The hyperparameters used in training for the proposed control framework are listed in Table III.

The observation and estimation losses in Fig.5a and Fig.5b converge to zero. This indicates that the deep DOB and deep state estimator have been successfully trained. Furthermore, incorporating $\Delta x_k^e$ as an input to the state estimator leads to faster convergence compared to when $\Delta x_k^e$ is not included.

### *B. Performance Evaluation*

In the validation process, various types of external disturbances and sensor noise are introduced for uncertain conditions. These external disturbances include additional body mass, a slippery coefficient of friction, and lateral body forces. The magnitude of the external disturbances and sensor noise for uncertain conditions is set higher than those used in the training under nominal conditions.

Fig. 6 shows a performance comparison of various deep DOB architectures under uncertain conditions. The results indicate that the deep DOB integrating an LSTM and MLP network achieves the best performance compared to the baseline without deep DOB. Furthermore, the combination of LSTM and MLP outperforms deep DOBs that use only an MLP or only an LSTM. Specifically, the deep DOB with LSTM + MLP achieves the lowest tracking error among all tested network types. This enhanced performance is attributed to its superior ability to estimate and cancel out uncertainties. Additionally,

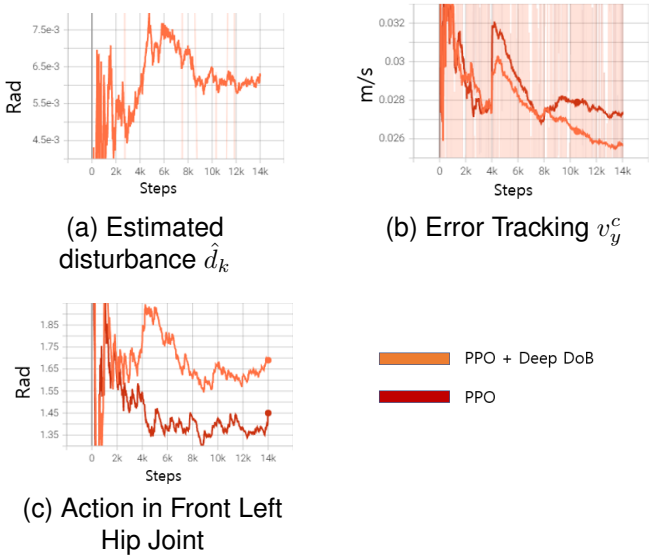



Fig. 7. Lateral force test on the robot body at steps 4k and 8k, and its effects on the front left hip joint action and body velocity error.

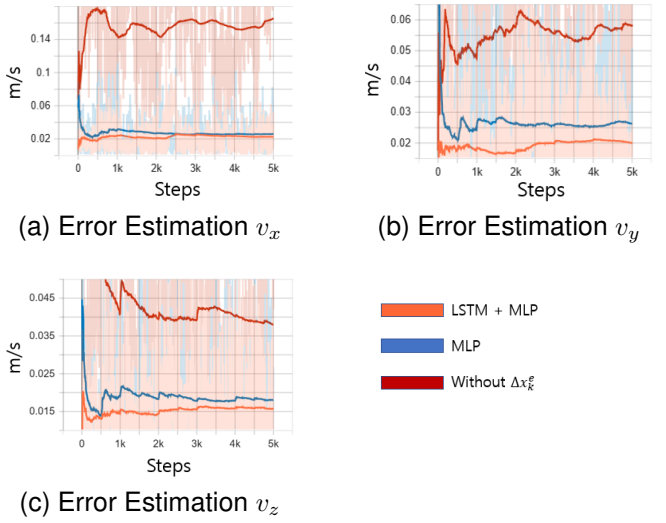



Fig. 8. Performance comparison of different types of deep state estimator networks.

incorporating $\Delta x_k^o$ into the deep DOB improves performance compared to configurations without it.

Fig. 7 compares the locomotion performance of PPO with and without deep DOB under a 200 N lateral body force applied at steps 4K and 8K. The deep DOB output shows fluctuations at these points, representing the estimated disturbances. As these are used to cancel uncertainties, tracking errors in $v_y^c$ decrease, resulting in more robust tracking with PPO + deep DOB. It also demonstrates that PPO with deep DOB performs better than PPO without handling such disturbances. The control actions also reflect stronger responses to counter the disturbance. Unlike Fig. 6, which evaluates various disturbance scenarios, Fig. 7 focuses solely on the push case. Although the improvements appear less significant, the graph still illustrates the characteristic response of deep DOB.

Moreover, Fig. 8 compares the performance of different deep-state estimator networks under uncertain conditions. The deep state estimator using LSTM + MLP outperforms networks that rely solely on MLP or without incorporate $\Delta x_k^e$. The state estimator with LSTM only is omitted, as its performance is significantly poorer. Specifically, the deep state estimator with LSTM + MLP estimates body linear velocity more accurately than the others, demonstrating that combining LSTM with MLP greatly enhances performance compared to using only MLP or LSTM networks.

Additionally, the performance of the proposed control framework is compared with existing methods from the literature. The first baseline is a PPO-based locomotion policy with domain randomization, which is trained under nominal conditions [4]. The second comparison is with a concurrent locomotion policy and state estimator [24]. Table IV shows that the proposed control framework outperforms the baseline and existing approaches in many uncertain scenarios. "x" in that table represents the robot's failure to locomote, while "-" means the robot is unable to estimate the state. Fig. 1 depicts the validation of the proposed framework in real-world scenarios under uncertain conditions.

The real-world experiments demonstrate that the proposed locomotion policy with a deep disturbance observer and deep state estimator outperforms the PPO-based locomotion policy alone when the robot faces various disturbances. The details of the experiment including resulting videos can be found in the link: https://bit.ly/3CF3OTQ

## IV. Conclusion

The proposed control framework, which integrates a locomotion policy with a deep disturbance observer and a deep state estimator, has been designed and validated to enhance robustness in locomotion performance. This robustness is demonstrated through experiments involving various external disturbances and sensor noise scenarios, showing the framework's ability to effectively handle conditions that are not encountered during training. Specifically, integrating LSTM + MLP networks into both the deep DOB and state estimator significantly improves performance compared to using MLP or LSTM alone. The LSTM + MLP architecture leverages temporal information to more accurately predict disturbances and estimate body linear velocity, resulting in more reliable and adaptive locomotion under uncertainty. These validate the effectiveness of incorporating LSTM + MLP networks into the proposed control framework.

## References

[1] M. Hutter, C. Gehring, A. Lauber, F. Gunther, C. D. Bellicoso, V. Tsounis, P. Fankhauser, R. Diethelm, S. Bachmann, M. Bloesch, H. Kolvenbach, M. Bjelonic, L. Isler, and K. Meyer, "ANYmal - toward legged robots for harsh environments," *Advanced Robotics*, vol. 31, no. 17, pp. 918–931, 2017.

[2] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, pp. 1–20, 2019.

[3] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter, "DeepGait: Planning and Control of Quadrupedal Gaits Using Deep Reinforcement Learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3699–3706, 2020.

TABLE IV
PERFORMANCE COMPARISON THE PROPOSED FRAMEWORK WITH BASELINE METHOD.

| | | PPO [4] | | | Concurrent [24] | | | Proposed Framework | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | tracking error | $v_x$ (m/s) | $v_y$ (m/s) | $\omega_z$ (rad/s) | $v_x$ | $v_y$ | $\omega_z$ | $v_x$ | $v_y$ | $\omega_z$ |
| | estimation error | $v_x$ (m/s) | $v_y$ (m/s) | $v_z$ (m/s) | $v_x$ | $v_y$ | $v_z$ | $v_x$ | $v_y$ | $v_z$ |
| Payload 5 Kg | tracking error | 0.1305 | 0.1014 | 0.211 | 0.2855 | 0.1755 | 0.2211 | **0.1193** | **0.097** | **0.1858** |
| | estimation error | - | - | - | 0.2609 | -0.0883 | 0.0903 | **0.1413** | **-0.0808** | **0.019** |
| Friction 0.3 | tracking error | x | x | x | 0.2646 | 0.1845 | 0.2093 | **0.1196** | **0.095** | **0.1903** |
| | estimation error | - | - | - | 0.1931 | -0.0895 | 0.0699 | **0.1127** | **-0.0701** | **0.0206** |
| Force 200 N | tracking error | x | x | x | 0.2855 | 0.1755 | 0.2211 | **0.1198** | **0.1008** | **0.1836** |
| | estimation error | - | - | - | 0.2609 | -0.0976 | 0.0752 | **0.1267** | **-0.0808** | **0.019** |
| Payload 5 Kg + Friction 0.3 | tracking error | x | x | x | 0.277 | 0.1773 | 0.2119 | **0.1146** | **0.0976** | **0.1858** |
| | estimation error | - | - | - | 0.2609 | -0.0976 | 0.0752 | **0.1781** | **-0.0523** | **0.024** |
| Payload 5 Kg + Force 200 N | tracking error | x | x | x | 0.2657 | 0.1847 | 0.2163 | **0.1082** | **0.1007** | **0.1812** |
| | estimation error | - | - | - | 0.2252 | -0.1133 | 0.0776 | **0.1299** | **-0.0487** | **0.0309** |
| Friction 0.3 + Force 200 N | tracking error | x | x | x | 0.2739 | 0.1863 | 0.2292 | **0.1075** | **0.0928** | **0.177** |
| | estimation error | - | - | - | 0.2607 | -0.1153 | 0.0891 | **0.179** | **-0.1779** | **0.0229** |
| Full disturbance | tracking error | x | x | x | 0.2809 | 0.1871 | 0.2243 | **0.12** | **0.0972** | **0.186** |
| | estimation error | - | - | - | 0.2001 | -0.107 | 0.0791 | **0.1541** | **-0.0923** | **0.0305** |


[4] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, "Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning," *CoRL*, pp. 1–14, 2021.

[5] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis, "RLOC: Terrain-Aware Legged Locomotion Using Reinforcement Learning and Optimal Control," *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 2908–2927, 2022.

[6] G. Bellegarda, Y. Chen, Z. Liu, and Q. Nguyen, "Robust High-Speed Running for Quadruped Robots via Deep Reinforcement Learning," *IEEE International Conference on Intelligent Robots and Systems*, pp. 10364–10370, 2022.

[7] H. Wang, H. Luo, W. Zhang, and H. Chen, "Combining Teacher-Student with Representation Learning: A Concurrent Teacher-Student Reinforcement Learning Paradigm for Legged Locomotion," *IEEE Robotics and Automation Letters*, pp. 1–8, 2024.

[8] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," *IEEE International Conference on Intelligent Robots and Systems*, pp. 23–30, 2017.

[9] J. Back and H. Shim, "Adding robustness to nominal output-feedback controllers for uncertain nonlinear systems: A nonlinear version of disturbance observer," *Automatica*, vol. 44, no. 10, pp. 2528–2537, 2008.

[10] I. Ullah and H. L. Pei, "Fixed Time Disturbance Observer Based Sliding Mode Control for a Miniature Unmanned Helicopter Hover Operations in Presence of External Disturbances," *IEEE Access*, vol. 8, pp. 73173–73181, 2020.

[11] Z. Huang and M. Chen, "Coordinated Disturbance Observer-Based Flight Control of Fixed-Wing UAV," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 69, no. 8, pp. 3545–3549, 2022.

[12] H. Hua and Y. Fang, "A Novel Reinforcement Learning-Based Robust Control Strategy for a Quadrotor," *IEEE Transactions on Industrial Electronics*, vol. 70, no. 3, pp. 2812–2821, 2023.

[13] J. Nubert, J. Köhler, V. Berenz, F. Allgöwer, and S. Trimpe, "Safe and Fast Tracking on a Robot Manipulator: Robust MPC and Neural Network Control," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3050–3057, 2020.

[14] D. Kang, J. Cheng, M. Zamora, F. Zargarbashi, and S. Coros, "RL + Model-based Control: Using On-demand Optimal Control to Learn Versatile Legged Locomotion," *IEEE Robotics and Automation Letters*, 2023.

[15] D. Youm, H. Jung, H. Kim, J. Hwangbo, H. W. Park, and S. Ha, "Imitating and Finetuning Model Predictive Control for Robust and Symmetric Quadrupedal Locomotion," *IEEE Robotics and Automation Letters*, vol. 8, no. 11, pp. 7799–7806, 2023.

[16] J. Shi, C. Bai, H. He, L. Han, D. Wang, B. Zhao, M. Zhao, X. Li, and X. Li, "Robust Quadrupedal Locomotion via Risk-Averse Policy Learning," 2023. arXiv preprint arXiv:2308.09405.

[17] S. Li, Y. Pang, P. Bai, J. Li, Z. Liu, S. Hu, L. Wang, and G. Wang, "Learning locomotion for quadruped robots via distributional ensemble actor-critic," *IEEE Robotics and Automation Letters*, vol. 9, no. 2, pp. 1811–1818, 2024.

[18] F. Shi, C. Zhang, T. Miki, J. Lee, M. Hutter, and S. Coros, "Rethinking robustness assessment: Adversarial attacks on learning-based quadrupedal locomotion controllers," 2024. arXiv preprint arXiv:2405.12424.

[19] M. Bloesch, M. Hutter, M. A. Hoepflinger, S. Leutenegger, C. Gehring, C. D. Remy, and R. Siegwart, "State estimation for legged robots: Consistent fusion of leg kinematics and IMU," *Robotics: Science and Systems*, vol. 8, pp. 17–24, 2013.

[20] 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.

[21] J. Lee, J. Hwangbo, and M. Hutter, "Robust recovery controller for a quadrupedal robot using deep reinforcement learning," 2019. arXiv preprint arXiv:1901.07517.

[22] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain," *Science Robotics*, vol. 5, no. 47, pp. 1–14, 2020.

[23] 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, 2022.

[24] 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.

[25] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic Locomotion in the MIT Cheetah 3 Through Convex Model-Predictive Control," *IEEE International Conference on Intelligent Robots and Systems*, pp. 7440–7447, 2018.

[26] R. Grandia, F. Farshidian, R. Ranftl, and M. Hutter, "Feedback MPC for Torque-Controlled Legged Robots," *IEEE International Conference on Intelligent Robots and Systems*, no. 1, pp. 4730–4737, 2019.

[27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," pp. 1–12, 2017. arXiv preprint arXiv:1707.06347.

[28] M. Fikri, S. Herdjunanto, and A. Cahyadi, "On the performance similarity between exponential moving average and discrete linear kalman filter," in *2019 Asia Pacific Conference on Research in Industrial and Systems Engineering (APCoRISE)*, pp. 1–5, 2019.

[29] H. Sussmann and P. Kokotovic, "The peaking phenomenon and the global stabilization of nonlinear systems," *IEEE Transactions on Automatic Control*, vol. 36, no. 4, pp. 424–440, 1991.

[30] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Z. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," *CoRR*, vol. abs/1912.01703, 2019.