Learning-based Adaptive Control of Quadruped Robots for Active Stabilization on Moving Platforms

Minsung Yoon[1], Heechan Shin[1], Jeil Jeong[2], and Sung-Eui Yoon[1†]

Abstract— A quadruped robot faces balancing challenges on a six-degrees-of-freedom moving platform, like subways, buses, airplanes, and yachts, due to independent platform motions and resultant diverse inertia forces on the robot. To alleviate these challenges, we present the Learning-based Active Stabilization on Moving Platforms (*LAS-MP*), featuring a self-balancing policy and system state estimators. The policy adaptively adjusts the robot's posture in response to the platform's motion. The estimators infer robot and platform states based on proprioceptive sensor data. For a systematic training scheme across various platform motions, we introduce platform trajectory generation and scheduling methods. Our evaluation demonstrates superior balancing performance across multiple metrics compared to three baselines. Furthermore, we conduct a detailed analysis of the *LAS-MP*, including ablation studies and evaluation of the estimators, to validate the effectiveness of each component.

I. INTRODUCTION

Quadruped robots have become essential in various real-world applications, such as rescue missions and surveillance, thanks to their adeptness at navigating diverse terrains [1], [2]. A crucial aspect of their deployment is the ability to maintain stability and balance while traversing different types of terrain, including rough and slippery surfaces. To do that, significant progress has been made in various ways, including Reinforcement Learning (RL)-based control strategies, terrain property estimation, foot-terrain interaction analysis, and adaptive mechanical foot designs (refer to Sec. II-A).

On the other hand, as quadruped robots are increasingly deployed in inhabited environments, they have opportunities to utilize transportation platforms like subways, buses, yachts, airplanes, and conveyors for efficient navigation. However, robots in non-inertial reference frames (e.g., moving platforms) encounter distinct balancing challenges compared to those on terrains which are inertial systems. As an example, we can first imagine fictitious inertial forces experienced by passengers in accelerating or turning vehicles. In addition, the absence of prior knowledge about platform movements further complicates prompt reaction to the forces to maintain balance. These challenges highlight the need for advanced control strategies to swiftly adapt to continuously changing platform motions to ensure a balance of the robot.

Thus, our goal is to ensure that quadruped robots maintain balance and avoid falling off moving platforms even with limited space, as illustrated in Fig. 1. Achieving this requires timely and adaptive postural adjustments based on
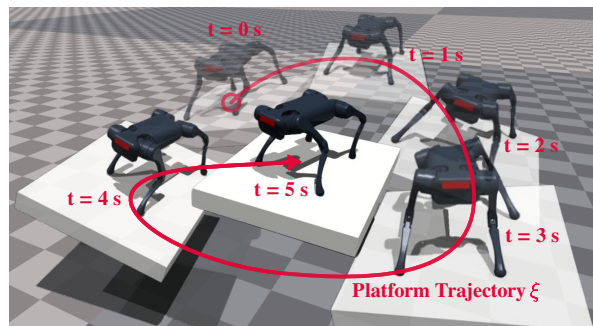

Fig. 1. An illustration demonstrating a balancing process of a quadruped robot on a moving platform exhibiting six-degrees-of-freedom motions.

an understanding of platform movements. To the best of our knowledge, we believe this work is the first effort to address the challenges associated with six-degrees-of-freedom (DoF) and unknown platform motions in three-dimensional space across the fields of quadrupedal locomotion and self-balancing research (refer to Sec. II-B, II-C). As the number of dimensions of the platform's motions increases, the robot encounters a broader array of inertial forces, complicating the balancing task, as further detailed in Sec. III-A.

Main Contributions. We introduce the Learning-based Active Stabilization method on Moving Platforms (*LAS-MP*), designed for adaptive posture adjustment in response to 6-DoF platform motions. This method features a self-balancing policy and system state estimators developed via RL with system identification (refer to Sec. II-D). The estimators infer system states related to the robot and platform (e.g., velocities) alongside the robot's intrinsic properties (e.g., friction), utilizing historical proprioceptive sensor data. These inferred states improve the situational awareness of the policy in non-inertial frames, facilitating adaptive balancing on moving platforms. In addition, we augment the policy with an alignment command, derived from the robot and platform states via feature engineering, which provides explicit velocity guidance to the policy so that it converges better solutions.

To train the policy in environments consisting of diverse platform motions, we introduce platform trajectory generators based on basis-splines (B-splines) [3]. Furthermore, we present a curriculum and scheduling process to progressively escalate task complexity for successful policy learning.

In experiments, we show that the *LAS-MP* enhances the balancing performance across various platform movements, outperforming three other baselines. We also indicate that baseline systems, which are competent in balancing on rough and stationary terrains, struggle on moving platforms. In addition, we carry out ablation studies to elucidate the impact

of explicitly estimating the robot and platform states and the effectiveness of the alignment command.

II. RELATED WORK

A. Quadrupedal Locomotion on Various Terrains

Despite advancements in quadrupedal locomotion on diverse terrains, traditional model-based methods encounter challenges in model accuracy, uncertainty, and adaptability [4]–[6]. Recent deep Reinforcement Learning (RL) offers promising solutions to these challenges [7]–[14]. However, RL methods still face limitations in handling various physical properties of the terrains, such as friction and restitution, when relying solely on geometrical sensor data. Thus, researchers have devised specialized strategies tailored to elastic [15], soft [16]–[18], and slippery [19]–[21] terrains, including foot-terrain interaction modeling and abnormal state detection. In addition, some RL studies [13], [14] simulate virtual external forces on robots to boost the robustness of the locomotion policy. However, these stability challenges only partially resemble the range of perturbations from moving platforms. Thus, we focus on ensuring the robot's safety on the platform in the face of active platform movements.

B. Quadrupedal Locomotion on Moving Platforms

Focusing on operations on moving platforms, recent locomotion research advances traditional model-based controllers and gait planners by integrating platform motions into the analytical models [22], [23]. However, their applicability to diverse platform motions is limited by several assumptions, such as known platform motions with negligible horizontal acceleration, constant body height, negligible angular momentum, and fixed walking cycles with three contact points. These studies explored 2-DoF platform movements featuring sinusoidal pitching and vertical motions, emulating the vessel movements in regular sea waves. In contrast, our research investigates more complex scenarios with 6-DoF platform motions without imposing constraints on the robot and platform's motions. Furthermore, our method operates without prior knowledge of the underlying platform motions, estimating them through state estimators from historical proprioceptive sensor data to adapt to the platform motions.

C. Quadrupedal Self-balancing on Moving Platforms

Rather than the locomotion, other researchers have concentrated on enhancing the postural stabilization of the robot on tilting platforms [24]–[27]. A recent RL-based approach formed an optimal self-balancing policy using table-based RL, integrating kinematic equations but neglecting physical factors like gravity and contact dynamics [26]. A subsequent study solved a continuous optimization problem, employing a parameterized stochastic policy with a physics simulation [27]. While these studies are somewhat in line with our objectives and methods, they experimented on platforms exhibiting 2-DoF (in roll and pitch) and 3-DoF (in z, roll, and pitch) motions with an 8-DoF quadruped robot. In contrast, our research advances to more challenging scenarios, featuring 6-DoF (in x, y, z, roll, pitch, and yaw) platform motions with a 12-DoF quadruped robot for experiments.

D. Privileged Learning and System Identification

System parameters, also known as privileged information in simulations, are pivotal for learning complex skills using RL [13], [28]. Asymmetric structures facilitate the training of actors by providing critics with the parameters [8], [29]. Moreover, actors can leverage the parameters by incorporating the system identification process into the RL framework. Upon deployments, actors should infer the privileged parameters via online optimization [30]–[33] or estimators. Estimators can predict the parameters from sensor data in three ways: their original form [34], [35], latent vectors [9], [11], [36]–[38], or a hybrid way [13]. Adopting the hybrid representation, we deploy two state estimators: one explicitly estimates the robot and platform states in their original form, and the other implicitly infers the intrinsic properties of the robot within a low-dimensional space. These estimations give actors clear situational awareness and domain adaptability, leading to improved balancing performance. Furthermore, we introduce alignment commands, derived by modulating robot and platform states, for enhanced convergence of the policy.

III. LEARNING-BASED ACTIVE STABILIZATION METHOD ON MOVING PLATFORMS (*LAS-MP*)

We present the *LAS-MP*, an advanced controller enabling quadruped robots to maintain balance against 6-DoF platform motions without falling off or toppling on the platform. Below, we discuss the challenges in self-balancing problems on moving platforms along with motivations for the *LAS-MP*, followed by a detailed explanation of each component.

A. Challenges and Motivations

Designing controllers for quadruped robots presents inherent challenges due to their nonlinear, time-varying dynamics encompassing multiple DoF and mechanical constraints [22]. Maintaining balance on moving platforms further demands rapid and suitable responses to external forces on the robot.

Platform motions, represented by continuously changing velocity, expose robots to various forces, including inertia and interaction forces. When platforms simultaneously exhibit both translational and rotational motions, the variety of forces increases. For example, according to Newton's laws of motion, horizontal acceleration induces inertial forces on the robot, sharp turns lead to centrifugal forces, and the platform's rotational motion introduces additional inertial forces around the axis of rotation, like Coriolis, Euler, and centrifugal forces. In addition, interactions through contacts between the robot and the platform generate normal and frictional forces. Thus, upward platform motions can apply vertical reaction forces at contact points, potentially propelling the robot into a flying phase where it loses contact with the platform. These forces heighten the risk of falling off or toppling on the platform unless using a sophisticated algorithm, equipping with a range of motor skills such as soft landings, contact maintenance, foot-slip prevention, and adaptive postural adjustments for weight redistribution.

Meanwhile, the lack of information on platform motions causes delayed responses to the forces, requiring more space
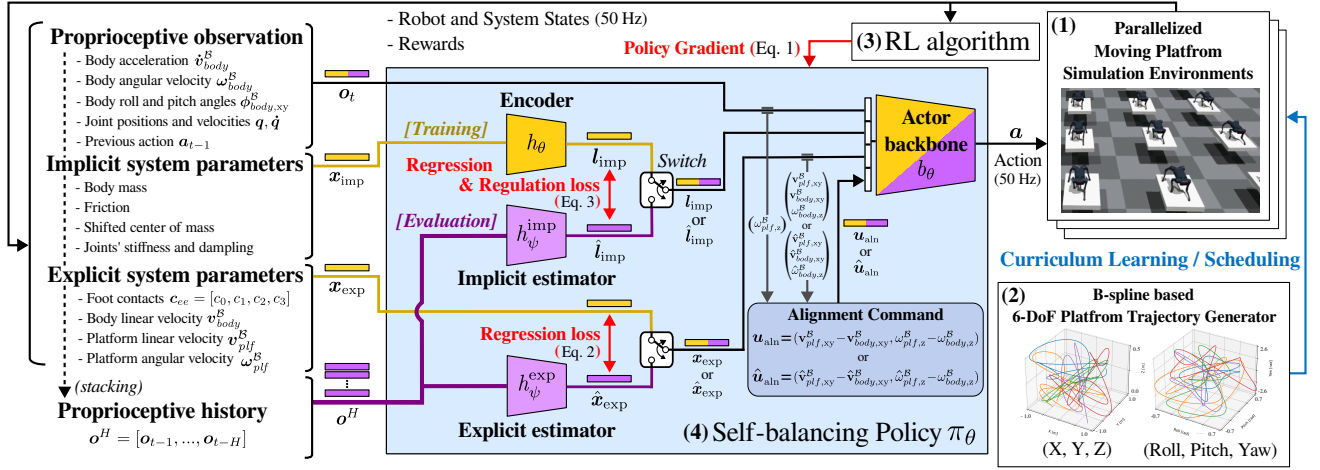
Fig. 2. **Overall framework of the Learning-based Active Stabilization method on Moving Platforms (*LAS-MP*)**. It consists of four key components: (1) parallelized simulation environments for moving platforms, (2) a platform trajectory generator with a scheduling mechanism for managing task complexity progression, (3) a reinforcement learning (RL) algorithm for policy optimization, and (4) a self-balancing policy with two state estimators. To leverage the privileged information in policy learning, we concurrently train the self-balancing policy with the two system state estimators in a single phase by employing the Regularized Online Adaptation (ROA) method [13]. To clearly differentiate components utilized in training or evaluation phases, we designate yellow for training components and purple for evaluation components. A combination of both colors represents components involved in both phases.
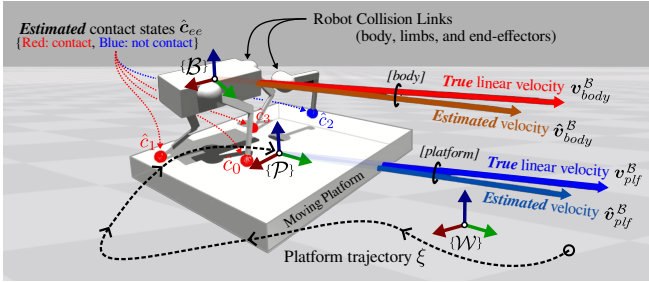

Fig. 3. This figure visually presents the notations used in the descriptions, such as the coordinate frames for the body $\mathcal{B}$, platform $\mathcal{P}$, and world $\mathcal{W}$. It also illustrates estimated robot and platform states, including contact states $\hat{\boldsymbol{c}}_{ee}$ and linear velocities of the body $\hat{\boldsymbol{v}}_{body}^{\mathcal{B}}$ and platform $\hat{\boldsymbol{v}}_{plf}^{\mathcal{B}}$. Platform velocities are visualized after conversion to the platform frame for clarity.

to regain balance or leading to non-recoverable states. This is particularly critical with spatial constraints on platforms.

To mitigate these challenges, we introduce the *LAS-MP*. In physics simulations, we generate diverse platform motions and apply RL to develop the necessary motor skills. We also enhance situational awareness of the policy by inferring robot and platform states in real time using system state estimators, trained via the system identification process, enabling quick and adaptive postural adjustments in the non-inertial frames. Fig. 2 demonstrates a schematic of the *LAS-MP*.

B. Variable Notation

We first introduce the variable notations to elaborate on the *LAS-MP*. As illustrated in Fig. 3, the locations of the robot and platform are determined by their body $\mathcal{B}$ and platform $\mathcal{P}$ frames, located at their respective center of mass. Our notation adopts superscripts for coordinate frames and subscripts for specific entities. For instance, $\boldsymbol{p}_{\langle\cdot\rangle,[\cdot]}^{\mathcal{W}} \in \mathbb{R}^3$ denotes the position within the world coordinate frame $\mathcal{W}$, where $\langle\cdot\rangle$ can be *body*, *plf*, or ee_i and $[\cdot]$ indicates specific elements in a variable if used. For brevity, *plf* and ee_i refer to the platform and each leg's end-effector with i ranging from 0 to 3. $\boldsymbol{v}$ and $\dot{\boldsymbol{v}} \in \mathbb{R}^3$ indicate linear velocity and acceleration, while ϕ and

$\boldsymbol{\omega} \in \mathbb{R}^3$ represent Euler angles and angular velocity. $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, and $\ddot{\boldsymbol{q}} \in \mathbb{R}^{12}$ denote joint positions, velocities, and accelerations of the quadruped robot, respectively. Feet contact states and forces are represented as $\boldsymbol{c}_{ee} = [c_0, c_1, c_2, c_3] \in \mathbb{B}^4$ and $\boldsymbol{f}_{ee} = [\boldsymbol{f}_0, \boldsymbol{f}_1, \boldsymbol{f}_2, \boldsymbol{f}_3] \in \mathbb{R}^{4\times3}$. $\boldsymbol{\xi}$ is assigned to the 6-DoF platform trajectory with a total duration of T. Lastly, we use $\hat{\cdot}$ to indicate estimated values in the subsequent descriptions.

C. Problem Formulation of RL

Our objective of RL is to develop a policy that embodies multiple motor skills required for maintaining balance with minimal deviations on moving platforms. We formulate the self-balancing problem as a Partially Observable Markov Decision Process (POMDP) due to the agent's restricted accessibility to privileged system parameters. The POMDP is composed of a 7-tuple $(\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{F}, \mathcal{R}, \mathcal{Q}_0, \gamma)$, where each element is the state space $\mathcal{S}$, the observation space $\mathcal{O} \subset \mathcal{S}$, the action space $\mathcal{A}$, the transition function $\mathcal{F} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, the reward function $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, the initial state distribution $\mathcal{Q}_0$, and the discount factor $\gamma \in [0, 1)$. The system parameters, denoted as $\mathcal{X} \subset \mathcal{S}$, comprise elements that are not directly observable through proprioceptive sensors, such as platform velocities and feet friction coefficients. For the initial states for each episode $\boldsymbol{s}_0 \sim \mathcal{Q}_0$, we position the robot at the center of the platform with a random yaw angle within a $[-180°, 180°]$ range. We then find a skillful self-balancing policy π_θ that maximizes the expected sum of discounted rewards over various platform movements:

$$J(\pi_\theta) = \mathbb{E}_{\boldsymbol{\xi}_{\text{train}} \sim \Xi_{\text{train}}} \left[\mathbb{E}_{\substack{(\boldsymbol{s},\boldsymbol{a}) \sim \rho_{\pi_\theta} \\ \boldsymbol{s}_0 \sim \mathcal{Q}_0}} \left[\sum_{t=0}^{T} \gamma^t \mathcal{R}(\boldsymbol{s}_t, \boldsymbol{a}_t) \right] \right], \quad (1)$$

where Ξ_{train} is a training set of platform trajectories $\boldsymbol{\xi}_{\text{train}}$ and ρ_{π_θ} is a state-action visitation probability given a stochastic policy π_θ. In the following description, we detail how RL is implemented in the *LAS-MP*, covering architectures, reward functions, training environments, and technical details.

D. Learning Self-balancing Policy

We employ the Regularized Online Adaptation [13] as a privileged learning approach to combine policy optimization and system identification process into a single phase, instead of the two-stage student-teacher approaches [9], [28]. This enables the utilization of the privileged system parameters $\mathcal{X}$ during training in simulation, replacing them with real-time estimates from state estimators upon deployments for evaluations. Thus, the *LAS-MP* is composed of a self-balancing policy and two system state estimators, as shown in Fig. 2.

1) Self-balancing Policy: Our policy π_θ features an actor backbone b_θ and an encoder h_θ, where θ is network parameters. The actor backbone, $b_\theta : \mathcal{O} \times \mathcal{X}_{\exp} \times \mathcal{L}_{\text{imp}} \times se(2) \to \mathcal{A}$, calculates the action $\boldsymbol{a} \in \mathcal{A}$, which is joint displacements $\Delta \boldsymbol{q}$, at each timestep. The action is combined with nominal joint values, yielding target positions, $\boldsymbol{q}_{\text{target}} = \Delta \boldsymbol{q} + \boldsymbol{q}_{\text{nominal}}$, for the Proportional Derivative (PD) controllers [39]. For notational clarity, we omit the notation of t here and after.

The input of the actor backbone b_θ consists of the observation $\boldsymbol{o} \in \mathcal{O}$, explicit system parameters $\boldsymbol{x}_{\exp} \in \mathcal{X}_{\exp}$, an implicit latent vector $\boldsymbol{l}_{\text{imp}} \in \mathcal{L}_{\text{imp}}$, and an alignment command $\boldsymbol{u}_{\text{aln}} \in se(2)$. The observation is a concatenation of proprioceptive sensor data and the previous action, $\boldsymbol{o} = [\dot{\boldsymbol{v}}^{\mathcal{B}}_{body}, \boldsymbol{\omega}^{\mathcal{B}}_{body}, \boldsymbol{\phi}^{\mathcal{B}}_{body,xy}, \boldsymbol{q}, \dot{\boldsymbol{q}}, \boldsymbol{a}_{t-1}]$. For the privileged system parameters $\mathcal{X}$, we categorize them into two subgroups: explicit and implicit ones, $\mathcal{X}_{\exp}$ and $\mathcal{X}_{\text{imp}}$. The explicit system parameters $\boldsymbol{x}_{\exp} \in \mathcal{X}_{\exp}$ include foot contacts and body linear velocity, and platform's linear and angular velocities, $\boldsymbol{x}_{\exp} = [\boldsymbol{c}_{ee}, \boldsymbol{v}^{\mathcal{B}}_{body}, \boldsymbol{v}^{\mathcal{B}}_{plf}, \boldsymbol{\omega}^{\mathcal{B}}_{plf}]$. This input allows the policy to recognize the motions of the reference frame, enabling adaptive postural adjustments in response to platform motions. The implicit system parameters $\boldsymbol{x}_{\text{imp}} \in \mathcal{X}_{\text{imp}}$, representing intrinsic properties of the robot, consist of four coefficients: $\boldsymbol{x}_{\text{imp}} = [\text{body mass, shifted center of mass (CoM), contact friction, stiffness and damping of joints}]$, encoded into the latent vector $\boldsymbol{l}_{\text{imp}}$ by the encoder $h_\theta : \mathcal{X}_{\text{imp}} \to \mathcal{L}_{\text{imp}}$. This vector helps the policy adapt to different intrinsic properties in deployments [9]. The last input, the alignment command $\boldsymbol{u}_{\text{aln}} = [\boldsymbol{v}^{\mathcal{B}}_{plf,xy} - \boldsymbol{v}^{\mathcal{B}}_{body,xy}, \omega^{\mathcal{B}}_{plf,z} - \omega^{\mathcal{B}}_{body,z}]$, is engineered to indicate the robot's relative velocity to the platform. This functions as a body velocity command, guiding the policy to correct misalignment between the robot and the platform.

2) System State Estimators: In deployments, since privileged information is inaccessible, we substitute the input elements of the actor backbone related to the privileged information with real-time estimates derived from the history of observations, $\boldsymbol{o}^H = [\boldsymbol{o}_{t-1}, ..., \boldsymbol{o}_{t-H}] \in \mathcal{O}^H$. To realize that, we employ explicit and implicit state estimators. The explicit estimator, $h^{\exp}_\psi : \mathcal{O}^H \to \mathcal{X}_{\exp}$, estimates the explicit system parameter, $\hat{\boldsymbol{x}}_{\exp} = [\hat{\boldsymbol{c}}_{ee}, \hat{\boldsymbol{v}}^{\mathcal{B}}_{body}, \hat{\boldsymbol{v}}^{\mathcal{B}}_{plf}, \hat{\boldsymbol{\omega}}^{\mathcal{B}}_{plf}]$, as depicted in Fig. 3. We then adopt the estimated explicit parameters to generate the inferred alignment command, $\hat{\boldsymbol{u}}_{\text{aln}} = [\hat{\boldsymbol{v}}^{\mathcal{B}}_{plf,xy} - \hat{\boldsymbol{v}}^{\mathcal{B}}_{body,xy}, \hat{\omega}^{\mathcal{B}}_{plf,z} - \omega^{\mathcal{B}}_{body,z}]$. For the z-directional body angular velocity $\omega^{\mathcal{B}}_{body,z}$, we use the measured value by accessible inertial measurement unit (IMU) sensors. The implicit state estimator, $h^{\text{imp}}_\psi : \mathcal{O}^H \to \mathcal{L}_{\text{imp}}$, estimates the implicit latent

TABLE I. Reward Function of RL: $\mathcal{R} = \mathcal{R}^{\text{task}} + \mathcal{R}^{\text{reg}}$
(Please refer to the description in Sec. III-D.3 for more details.)

Reward	Function Expression
Task Reward: $\mathcal{R}^{\text{task}} = \sum_{k=0}^{4} r_k^{\text{task}}$	
r_0^{task}	$-k_0 \, \mathbb{1}_{collision}(\boldsymbol{q}, \boldsymbol{p}^{\mathcal{P}}_{body}, \phi^{\mathcal{P}}_{body})$
r_1^{task}	$k_1 \, exp(-||\boldsymbol{p}^{\mathcal{P}}_{body,xy}||_2 / k_2)$
r_2^{task}	$k_3 \, exp(-||\boldsymbol{u}_{\text{aln}}||_2 / k_4)$
r_3^{task}	$k_5 \, exp(-||\phi^{\mathcal{W}}_{body,xy}||_2 / k_6)$
r_4^{task}	$k_7 \, exp(-||p^{\mathcal{P}}_{body,z} - h^{body}_{\text{des}}||_2 / k_8)$
Regularization Reward: $\mathcal{R}^{\text{reg}} = \sum_{k=0}^{6} r_k^{\text{reg}}$	
r_0^{reg}	$-(k_9 \, ||\boldsymbol{\tau} - \boldsymbol{\tau}_{t-1}||_2 + k_{10} \, ||\boldsymbol{a} - \boldsymbol{a}_{t-1}||_2)$
r_1^{reg}	$-(k_{11} \, ||\boldsymbol{\tau}||_2 + k_{12} \, ||\dot{\boldsymbol{q}}||_2 + k_{13} \, ||\ddot{\boldsymbol{q}}||_2)$
r_2^{reg}	$-k_{14} \sum_{j=0}^{11} \max(\tau_j \dot{q}_j, 0.0)$
r_3^{reg}	$-k_{15} \sum_{i=0}^{3} \max(||\boldsymbol{f}_i||_2 - f_{\text{tol}}, 0.0)$
r_4^{reg}	$-k_{16} \sum_{i=0}^{3} (t_i^{\text{swing}} - t_{\text{des}}^{\text{swing}})(c_i \, (1 - c_{i,t-1}))$
r_5^{reg}	$-k_{17} \sum_{i=0}^{3} (t_i^{\text{contact}} - t_{\text{des}}^{\text{contact}})((1 - c_i) \, c_{i,t-1})$
r_6^{reg}	$-k_{18} \sum_{i=0}^{3} (||\boldsymbol{v}^{\mathcal{P}}_{ee_i,xy} - (\omega^{\mathcal{P}}_{plf,z} \, \boldsymbol{p}^{\mathcal{P}}_{ee_i,xy} + \boldsymbol{v}^{\mathcal{P}}_{plf,xy})||_2)$

- $\mathbb{1}$: an indicator function, $k_{0,1,...,18}$: non-negative coefficients
- h^{body}_{des}: a desired body height, f_{tol}: a maximum tolerated contact force
- $t_{\text{des}}^{\text{swing}}$: a desired contact duration, $t_{\text{des}}^{\text{contact}}$: a desired swing duration

vector $\hat{\boldsymbol{l}}_{\text{imp}}$ which represents the inferred intrinsic properties of the robot. The network parameters of the state estimators, denoted as ψ, are trained concurrently with the RL objective function (Eq. 1) using Mean Squared Error (MSE) losses:

$$L^{\exp}_{\text{MSE}} = ||\hat{\boldsymbol{x}}_{\exp} - \boldsymbol{x}_{\exp}||_2^2, \tag{2}$$

$$L^{\text{imp}}_{\text{MSE}} = ||\hat{\boldsymbol{l}}_{\text{imp}} - sg[\boldsymbol{l}_{\text{imp}}]||_2^2 + \lambda ||sg[\hat{\boldsymbol{l}}_{\text{imp}}] - \boldsymbol{l}_{\text{imp}}||_2^2, \tag{3}$$

where $sg[\cdot]$ is a stop gradient operator and λ is a Lagrangian multiplier. The first term in each loss function is for regression of each estimator, while the second term of Eq. 3 aims for regularization of the encoder h_θ within the policy π_θ.

3) Reward Function: To develop the essential motor skills using RL for self-balancing tasks on moving platforms, we design the reward function combining task and regularization rewards, represented as $\mathcal{R} = \mathcal{R}^{\text{task}} + \mathcal{R}^{\text{reg}}$. TABLE I shows the detailed composition of each reward function.

We form the task reward $\mathcal{R}^{\text{task}} = \sum_{k=0}^{4} r_k^{\text{task}}$ to encompass the necessary aspects of solving the task. The r_0^{task} penalizes collisions to ensure safety, the r_1^{task} minimizes positional deviations to prevent falls, the r_2^{task} aligns the robot's motion with the platform, and the r_3^{task} reduces body tilt for postural stabilization. The r_4^{task} ensures that the robot maintains its desired body height to preserve its functionality, such as locomotion and manipulation, rather than just lying down.

Adhering to only the task reward might lead the agent into local minima, resulting in unnatural motion and excessive power usage [18], [35]. To mitigate this, we adopt the regularization reward $\mathcal{R}^{\text{reg}} = \sum_{k=0}^{6} r_k^{\text{reg}}$. The r_0^{reg} and r_1^{reg} reduce jerky robot motion, and the r_2^{reg} lessens power consumption. The r_3^{reg} aims to regulate the magnitude of contact forces, and the r_4^{reg} and r_5^{reg} discourage frequent contact and lifting of the feet. Foot replacements should be performed only when required to get a re-balance to prevent deviations caused by the platform's independent movements. Lastly, we propose the r_6^{reg} to penalize the foot slip on the moving platform by taking the underlying platform's motions into account.

TABLE II. Training and Testing Parameter Ranges

Parameters	Training Range	Testing Range
Implicit System Parameters $\boldsymbol{x}_{\mathrm{imp}}$		
Body mass (kg)	$U(4.0, 5.0)$	$U(3.5, 5.5)$
Shifted CoM (m)	$U(-0.2, 0.2)^3$	$U(-0.25, 0.25)^3$
Contact friction	$U(0.8, 1.2)$	$U(0.7, 1.3)$
Stiffness (K_p)	$U(36, 44)^{12}$	$U(32, 48)^{12}$
Damping (K_d)	$U(0.8, 1.2)^{12}$	$U(0.6, 1.4)^{12}$
Platform Trajectory $\boldsymbol{\xi}$		
Number of waypoints	$[5, 6, ..., 15]$	$[4, 5, ..., 16]$
Stiffness (K_p)	$U(1.0, 1.5)^6$	$U(0.5, 2.0)^6$
Damping (K_d)	$U(0.02, 0.03)^6$	$U(0.01, 0.04)^6$

- $U(\cdot, \cdot)$: an uniform distribution

4) Training Environments: We formed parallel simulation environments for efficient data collection during the training phase; each environment consisted of a quadruped robot and a box-shaped moving platform as agents. Each episode starts by positioning the robot on the platform according to the initial state distribution $\mathcal{Q}_0$. To improve generalization for varying intrinsic properties of the robot, the intrinsic parameters $\boldsymbol{x}_{\mathrm{imp}}$ are randomly sampled as described in TABLE II.

To realize platform motions, we parameterized the shape of the platform with [width, length, height] (unit: m), and the PD controllers are used to track the 6-DoF training platform trajectories $\boldsymbol{\xi}_{\mathrm{train}} \in \Xi_{\mathrm{train}}$. We diversified platform motions by equipping each environment with trajectory generators and randomizing controller parameters for every episode. These generators craft a training set of platform trajectories Ξ_{train} utilizing B-spline interpolation [3] with randomly chosen waypoints in the ranges: x and y: $[-1.0, 1.0]$ m, z: $[0.0, 5.0]$ m, roll and pitch: $[-0.7, 0.7]$ rad, and yaw: $[-2.6, 2.6]$ rad, respectively. Each trajectory spans for $T = 10$ s.

Figure 4 shows translational platform trajectory examples and provides statistical analysis for the Ξ_{train}. Adding more waypoints while maintaining a constant duration of 10 s led to increased path lengths and higher speeds on average. Rapid direction changes at high speeds posed significant balancing challenges for the robot, especially at the early stages of training, hindering the collection of meaningful data for policy training. To successfully train the policy, we adopted curriculum learning methods [9], [13], [40], setting task complexity based on the number of waypoints as shown in TABLE II. A scheduler then incrementally escalates the task level after achieving an 80% success rate at each level.

5) Training Details: We adopted the Proximal Policy Optimization [41] with ROA regularization [13], aiming to maximize the RL objective function (Eq. 1) and minimize the regression and regularization losses (Eq. 2, 3). The actor backbone b_θ is constructed using a four-layer Multi-Layer Perceptron (MLP) network, while the encoder h_θ is implemented as a two-layer MLP. Concurrently, both the explicit and implicit state estimators, h_ψ^{exp} and h_ψ^{imp}, are designed with a two-layer 1D Convolutional Neural Network (CNN) in between two one-layer MLPs. The stochastic policy π_θ is conceptualized as a diagonal Gaussian distribution, wherein the mean value is generated by the actor backbone b_θ with a parameterized standard deviation $\boldsymbol{\theta}_{std} \in \mathbb{R}^{12}$. Then, actions

Training Platform Trajectory Examples (trajectory duration: 10 seconds)
- Number of waypoints: **5**   **10**   **15**

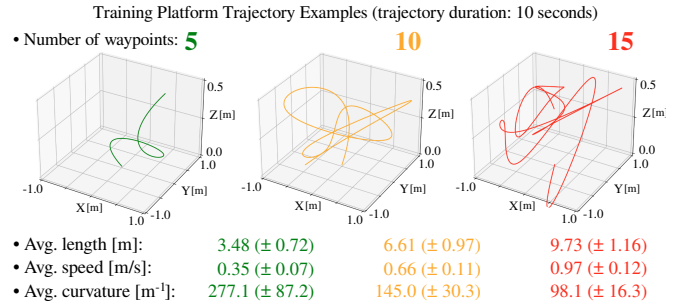|	5	10	15
- Avg. length [m]:	3.48 (± 0.72)	6.61 (± 0.97)	9.73 (± 1.16)
- Avg. speed [m/s]:	0.35 (± 0.07)	0.66 (± 0.11)	0.97 (± 0.12)
- Avg. curvature [m⁻¹]:	277.1 (± 87.2)	145.0 (± 30.3)	98.1 (± 16.3)

Fig. 4. Examples of training platform trajectories Ξ_{train} in translational space and their statistical analysis. A similar trend is observed in rotational space.

are sampled following the distribution, $\boldsymbol{a} \sim \mathcal{N}(b_\theta, \boldsymbol{\theta}_{std})$.

We empirically found the best performing reward coefficients $k_{0,...,18}$ as [10.0, 3.0, 1.2, 2.0, 0.3, 1.0, 0.2, 4.0, 0.1, 10^{-7}, 10^{-4}, 10^{-4}, 10^{-7}, 10^{-6}, 10^{-5}, 0.01, 2.0, 3.0, 0.05]. We also set the parameters H, λ, $h_{\mathrm{des}}^{\mathrm{body}}$, f_{tol}, $t_{\mathrm{des}}^{\mathrm{swing}}$, and $t_{\mathrm{des}}^{\mathrm{contact}}$ as 20, 0.2, 0.37 m, 50 N, 0.1 s, and 0.5 s, respectively.

We utilized the Isaac Gym [42] to operate $8,192$ simulation environments in parallel. As a quadruped robot, we adopted a Unitree A1 [43], which weights 11.74 kg and has dimensions of 48 cm in length, 32 cm in width, and 37 cm in height at the nominal standing configuration $\boldsymbol{q}_{\mathrm{nominal}}$. Additionally, we configured the shape of the platforms as $[2.0, 2.0, 0.2]$ m. The total training consisted of $4,000$ iterations, with each iteration processing data equivalent to 0.24 s. This equates to 96 trajectories $\boldsymbol{\xi}_{\mathrm{train}}$ per environment, which is about 0.8 million trajectories Ξ_{train} when summed across all environments. The overall training required about 2 hours and 30 minutes, executed on a standard desktop configuration with an Intel i9-9900K CPU and an RTX 4090 GPU.

IV. EXPERIMENTAL RESULTS

The major advantage of the *LAS-MP* lies in its balancing proficiency across various platform motions. In experiments, to corroborate the proficiency, we compare the *LAS-MP* with three other baseline methods. We also perform ablation studies to validate the impacts of platform state estimation using explicit estimators (*EE*) and of alignment commands (*AC*). Lastly, we assess the system state estimators to demonstrate their feasibility in inferring system states using a history of proprioceptive sensor data.

A. Evaluation of Balancing Performance

1) Experimental Setup: We aim to evaluate the balancing and domain adaptation capabilities across diverse platform motions and robot intrinsic properties, using a broader range of parameters than those used in training, as shown in TABLE II. To this end, we created an evaluation benchmark set Ξ_{eval} consisting of $10,000$ evaluation platform trajectories ξ_{eval}, following the same procedure in Sec. III-D.4 with the random number of waypoints within the testing range. This benchmark statistically shows a length of 7.12 m, a speed of 0.69 m/s, and a curvature of 132.6 m^{-1} on average in translational space. Fig. 5 displays an evaluation episode of the *LAS-MP* alongside 6-DoF platform movements examples. We also prepared 1,024 quadruped robots having different intrinsic properties for each robot.
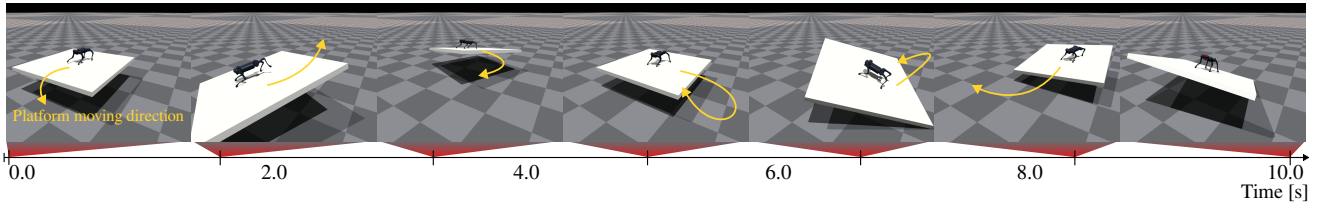
Fig. 5. Snapshots undergoing an evaluation process of the *LAS-MP*, captured with a fixed camera in the world coordinate frame $\mathcal{W}$.

TABLE III. Balancing Performance on Evaluation Benchmark Set Ξ_{eval} consisting of $10,000$ Evaluation Platform Trajectories ξ_{eval}

Method	Constraint violation rate		Deviations from initial pose		Power (W) ↓
	Collision (%*) ↓	Height (%*) ↓	Position (m) ↓	Rotation (rad) ↓	
LAS-MP (ours)	**0.029 (±0.03)**	**0.009 (±0.01)**	**0.366 (±0.01)**	**0.071 (±0.01)**	2.904 (±0.45)
Stand Still	0.612 (±0.29)	0.142 (±0.07)	0.394 (±0.04)	0.182 (±0.07)	**1.031 (±0.17)**
Rough-Static Policy (R-S Policy) [13]	0.922 (±0.12)	0.232 (±0.07)	0.478 (±0.06)	0.241 (±0.05)	8.121 (±2.86)
R-S Policy w/ Oracle Command	0.891 (±0.18)	0.198 (±0.01)	0.452 (±0.05)	0.201 (±0.04)	7.817 (±2.95)
Ablation Study		• %*: a normalized percentage in the range of [0.0, 1.0]			
w/o Alignment Command (AC)	0.287 (±0.27)	0.327 (±0.12)	0.367 (±0.05)	0.195 (±0.08)	4.658 (±1.55)
w/o EE{PLV, PAV} & w/o AC	0.634 (±0.37)	0.454 (±0.29)	0.376 (±0.04)	0.122 (±0.06)	3.388 (±0.83)
w/o EE & w/o AC	0.359 (±0.13)	0.534 (±0.16)	0.496 (±0.03)	0.159 (±0.04)	6.614 (±2.63)
w/o EE & w/o AC & w/ History Obs.	0.314 (±0.19)	0.525 (±0.29)	0.456 (±0.07)	0.177 (±0.09)	6.139 (±4.36)

• *EE*: Explicit Estimator, *PLV*: Platform Linear Velocity, *PAV*: Platform Angular Velocity, *Obs.*: Observations

For evaluation, we organized four metrics: (1) the collision rate on the body and limbs excepting the end-effectors (i.e., feet), (2) the rate of height constraint violations beyond the ± 0.1 m range from the desired body height $h_{\text{des}}^{\text{body}}$, (3) the deviations from initial poses, and (4) the power consumption. Note that we consider falling off the platform as a collision, and height constraint violations and deviations are measured up until the first collision happens.

To demonstrate the effectiveness of our active stabilization method, we prepared the following baselines for comparison:

- *LAS-MP (ours)*: It is our proposed method in Sec. III.
- *Stand Still*: It keeps the robot in a standing posture by setting the nominal configuration q_{nominal} as the target of the PD controller.
- *Rough-Static Policy (R-S Policy)* [13]: It is a robust locomotion policy trained on static and rough terrains with push perturbations on the body. While structurally similar to our method, having explicit and implicit estimators, it does not consider the platform states. Also, the policy is not trained on the moving platform environments. In the experiments, x, y, and yaw velocity commands are set to zero, reflecting the assumption that no prior knowledge is given about the platform's movements.
- *R-S Policy w/ Oracle Command*: It is designed to augment the *R-S Policy* with Oracle information, such as positional information of the robot and platform in the world coordinate frame $\mathcal{W}$. The policy takes the velocity commands, aimed toward the center of the platform, as input. It helps the robot align with the platform movements and compensate for deviations via locomotion.

We prepared the *Stand Still* to verify if the intrinsic structural stability of four-legged robots is adequate for maintaining balance on moving platforms without active adjustments. In addition, we explored the *R-S Policy* to determine if policies trained exclusively in static environments can sustain balance on dynamically moving platforms.

2) Experimental Result: TABLE III shows the balancing performance of each method on the Ξ_{eval} benchmark set. The *LAS-MP* stands out with superior balancing performance, showing minimal constraint violations and deviations. On the other hand, the *Stand Still* shows the least power usage with a reasonable height constraint violation rate, since it remains the standing posture. However, its high collision rate indicates its inadequacy in maintaining balance on the moving platforms, showing the necessity for advanced balancing strategies. Furthermore, the *Rough-Static Policy* exhibits the highest collision rate, and even with the Oracle information, only limited improvement is observed. This result indicates the existence of distinct requirements for getting the balance on moving platforms compared to static environments.

B. Ablation Study

We carried out ablation studies to assess the efficacy of platform state estimation via the explicit estimator (*EE*) and of the alignment command (*AC*) within the *LAS-MP*. Note that in this experiment, we kept the environment setting and learning process consistent with the approach outlined in Sec. III, varying only in the model architecture. TABLE III shows the balancing performance of each architecture. In the *w/o Alignment Command (AC)* case, it shows the policy converged to a local minimum without the *AC*, indicating a performance drop compared to the *LAS-MP*. It represents the *AC* helps the policy converge to a better local minimum by explicitly extracting task-relevant features. As mentioned in [44], feature engineering is crucial for policy learning. In the *w/o EE{PLV, PAV} & w/o AC* study, we adjusted the explicit estimator (*EE*) only to infer the robot-related states, excluding the platform linear velocity (*PLV*) and the

TABLE IV. Prediction Accuracy of Explicit (h_ψ^{exp}) and Implicit (h_ψ^{imp}) State Estimators within the *LAS-MP* on the Ξ_{eval} Benchmark Set

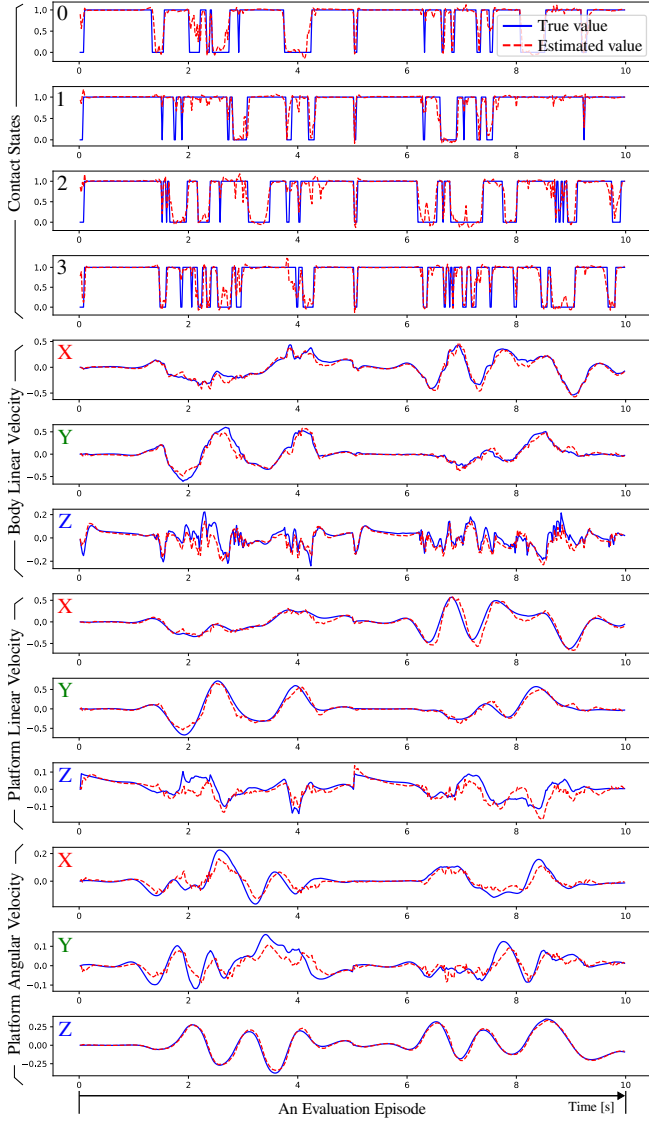| h_ψ^{exp}: $||\hat{\boldsymbol{x}}_{\text{exp},i} - \boldsymbol{x}_{\text{exp},i}||_1 (i = [0, 1, ..., 12])$													h_ψ^{imp}: $||\hat{\boldsymbol{l}}_{\text{imp}} - \boldsymbol{l}_{\text{imp}}||_2$
Foot Contacts $\in \mathbb{R}^4$				Body Linear Velocity $\in \mathbb{R}^3$			Platform Linear Velocity $\in \mathbb{R}^3$			Platform Angular Velocity $\in \mathbb{R}^3$			
0.074	0.077	0.081	0.090	0.035	0.047	0.038	0.040	0.051	0.035	0.027	0.028	0.016	0.0059
(±0.019)	(±0.012)	(±0.008)	(±0.016)	(±0.005)	(±0.005)	(±0.002)	(±0.005)	(±0.005)	(±0.002)	(±0.003)	(±0.003)	(±0.002)	(±0.0018)

Fig. 6. This figure presents the predicted $\hat{\boldsymbol{x}}_{\text{exp}}$ (depicted as red dotted lines) and ground-truth $\boldsymbol{x}_{\text{exp}}$ (shown as blue solid lines) values for each explicit parameter over the course of an evaluation episode spanning $10\,\text{sec}$.
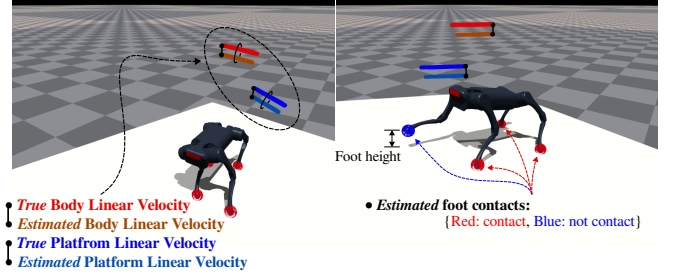

Fig. 7. Snapshots, captured during the evaluation of the *LAS-MP*, show the estimated states from the explicit estimator. To clearly display each velocity, we draw them by moving a certain distance along the z-axis from the center of both the robot and the platform in the world coordinate frame $\mathcal{W}$.

violation rate, power usage, and positional deviation. Lastly, in the case of *w/o EE & w/o AC & w/ History Obs.*, aiming for a fair comparison, we utilized historical proprioceptive sensor data as input for the policy. This resulted in a slight improvement in overall balancing performance. Nonetheless, the policy still converged to the same jumping strategy.

C. Analysis of State Estimators

We further examined the accuracy of estimation results of each estimator within the *LAS-MP*. TABLE IV shows the prediction accuracy of the explicit (h_ψ^{exp}) and implicit (h_ψ^{imp}) estimators. Accuracy is measured by $||\hat{\boldsymbol{x}}_{\text{exp},i} - \boldsymbol{x}_{\text{exp},i}||_1$ for each explicit parameter $i = [0, 1, ..., 12]$ and $||\hat{\boldsymbol{l}}_{\text{imp}} - \boldsymbol{l}_{\text{imp}}||_2$ for implicit parameters. The explicit estimator shows quite low prediction errors for each parameter, indicating it can regress the robot and platform states using historical proprioceptive data. The accuracy of the implicit estimator also exhibits its ability to identify the robot's intrinsic properties during the balancing act. Fig. 6 and 7 display estimation results of the explicit estimator h_ψ^{exp} on some representative situations.

For a more intuitive understanding of the experimental results, please refer to the supplementary video.

V. CONCLUSION

We have presented the Learning-based Active Stabilization Method on Moving Platforms (*LAS-MP*). Through a series of experiments, we have demonstrated that the *LAS-MP* significantly enhances balancing performance on moving platforms exhibiting 6-DoF movements. In addition, experiments have shown that disturbances generated by dynamically moving platforms present a potential safety risk to the baseline systems. In future works, we will extend this work to real-world experiments to verify the feasibility. In addition, although the *LAS-MP* aligns the robot motions with those of the platform by generating instantaneous alignment commands, it lacks a mechanism to correct already established steady-state deviations. Therefore, we plan to localize the center of the platform and reduce deviations via locomotion by utilizing exteroceptive sensor data.

platform angular velocity (*PAV*). This is the same with the *R-S Policy* [13] but trained on moving platform environments. However, it shows higher constraint violation rates. The robot's linear velocity alone, influenced by both robot and platform motions, failed to provide a clear situation awareness in non-inertia frames, introducing noise into the decision-making process. In the *w/o EE & w/o AC* case, we completely omitted the usage of the explicit estimator. Interestingly, it shows improved performance in the collision rate with high power consumption. The converged policy developed a jumping strategy to effectively ignore the diverse disturbances from platforms moving in any direction. However, this naturally led to the increased height constraint

REFERENCES

[1] D. Bellicoso, M. Bjelonic, L. Wellhausen, K. Holtmann, F. Gunther, M. Tranzatto, P. Fankhauser, and M. Hutter, "Advances in real-world applications for legged robots", *Journal of Field Robotics*, vol. 35, no. 8, pp. 1311–1326, 2018.

[2] J. Delmerico, S. Mintchev, A. Giusti, B. Gromov, K. Melo, T. Horvat, C. Cadena, M. Hutter, et al., "The current state and future outlook of rescue robotics", *Journal of Field Robotics*, vol. 36, no. 7, pp. 1171–1191, 2019.

[3] G. D Knott, *Interpolating cubic splines*, vol. 18, Springer Science & Business Media, 1999.

[4] P. Fankhauser, M. Bjelonic, C. D. Bellicoso, T. Miki, and M. Hutter, "Robust rough-terrain locomotion with a quadrupedal robot", in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 5761–5768.

[5] C. Mastalli, W. Merkt, G. Xin, J. Shim, M. Mistry, I. Havoutis, and S. Vijayakumar, "Agile maneuvers in legged robots: a predictive control approach", *arXiv preprint arXiv:2203.07554*, 2022.

[6] G. Lu, T. Chen, X. Rong, G. Zhang, J. Bi, J. Cao, H. Jiang, and Y. Li, "Whole-body motion planning and control of a quadruped robot for challenging terrain", *Journal of Field Robotics*, pp. 1657–1677, 2023.

[7] 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 (T-RO)*, vol. 38, no. 5, pp. 2908–2927, 2022.

[8] I. M. A. Nahrendra, B. Yu, and H. Myung, "Dreamwaq: Learning robust quadrupedal locomotion with implicit terrain imagination via deep reinforcement learning", in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5078–5084.

[9] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots", in *Proceedings of the Robotics: Science and Systems (RSS)*, 2021.

[10] A. Agarwal, A. Kumar, J. Malik, and D. Pathak, "Legged locomotion in challenging terrains using egocentric vision", in *Proceedings of the Conference on Robot Learning (CoRL)*. PMLR, 2023, pp. 403–415.

[11] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning quadrupedal locomotion over challenging terrain", *Science Robotics*, vol. 5, no. 47, pp. eabc5986, 2020.

[12] N. Rudin, D. Hoeller, M. Bjelonic, and M. Hutter, "Advanced skills by learning locomotion and local navigation end-to-end", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 2497–2503.

[13] X. Cheng, K. Shi, A. Agarwal, and D. Pathak, "Extreme parkour with legged robots", *arXiv preprint arXiv:2309.14341*, 2023.

[14] 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*. PMLR, 2022, pp. 91–100.

[15] T. Kim and S. Lee, "Quadruped locomotion on non-rigid terrain using reinforcement learning", *arXiv preprint arXiv:2107.02955*, 2021.

[16] S. Fahmi, M. Focchi, A. Radulescu, G. Fink, V. Barasuol, and C. Semini, "Stance: Locomotion adaptation over soft terrain", *IEEE Transactions on Robotics (T-RO)*, vol. 36, no. 2, pp. 443–457, 2020.

[17] C. Yao, G. Shi, Y. Ge, Z. Zhu, and Z. Jia, "Predict the physics-informed terrain properties over deformable soils using sensorized foot for quadruped robots", in *Proceedings of the International Conference on Advanced Robotics and Mechatronics*. IEEE, 2023, pp. 330–335.

[18] S. Choi, G. Ji, J. Park, H. Kim, J. Mun, J. Lee, and J Hwangbo, "Learning quadrupedal locomotion on deformable terrain", *Science Robotics*, vol. 8, no. 74, pp. eade2256, 2023.

[19] F. Jenelten, J. Hwangbo, F. Tresoldi, C. D. Bellicoso, and M. Hutter, "Dynamic locomotion on slippery ground", *IEEE Robotics and Automation Letters (RA-L)*, vol. 4, no. 4, pp. 4170–4176, 2019.

[20] M. Focchi, V. Barasuol, M. Frigerio, D. Caldwell, and C. Semini, "Slip detection and recovery for quadruped robots", *Robotics Research:*, vol. 2, pp. 185–199, 2018.

[21] Y. Nisticò, S. Fahmi, L. Pallottino, C. Semini, and G. Fink, "On slip detection for quadruped robots", *Sensors*, vol. 22, no. 8, 2022.

[22] A. Iqbal, Y. Gao, and Y. Gu, "Provably stabilizing controllers for quadrupedal robot locomotion on dynamic rigid platforms", *IEEE/ASME Transactions on Mechatronics*, vol. 25, no. 4, pp. 2035–2044, 2020.

[23] A. Iqbal, S. Veer, and Y. Gu, "Analytical solution to a time-varying lip model for quadrupedal walking on a vertically oscillating surface", *Mechatronics*, vol. 96, pp. 103073, 2023.

[24] J. Li, L. Ye, Z. Jin, H. Liu, and B. Liang, "Posture stabilization control for a quadruped robot walking on swaying platforms", in *Proceedings of the International Conference on Automation Science and Engineering (CASE)*. IEEE, 2022, pp. 1959–1964.

[25] F. Grimminger, A. Meduri, M. Khadiv, J. Viereck, M. Naveau, et al., "An open torque-controlled modular robot architecture for legged locomotion research", *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, no. 2, pp. 3650–3657, 2020.

[26] C. Lee and D. An, "Reinforcement learning and neural network-based artificial intelligence control algorithm for self-balancing quadruped robot", *Journal of Mechanical Science and Technology*, vol. 35, pp. 307–322, 2021.

[27] H. Sun, T. Fu, Y. Ling, and C. He, "Adaptive quadruped balance control for dynamic environments using maximum-entropy reinforcement learning", *Sensors*, vol. 21, no. 17, pp. 5907, 2021.

[28] Z. Zhuang, Z. Fu, J. Wang, C. Atkeson, S. Schwertfeger, C. Finn, and H. Zhao, "Robot parkour learning", in *Proceedings of the Conference on Robot Learning (CoRL)*, 2023.

[29] Y. Ma, F. Farshidian, and M. Hutter, "Learning arm-assisted fall damage reduction and recovery for legged mobile manipulators", in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 12149–12155.

[30] W. Yu, K. Liu, and G. Turk, "Policy transfer with strategy optimization", in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[31] X. B. Peng, E. Coumans, T. Zhang, T-W Lee, J. Tan, and S. Levine, "Learning agile robotic locomotion skills by imitating animals", in *Proceedings of the Robotics: Science and Systems (RSS)*, 2020.

[32] W. Yu, V. C. Kumar, G. Turk, and C. K. Liu, "Sim-to-real transfer for biped locomotion", in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 3503–3510.

[33] W. Yu, J. Tan, Y. Bai, E. Coumans, and S. Ha, "Learning fast adaptation with meta strategy optimization", *IEEE Robotics and Automation Letters (RA-L)*, vol. 5, no. 2, pp. 2950–2957, 2020.

[34] W. Yu, J. Tan, K. Liu, and G. Turk, "Preparing for the unknown: Learning a universal policy with online system identification", in *Proceedings of the Robotics: Science and Systems (RSS)*, 2017.

[35] 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 (RA-L)*, vol. 7, no. 2, pp. 4630–4637, 2022.

[36] Z. Fu, X. Cheng, and D. Pathak, "Deep whole-body control: learning a unified policy for manipulation and locomotion", in *Proceedings of the Conference on Robot Learning (CoRL)*. PMLR, 2023, pp. 138–149.

[37] Y. Wang, Z. Jiang, and J. Chen, "Learning robust, agile, natural legged locomotion skills in the wild", in *RoboLetics: Workshop on Robot Learning in Athletics @ CoRL*, 2023.

[38] 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, pp. eabk2822, 2022.

[39] R. P. Borase, D. K. Maghade, S. Y. Sondkar, and S. N. Pawar, "A review of pid control, tuning methods and applications", *International Journal of Dynamics and Control*, vol. 9, pp. 818–827, 2021.

[40] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, "Rapid locomotion via reinforcement learning", *Proceedings of the Robotics: Science and Systems (RSS)*, 2022.

[41] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms", *arXiv preprint arXiv:1707.06347*, 2017.

[42] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, et al., "Isaac gym: High performance gpu-based physics simulation for robot learning", *arXiv preprint arXiv:2108.10470*, 2021.

[43] W. Wang, "Unitree robotics", https://www.unitree.com/, 2024, Accessed on 5th January 2024.

[44] T. Lesort, N. Díaz-Rodríguez, J. Goudou, and D. Filliat, "State representation learning for control: An overview", *Neural Networks*, vol. 108, pp. 379–392, 2018.