Runtime Learning of Quadruped Robots in Wild Environments

Yihao Cai[1], Yanbing Mao[1], Lui Sha[2], Hongpeng Cao[3], and Marco Caccamo[3]

***Abstract*— This paper presents a runtime learning framework for quadruped robots, enabling them to learn and adapt safely in dynamic wild environments. The framework integrates sensing, navigation, and control, forming a closed-loop system for the robot. The core novelty of this framework lies in two interactive and complementary components within the control module: the high-performance (HP)-Student and the high-assurance (HA)-Teacher. HP-Student is a deep reinforcement learning (DRL) agent that engages in self-learning and teaching-to-learn to develop a safe and high-performance action policy. HA-Teacher is a simplified yet verifiable physics-model-based controller, with the role of teaching HP-Student about safety while providing a backup for the robot's safe locomotion. HA-Teacher is innovative due to its real-time physics model, real-time action policy, and real-time control goals, all tailored to respond effectively to real-time wild environments, ensuring safety. The framework also includes a coordinator who effectively manages the interaction between HP-Student and HA-Teacher. Experiments involving a Unitree Go2 robot in Nvidia Isaac Gym and comparisons with state-of-the-art safe DRLs demonstrate the effectiveness of the proposed runtime learning framework. The corresponding open source code is available at github.com/Charlescai123/isaac-runtime-go2.**

I. Introduction

Quadruped robots have become a promising solution for navigating challenging wild environments, such as forests, disaster zones, and mountainous regions [1], [2]. These robots are specifically designed to traverse unstructured terrains, slopes, and dynamic obstacles while maintaining stability and operational efficiency [3]. A typical example is their use in search and rescue tasks, where they assist in locating and aiding victims during earthquakes, building collapses, and other hazardous situations [4].

A. Challenge and Open Problem

Deep reinforcement learning (DRL) has demonstrated considerable success in developing action policies that facilitate agile and efficient locomotion in quadruped robots [5]–[7]. However, quadruped robots are typical safety-critical autonomous systems. A fundamental safety challenge emerges when implementing DRL-enabled locomotion strategies for quadruped robots in wild environments, detailed below.

Challenge: Dynamic Wild Environments. The quadruped robots have dynamics-environments interactions, meaning their behavior and performance depend significantly on the terrains they traverse, such as flat ground versus uneven surfaces, and sandy terrain versus icy conditions. Furthermore, real-time wild environments are often unpredictable and can change unexpectedly, such as during freezing rain [8]. These variations can create a domain gap for the trained locomotion policies, making it difficult for quadruped robots to operate safely and effectively in such environments. Therefore, it is essential to ensure that quadruped robots are resilient to these domain gaps, particularly those arising from dynamic and unpredictable wild environments.

An appealing prospect for addressing the aforementioned safety challenge is the DRL agent's runtime learning for adaptive action policies in wild environments. However, the open problems are *If the DRL agent's actions lead to a safety violation, how can we correct its unsafe learning and guarantee robot's safety in a timely manner? How to adapt to dynamic wild environments for assuring safety?* Before presenting our approach to addressing them, we first review the existing related work.

B. Related Work

DRL-enabled Locomotion. Current DRL frameworks for quadruped robots typically involve pre-training locomotion policies in a source domain, such as a simulator, and then transferring these policies to a target domain, like the real world. The techniques used for addressing domain gaps during this transfer include zero-shot deployment [5], [9], fine-tuning [10], and domain randomization [11], etc. However, these methods do not enable DRL's safe runtime learning in dynamic environments to have continuously adaptive DRL models after deployment. Consequently, they often struggle to handle unexpected environmental variations post-transfer. This limitation poses safety challenges, particularly in unpredictable wild environments that differ significantly from the training conditions in the source domain.

Fault-tolerant DRL. Recent approaches include neural Simplex [12] and runtime assurance [13]–[15]. They treat the DRL agent as a high-performance module (HPM) but a black box that runs in parallel with a verified high-assurance module (HAM). Normally, HPM controls the real plants. HAM takes over once safety violation occurs. These architectures can ensure the safe running of DRL in real plants under the assumption that the real-time wild environments do not cause HAM to fail, which is not practical for quadruped robots in dynamic and unpredictable wild environments. Specifically, HAM is the static model-based controller, and its action will be unreliable if the real-time wild environments create a significant model mismatch for HAM design.

[1]Yihao Cai and Yanbing Mao are with Engineering Technology Division, Wayne State University, Detroit, MI 48201, USA {yihao.cai, hm9062}@wayne.edu

[2]Lui Sha is with Department of Computer Science, University of Illinois Urbana-Champaign, Champaign, IL, 61820, USA lrs@illinois.edu

[3]Hongpeng Cao and Marco Caccamo are with School of Engineering and Design, Technical University of Munich, Munich, 85748, Germany {cao.hongpeng, mcaccamo}@tum.de

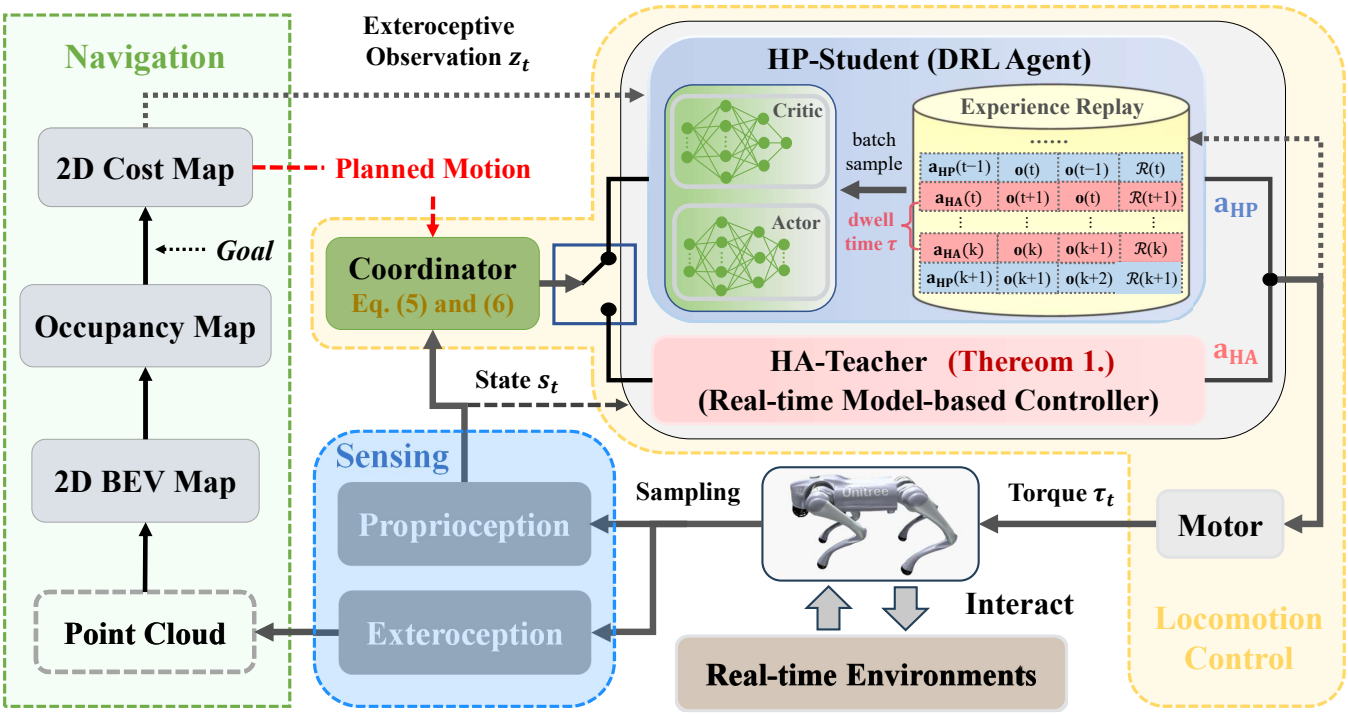


Fig. 1: Runtime Learning Framework: A seamless integration of perception, planning, learning, and control.

The **sensing** module outputs both proprioceptive and exteroceptive data from the robot operating in wild environments. The **navigation** module constructs a Bird's-Eye View (BEV) map by filtering Regions of Interest, generates an occupancy map using height thresholding, and computes the cost map with the Fast Marching method. Based on the cost map, a planner generates reference commands for the locomotion controller while also providing visual observations for the HP-Student. In the **locomotion control** module, the coordinator manages the interaction between the HA-Teacher and the HP-Student to ensure the robot's safety. This setup allows the HP-Student to learn from the HA-Teacher through $\mathbf{a}_{\mathrm{HA}}$ while also engaging in self-learning by $\mathbf{a}_{\mathrm{HP}}$ to develop a safe and high-performance action policy.

C. Contribution: Runtime Learning Framework

To address the aforementioned safety challenge associated with the quadruped robots operating in dynamic wild environments, we introduce our runtime learning framework shown in fig. 1, which integrate perception, planning, learning, and control. This framework enables quadruped robots to autonomously navigate and safely learn in real-time wild environments. Beyond addressing these safety concerns, this framework also tackles fundamental open problems in DRL-enabled locomotion and fault-tolerant DRL. Its core novelties are outlined as below:

- HP-Student – a DRL agent, which performs both self-learning and learning-from-HA-Teacher for robots operating in real-time wild environments, targeting a safe and high-performance action policy.
- HA-Teacher – a simplified yet verifiable controller based on physics models, designed solely to ensure safety. Its mission is to teach HP-Student about safe policies while backing up the robot's safe control. HA-Teacher utilizes a *real-time* physics model, *real-time* action, and *real-time* control goals, all tailored to respond effectively to complex and dynamic environments, ensuring robot's runtime safety.

The interaction between HP-Student and HA-Teacher operates as follows: When the real-time status of the robot controlled by HP-Student approaches the safety boundary, HA-Teacher intervenes to enforce the robot's safe control. During this period, HP-Student will learn from the HA-Teacher regarding the safe action policies (i.e., teaching-to-learn). Once the robot's real-time status recovers from the safety boundary, control is handed back to HP-Student for continuing its self-learning. This interactive design fosters HP-Student's learning of a safe and high-performance policy under dynamic wild environments.

II. Preliminaries

A. Notation

We use $\mathbf{P} \succ 0$ to represent that $\mathbf{P}$ is a positive definite matrix. $\mathbb{R}^n$ indicates the set of n-dimensional real vectors, while $\mathbb{N}$ denotes the set of natural numbers. The superscript '$\top$' indicates matrix transposition. We define $\mathbf{0}_m$ as an m-dimensional zero vector and $\mathbf{I}$ as the identity matrix of appropriate dimensions. The notation $\mathbf{c} > \mathbf{0}_n$ implies that the vector $\mathbf{c} \in \mathbb{R}^n$ is positive, meaning all its elements are strictly positive. Lastly, $\mathbf{diag}\{\mathbf{c}\}$ is diagonal matrix whose diagonal entries are given by the elements of $\mathbf{c}$.

B. Definitions

For quadruped robots, we define the following safety and action sets, to which both system states and action commands must always be constrained.

$$\mathbb{S} \triangleq \{\mathbf{e} \in \mathbb{R}^n | -\mathbf{c} \leq \mathbf{C} \cdot \mathbf{e} \leq \mathbf{c}, \quad \text{with } \mathbf{c} > \mathbf{0}_p\}, \tag{1}$$

$$\mathbb{A} \triangleq \{\mathbf{a} \in \mathbb{R}^m \mid -\mathbf{d} \leq \mathbf{D} \cdot \mathbf{a} \leq \mathbf{d}, \quad \text{with } \mathbf{d} > \mathbf{0}_q\}, \tag{2}$$

where $\mathbf{e} \triangleq \mathbf{s} - \mathbf{r}$ denotes the tracking error of proprioceptive sampling $\mathbf{s}$ with respect to the planned motion $\mathbf{r}$ from the navigation module shown in fig. 1. Meanwhile, $\mathbf{C}$ and $\mathbf{c}$ are provided in advance to describe $p \in \mathbb{N}$ safety conditions, while $\mathbf{D}$ and $\mathbf{d}$ are used to describe $q \in \mathbb{N}$ conditions on physically-feasible action space for safe locomotion control. The inequalities outlined in eq. (1) are sufficiently general to cover various safety conditions, such as robot's velocity regulation, yaw control for avoiding collisions, and management of the center of gravity to prevent falling.

In our framework, the HP-Student (i.e., DRL agent) will engage in self-learning and teaching-to-learn for a safe and high-performance action policy. To understand them better, we refer to the safety set (1) and introduce the self-learning space of HP-Student:

$$\mathbb{L} \triangleq \{\mathbf{e} \in \mathbb{R}^n | -\eta \cdot \mathbf{c} \leq \mathbf{C} \cdot \mathbf{e} \leq \eta \cdot \mathbf{c}, \ 0 < \eta < 1\}. \tag{3}$$

Given $0 < \eta < 1$, and $\mathbf{c} > \mathbf{0}_p$, it follows directly from eqs. (1) and (3) that the self-learning space is a subset of the safety set, i.e., $\mathbb{L} \subset \mathbb{S}$. The intentional gap between these two sets accounts for the system's response time and decision-making latency, which stem from physical constraints, computational limitations, and communication delays. This means when system states arrive at a safety boundary, action policy cannot immediately drive it back to the safety set, without time delay. As shown in fig. 2, the boundaries of the self-learning space $\mathbb{L}$ can thus be regarded as the marginal-safety boundaries. Based on the set, we introduce the definition of HP-Student's safe action policy.

Definition 1 (Safe Action Policy of HP-Student)**.** *Consider the self-learning space* $\mathbb{L}$ (3)*. The action policy of the DRL-agent (i.e., HP-Student) is said to be safe if, under its control, the robot's state satisfies that given* $\mathbf{e}(1) \in \mathbb{L}$*, the* $\mathbf{e}(t) \in \mathbb{L}$ *holds for all time steps* $t \in \mathbb{N}$*.*

In our framework fig. 1, the HA-Teacher serves as a physics-model-based controller dedicated solely to ensuring safety. When HP-Student's real-time actions are unsafe according to definition 1, the HA-Teacher will instruct the HP-Student about safety, fostering a "teaching-to-learn" paradigm, while backup the robot's safe locomotion control simultaneously. As indicated in eqs. (1) and (3) and fig. 2, the set $\mathbb{S} \setminus \mathbb{L}$ defines the HP-Student's teaching-to-learn space, also referred to as the marginally-safe space. We note the action policy of the HA-Teacher must have assured safety; otherwise, its safety teaching will mislead HP-Student. Therefore, we introduce the definition of "assured safety," which guides the design of HA-Teacher later.

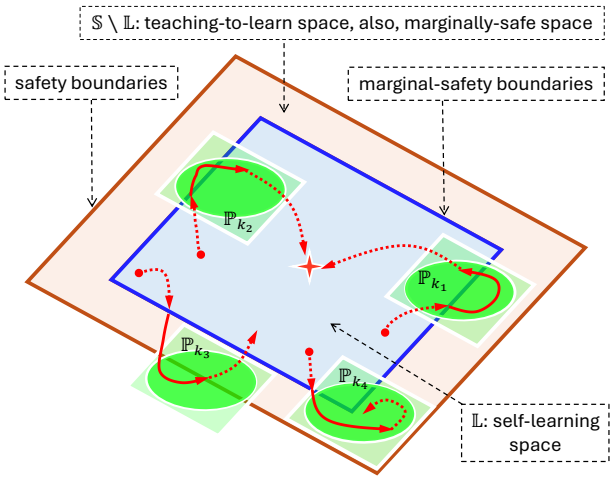


Fig. 2: Illustrations of self-learning space, teaching-to-learn space (also referred to as marginally-safe space), real-time patches, safety boundaries, and marginal-safety boundaries. The $\mathbb{P}_{k_1}$ and $\mathbb{P}_{k_2}$ are successful real-time patches designed by theorem 1, while $\mathbb{P}_{k_3}$ and $\mathbb{P}_{k_4}$ are failure cases since robot system under their control either leaves the safety set $\mathbb{S}$ (by $\mathbb{P}_{k_3}$) or cannot return to the self-learning space $\mathbb{L}$ (by $\mathbb{P}_{k_4}$).

Definition 2 (Safe Action Policy of HA-Teacher)**.** *Consider safe set* $\mathbb{S}$ (1)*, action set* $\mathbb{A}$ (2)*, self-learning space* $\mathbb{L}$ (3)*, and set teaching horizon as* $\tau \in \mathbb{N}$ *and teaching period as*

$$\mathbb{T}_k^\tau \triangleq \{k+1, \ k+2, \ \ldots, \ k+\tau\}. \tag{4}$$

HA-Teacher's action policy denoted by $\pi_{HA}(\cdot)$ *is said to have assured safety, if* $\mathbf{e}(k) \in \mathbb{S} \setminus \mathbb{L}$*, then i) the* $\mathbf{e}(t) \in \mathbb{S}$ *holds for any time* $t \in \mathbb{T}_k^\tau$*, ii)* $\mathbf{a}_{HA}(t) = \pi_{HA}(\mathbf{e}(t)) \in \mathbb{A}$ *holds for any time* $t \in \mathbb{T}_k^\tau$*, and iii)* $\mathbf{s}(k+\tau) \in \mathbb{L}$*.*

By definition 2, the HA-Teacher is deemed trustworthy for guiding the HP-Student in safety learning only if its action policy satisfies the following conditions: i) it must ensure the robot consistently adheres to the safety regulations within $\mathbb{S}$; ii). it must keep the real-time actions within a physically-feasible action space $\mathbb{A}$; and iii) it must ensure the robot states return to the self-learning space $\mathbb{L}$ as teaching session ends. Notably, if condition iii) is not met, the HA-Teacher will dominate, preventing the HP-Student from developing a high-performance action policy through self-learning.

III. Runtime Learning Framework

Referring to fig. 1, we describe the design of our runtime learning framework for enabling safe runtime learning in dynamic wild environments.

A. Sensing Module

The sensing module provides both proprioceptive and exteroceptive data from the robot's onboard sensors. Proprioceptive sensing delivers robot's state data that includes: 1) direct readings from the Inertial Measurement Unit, gyroscope, and motor encoders, which measure orientation and angular velocity; and 2) estimated center of mass (CoM) height (using a Kalman filter), and velocities in the CoM-x,

CoM-y, and CoM-z directions. Exteroceptive sensing enables the perception of external terrain and obstacles. It creates 3D point cloud data using a depth camera, aiding in environmental mapping and obstacle avoidance for navigation.

B. Navigation Module

During runtime learning, the navigation module functions as a high-level motion planner, generating planned motions for the robot's locomotion controller to navigate wild environments. As shown in fig. 1, the pipeline includes: 1) constructing a Bird's-Eye View (BEV) map and an occupancy map; ii) applying the Fast Marching Method (FMM) to compute a 2D cost map; and iii) planning motions.

*1) **2D Map Construction**:* The point cloud data is obtained from the quadruped robot's depth camera and transformed into the world frame. To enhance computational efficiency, we filter the point cloud using a Region of Interest, which allows us to retain only the relevant environmental features. The filtered data is then projected onto a discretized 2D occupancy grid, with each grid cell encoding local terrain characteristics. This process helps create the BEV map. Next, to further process the BEV representation, we generate a binary occupancy map based on a pre-defined maximum height of interest. In this map, any region that exceeds the height of the robot's body is classified as an obstacle. Within the occupancy map, we project the goal point from the world frame onto the map and apply the FMM to compute a 2D cost map, which will guide motion planning [16]. The 2D cost map generation pipeline can be found in fig. 3.

*2) **Motion Planning**:* Based on the robot's position on the 2D cost map and its field of view, multiple short-term goals are identified. For each candidate goal, potential velocity sets are calculated, taking into account the robot's current velocity and control frequency. To ensure smooth movement, spline-based interpolation is employed to generate continuous motion references. The optimal reference is selected by minimizing the accumulated cost along the path on the map. Finally, this optimal motion reference is sent to the locomotion control module, allowing for seamless navigation with dynamically feasible motions.

C. Locomotion Module

This module consists of two components: HA-Teacher, which is a real-time physics-model-based controller, and HP-Student, which is a DRL agent. Their interactive design incorporates the core innovations of the proposed runtime learning framework. The motivation for developing this module stems from the two common approaches to achieving locomotion control in quadruped robots. The first approach is data-driven DRL, which delivers superior performance but poses challenges regarding verifiable safety. This is due to the vast number of parameters in DNNs, nonlinear activation functions, and various random factors. On the other hand, the physics-model-based controller (e.g., LQR) offers verifiable safety and stability, but its performance is often limited due to model mismatches. The characteristics of both approaches inspire us to integrate them, aiming to bring safe runtime learning into reality. Next, we detail the design.

*1) **Coordinator**:* As shown in fig. 1, the locomotion module has a coordinator, which is responsible for managing interactions between HP-Student and HA-Teacher through monitoring the condition:

$$\mathbf{e}(k-1) \in \mathbb{L} \text{ and } \mathbf{e}(k) \in \mathbb{S} \setminus \mathbb{L} \tag{5}$$

by which, the data series in eq. (19) for teaching-to-learn in $\mathbb{S} \setminus \mathbb{L}$ can be generated, and the switching logic of actions applied to the robot for safe locomotion is as follows:

$$\mathbf{a}(t) \leftarrow \begin{cases} \mathbf{a}_{\text{HA}}(t), & \text{if condition (5) holds and } t \in \mathbb{T}_k \\ \mathbf{a}_{\text{HP}}(t). & \text{otherwise} \end{cases}. \tag{6}$$

*2) **HA-Teacher: Safety Only**:* HA-Teacher's action policy is designed to be adaptive to real-time wild environments to assure safety only. Quadruped robots have dynamics-environment interactions. When a real-time environment creates safety issues, it is crucial to update the dynamics models, action policy, and control goals promptly to ensure safe and effective responses in real time. This insight has inspired us to develop a real-time patch for HA-Teacher:

$$\text{Patch: } \mathbb{P}_k \triangleq \{\, \mathbf{e} \in \mathbb{R}^n \,|\, -\theta \cdot \mathbf{c} \le \mathbf{C} \cdot (\mathbf{e} - \mathbf{e}_k^*) \le \theta \cdot \mathbf{c}, \; 0 < \theta < 1 \,\}, \tag{7}$$

where $\mathbf{e}_k^*$ represents the patch center and serves as the real-time control goal, which is defined below.

$$\mathbf{e}_k^* \triangleq \chi \cdot \mathbf{e}(k), \text{ where } 0 < \chi < 1 \text{ and condition (5) holds.} \tag{8}$$

With the control goal at hand, we introduce the following real-time dynamics model of tracking errors w.r.t. $\mathbf{e}_k^*$, which is derived from the robot's dynamics model in [17].

$$\widehat{\mathbf{e}}(t+1) = \mathbf{A}(\mathbf{s}(k)) \cdot \widehat{\mathbf{e}}(t) + \mathbf{B}(\mathbf{s}(k)) \cdot \mathbf{a}_{\text{HA}}(t) + \mathbf{h}(\widehat{\mathbf{e}}(t)), \quad \text{for } t \in \mathbb{T}_k^\tau \triangleq \{k+1, k+2, \ldots, k+\tau\} \tag{9}$$

where $\widehat{\mathbf{e}}(t) \triangleq \mathbf{e}(t) - \mathbf{e}_k^*$, $\mathbf{h}(\widehat{\mathbf{e}}(t))$ is model mismatch, and $\mathbf{s} = [h, \Theta, v, \omega]^\top \in \mathbb{R}^{10}$ with h being CoM height, $v = [\text{CoM x-velocity, CoM y-velocity, CoM z-velocity}]^\top$, $\Theta = [\text{roll, pitch, yaw}]^\top$, and w being the angular velocity of Θ in world coordinates. The $(\mathbf{A}(\mathbf{s}(k)), \mathbf{B}(\mathbf{s}(k)))$ in eq. (9) is the available knowledge of physics model for designing HA-Teacher's action policy:

$$\mathbf{a}_{\text{HA}}(t) = \mathbf{F}_k \cdot \widehat{\mathbf{e}}(t) = \mathbf{F}_k \cdot (\mathbf{e}(t) - \mathbf{e}_k^*), \quad t \in \mathbb{T}_k^\tau \tag{10}$$

Hereto, we summarize HA-Teacher's working mechanism by considering eqs. (7) to (10). When the real-time states of the robot, controlled by HP-Student, move from safe self-learning space $\mathbb{L}$ to marginally-safe space $\mathbb{S} \setminus \mathbb{L}$, HA-Teacher uses the most recent sensor data, $\mathbf{s}(k)$, to update the physics model, denoted as $(\mathbf{A}(\mathbf{s}(k)), \mathbf{B}(\mathbf{s}(k)))$. This update facilitates the computation of both the real-time patch and the coupled action policy. The patch center, represented as $\mathbf{e}_k^* = \chi \cdot \mathbf{e}(k)$, is situated within HP-Student's self-learning space and aligns with HA-Teacher's real-time control objectives. This setup defines HA-Teacher's teaching task: to guide HP-Student towards an action policy that achieves high performance while ensuring safety at the safety boundaries.

Next, we present the design of HA-Teacher's action policy. Before proceeding, we outline a practical and common assumption regarding the model mismatch $\mathbf{h}(\mathbf{e}(k))$.

Assumption 1. *The model mismatch in* $\mathbf{h}(\cdot)$ *in eq.* (9) *is locally Lipschitz in* $\mathbb{P}_t$ (7)*, i.e.,*

$$(\mathbf{h}^\top(\mathbf{e}_1)-\mathbf{h}(\mathbf{e}_2))^\top\cdot\mathbf{P}_k\cdot(\mathbf{h}(\mathbf{e}_1)-\mathbf{h}(\mathbf{e}_2)) \leq \kappa\cdot(\mathbf{e}_1-\mathbf{e}_2)^\top\cdot\mathbf{P}_k\cdot(\mathbf{e}_1-\mathbf{e}_2),\ \mathbf{P}_k\succ 0,\ \forall\mathbf{e}_1,\mathbf{e}_2\in\mathbb{P}_t.$$

We present the design of HA-Teacher's safety-assured action policy in the following theorem. Due to page limit, its detailed proof is provided in [18].

Theorem 1. *[18] Consider the self-learning space* (3) *and the real-time patch* (7) *with its patch center* (8)*, with their parameters satisfying:*

$$\theta+\chi\cdot\eta<1. \tag{11}$$

Meanwhile, compute the matrix $\mathbf{F}_k$ *in HA-Teacher's action policy in eq.* (10) *according to*

$$\mathbf{F}_k=\mathbf{R}_k\cdot\mathbf{Q}_k^{-1},\qquad \mathbf{P}_k=\mathbf{Q}_k^{-1}, \tag{12}$$

with $\mathbf{R}_k$ *and* $\mathbf{Q}_k$ *satisfying:*

$$\begin{bmatrix}\mathbf{Q}_k & \mathbf{R}_k^\top\\ \mathbf{R}_k & \mathbf{T}_k\end{bmatrix}\succ 0, \tag{13}$$

$$\mathbf{I}-\overline{\mathbf{C}}\cdot\mathbf{Q}_k\cdot\overline{\mathbf{C}}^\top\succ 0, \tag{14}$$

$$\mathbf{I}-\overline{\mathbf{D}}\cdot\mathbf{T}_k\cdot\overline{\mathbf{D}}^\top\succ 0, \tag{15}$$

$$\mathbf{Q}_k-n\cdot\boldsymbol{diag}^2(\mathbf{s}(k))\succ 0, \tag{16}$$

$$\begin{bmatrix}(\alpha-(1+\frac{1}{\gamma})\cdot\kappa)\cdot\mathbf{Q}_k & \mathbf{Q}_k\cdot\mathbf{A}^\top(\mathbf{s}(k))+\mathbf{R}_k^\top\cdot\mathbf{B}^\top\\ \mathbf{A}(\mathbf{s}(k))\cdot\mathbf{Q}_k+\mathbf{B}\cdot\mathbf{R}_k & (1+\gamma)^{-1}\cdot\mathbf{Q}_k\end{bmatrix}\succ 0, \tag{17}$$

where $\overline{\mathbf{C}}\triangleq\mathbf{C}\cdot\boldsymbol{diag}^{-1}\{\theta\cdot\mathbf{c}\}$, $\overline{\mathbf{D}}=\mathbf{D}\cdot\boldsymbol{diag}^{-1}\{\mathbf{d}\}$, $\gamma>0$, $0<\alpha<1$*, and* κ *is the Lipschitz bound given in assumption 1. According to definition 2, HA-Teacher's action policy has assured safety.*

We note that using the CVX toolbox [19], the $\mathbf{Q}_k$ and $\mathbf{R}_k$ can be computed from eqs. (13) to (17) for the $\mathbf{F}_k$.

3) ***HP-Student: Self-Learning and Teaching-to-Learn****:* HP-Student will learn from the HA-Teacher for a safe action policy in the teaching-to-learn space and engage in self-learning for a high-performance action policy in the self-learning space. The integration of these two learning paradigms enables the HP-Student to achieve a safe and high-performance action policy. The teaching horizon of the HA-Teacher is a crucial design. Specifically, during the teaching period, HA-Teacher generates a sequence of safe experience tuples, stored in HP-Student's replay buffer for safety learning, as illustrated in fig. 1. This allows HP-Student to progressively internalize safety constraints, ensuring its learned policies adhere to safe operational bounds.

If the HA-Teacher's action policy cannot guide the robot to return to the self-learning space for a high-performance action policy (as illustrated by $\mathbb{P}_{k4}$ in fig. 2), 'Teach-to-Learn' will take precedence over 'Self-Learn' solely for the sake of learning safety. We note Theorem 1 presents a design for a safety-assured action policy; however, the computing of the teaching horizon remains open. Hence, we offer guidance on the reasonable teaching horizons, formally stated in the following theorem. Due to page limit, its proof is in [18].

Theorem 2 (Teaching Horizon [18]**).** *Given parameters* $\epsilon>0$*,* $0<\chi<1$*, and* $0<\alpha<1$*, if HA-Teacher's teaching horizon* τ *satisfies:*

$$\tau\geq\tau_{\min}\triangleq\frac{\ln(1-(1+\epsilon)\cdot\chi^2)-2\ln(1+\epsilon^{-1})-\ln(1+\chi)}{\ln\alpha},$$

we have $\mathbf{e}(k+\tau)\in\mathbb{L}$*, i.e., the robot states can come back to HP-Student's self-learning space.*

With the teaching horizon in sight, we can proceed to present the self-learning and teaching-to-learn.

Self-Learning. As shown in fig. 1, HP-Student adopts an *actor-critic* architecture [20], [21] in DRL, with an experience replay buffer to enhance sample efficiency and mitigate temporal correlations. Its observation space is structured as: $\mathbf{o}_t=[\mathbf{z}_t,\mathbf{e}_t]^T$, where $\mathbf{z}_t$ is the exteroceptive observation, and $\mathbf{e}_t$ is the proprioceptive tracking error: $\mathbf{e}_t\triangleq\mathbf{s}_t-\mathbf{s}_d$, with $\mathbf{s}_d$ being the desired state. The self-learning objective is to optimize the action policy $\mathbf{a}_{\text{HP}}(k)=\pi(\mathbf{o}(k))$ which maximizes the expected return from the initial state distribution:

$$\mathcal{Q}^\pi(\mathbf{e}(k),\mathbf{a}_{\text{HP}}(k)) = \mathbf{E}_{\mathbf{e}(k)\sim\mathbb{L}}\left[\sum_{t=k}^{\infty}\gamma^{t-k}\cdot\mathcal{R}\left(\mathbf{e}(t),\mathbf{a}_{\text{HP}}(t)\right)\right], \tag{18}$$

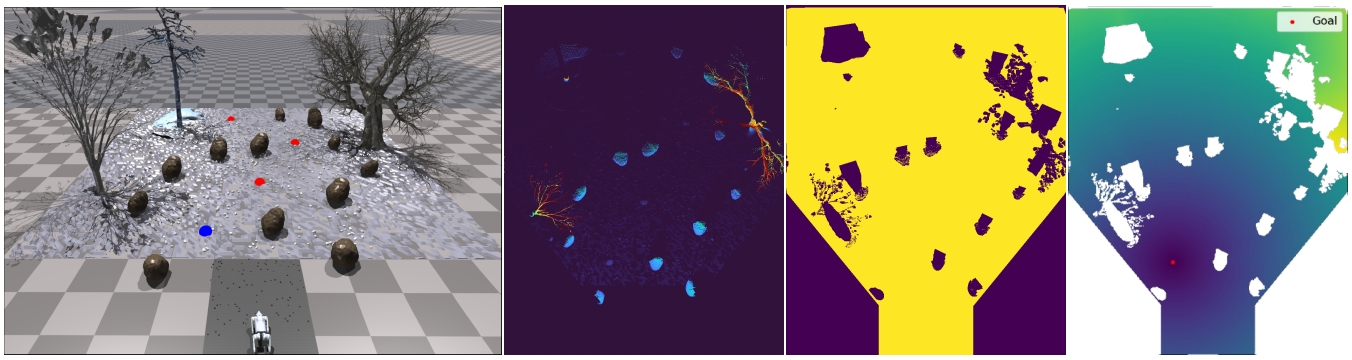
where $\mathbb{L}$ denotes the self-learning space defined in eq. (3), $\mathcal{R}(\cdot)$ is the reward function that maps the state-action pairs to real-value rewards, $\gamma\in[0,1]$ is the discount factor, balancing the immediate and future rewards. The expected return (18) and action policy $\pi(\cdot)$ are parameterized by the critic and actor networks, respectively. The reward function of HP-Student is designed to enhance task-oriented performance in robots through runtime learning. Examples include minimizing travel time and power consumption in navigation and ensuring accurate state tracking in locomotion.

Teaching-to-Learn. HP-Student controls the robot in normal situation. If its action $\mathbf{a}_{\text{HP}}(t)$ is unsafe by definition 1, HA-Teacher intervenes to ensure safe locomotion and generates a series of experience tuples over the teaching horizon τ:

$$\mathrm{E}_\tau=\{\mathbf{a}_{\text{HA}}(t),\ \mathbf{o}(t),\ \mathbf{o}(t+1),\mathcal{R}\left(\mathbf{o}(t),\mathbf{a}_{\text{HA}}(t)\right)\}_{t=k}^{k+\tau} \tag{19}$$

which are continuously stored in HP-Student's replay buffer and uniformly sampled for safety learning.

Remark. *We note the action policy designed by the HA-Teacher in theorem 1, is inherently safety-assured. The group of tuples* E_τ *in eq.* (19) *documents the HA-Teacher's successful experiences in controlling a robot from the safety boundary to a secured self-learning space. Consequently, the HP-Student will learn from these HA-Teacher's experience, specifically on how to safely manage the robot at the safety boundary. This approach enables the HP-Student to develop a safe and high-performance action policy suitable for real-time operations in dynamics wild environments.*



(a) Dynamic wild environments in Isaac Gym (b) Raw BEV Map (c) Occupancy Map (d) Cost Map

Fig. 3: Fig (a) is an overview of the dynamic wild environments: transiting from the flat terrain to unstructured and uneven ground. The quadruped robot navigates to the destination following the waypoints with minimum costs through runtime learning. The next waypoint is highlighted in blue, and the remaining waypoints are marked in red. Figs. (b), (c) and (d) illustrate the cost map generation pipeline using the point cloud data.

IV. EVALUATION

We utilize Nvidia Isaac Gym [22] and the Unitree Go2 robot to evaluate our proposed runtime learning framework. In Isaac Gym, we create dynamic wild environments, which include various natural elements, such as unstructured terrain, movable stones, and obstacles like trees and large rocks. Additionally, we arrange multiple waypoints at reasonable intervals across the terrain to simulate real-world tasks for the robot, such as outdoor exploration and search-and-rescue operations. The aim is to guide the robot to its destination by sequentially following these waypoints while minimizing traversal costs and adhering to safety constraints. The established wild environments can be found in fig. 3 (a).

A. HP-Student: Task-Oriented Reward

The reward for the DRL agent (i.e., our HP-Student) primarily focuses on three key aspects: robot safety and stability, travel time, and energy efficiency.

1) Stability: Stability is essential for the robot's safe locomotion control. Thus we consider a Lyapunov-like reward in [23], [24], which incorporates both safety and stability:

$$r_{stability} = \mathbf{e}^\top(t) \cdot \mathbf{P} \cdot \mathbf{e}(t) - \mathbf{e}^\top(t+1) \cdot \mathbf{P} \cdot \mathbf{e}(t+1), \quad (20)$$

The computation of $\mathbf{P}$ can follow the guidance in [24].

2) Travel Cost: The Euclidean distance between the robot and the waypoint is defined as:

$$d(t) = \|\hat{\mathbf{x}}_b(t) - \hat{\mathbf{p}}_{wp}\|_2 \quad (21)$$

where $\hat{\mathbf{p}}_{wp}$ is the location of next waypoint, and $\hat{\mathbf{x}}_b$ is the position of the robot base in the world frame. Inspired by [25] and [26], the navigation reward is defined as:

$$r_{nav}(t) = c_1 \cdot r_{dis}(t) + \ c_2 \cdot r_{wp} + r_{obs} \quad (22)$$

where $r_{dis}(t) = d(t) - d(t-1)$ is the reward for forwarding the waypoint. $r_{wp} = e^{-\lambda \cdot T_{reach}}$ rewards the robot as it reaches the waypoint. $\lambda \in (0,1)$ is a time decay factor and T_{reach} denotes the time step when the waypoint is reached. r_{obs} serves as a penalty when the quadruped collides with the obstacles. c_1, c_2 and c_3 are used hyperparamters.

3) Energy Consumption: Power efficiency remains a major challenge for robots in outdoor settings. We model the motor as a non-regenerative braking system [27]:

$$p_{motor} = \max\{ \underbrace{\tau_m \cdot \omega_m}_{output\ power} + \underbrace{L_{copper} \cdot \tau_m^2}_{heat\ dissipation}, \ 0\} \quad (23)$$

where τ_m and ω_m are motor's torque and angular velocity respectively. L_{copper} is copper loss coefficient. Taking c_m as a hyperparameter, we define the energy consumption reward:

$$r_{energy} = -c_m \cdot p_{motor} \quad (24)$$

The ultimate reward function, designed to guide the HP-Student in learning a safe and high-performance policy as defined in eq. (18), is given by integrating the components from above eqs. (20), (22) and (24), i.e., $\mathcal{R}(\mathbf{e}(t), \mathbf{a}_{\text{drl}}(t)) = r_{stability} + r_{nav} + r_{energy} + \widehat{c} \cdot r_{aux}$, where r_{aux} denotes the auxiliary reward for tracking velocity and orientation, with a small coefficient $\widehat{c}$ to promote smooth locomotion.

HP-Student's exteroceptive observation includes the Euclidean distance d_{wp} between robot and next waypoint, and the robot's heading angle deviation to the waypoint ψ_{wp}. Its priproceptive tracking error is shared with HA-Teacher.

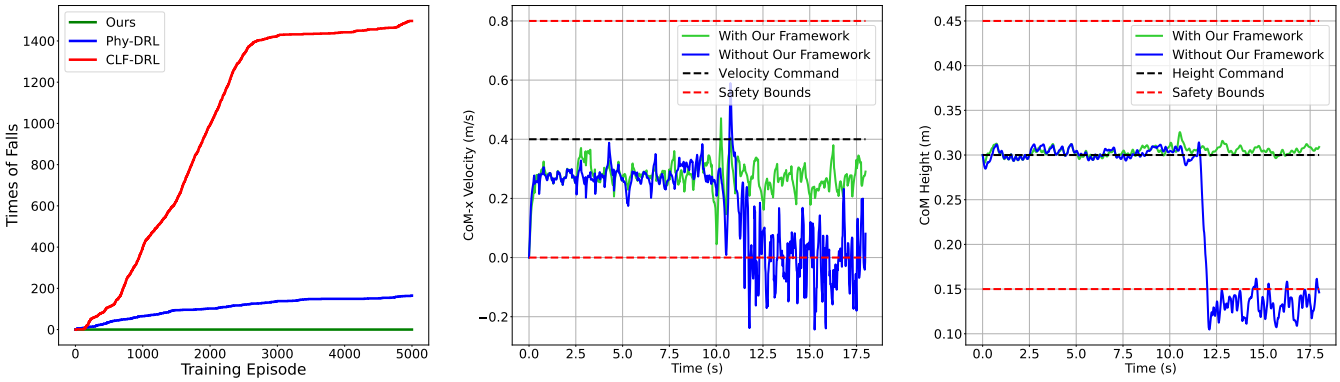
B. HA-Teacher: Safety Critical Design

The real-time physics-model knowledge in eq. (9) – used for HA-Teacher design in theorem 1 – is obtained by considering the robot's dynamics as described in [17]:

$$\mathbf{B}(\mathbf{s}(k)) = \begin{bmatrix} \mathbf{O}_3 & \mathbf{O}_3 & \mathbf{O}_3 & \mathbf{O}_3 \\ \mathbf{O}_3 & \mathbf{O}_3 & \mathbf{O}_3 & \mathbf{O}_3 \\ \mathbf{O}_3 & \mathbf{O}_3 & T \cdot \mathbf{I}_3 & \mathbf{O}_3 \\ \mathbf{O}_3 & \mathbf{O}_3 & \mathbf{O}_3 & T \cdot \mathbf{I}_3 \end{bmatrix},$$

$$\mathbf{A}(\mathbf{s}(k)) = \begin{bmatrix} 1 & \mathbf{O}_{1\times 5} & T & \mathbf{O}_{1\times 3} \\ \mathbf{O}_3 & \mathbf{I}_3 & \mathbf{O}_3 & T \cdot \mathbf{R}(\phi(k), \theta(k), \psi(k)) \\ \mathbf{O}_3 & \mathbf{O}_3 & \mathbf{I}_3 & \mathbf{O}_3 \\ \mathbf{O}_3 & \mathbf{O}_3 & \mathbf{O}_3 & \mathbf{I}_3 \end{bmatrix},$$

where $\mathbf{R}(\phi(k), \theta(k), \psi(k))$ is the real-time rotation matrix and T is the sampling period of the robot system.



(a) Accumulated number of robot's falls (b) Trajectory of the robot's CoM velocity (c) Trajectory of the robot's CoM height

Fig. 4: (a) is the accumulated number of robot falls over learning episodes. (b) and (c) depict the robot's CoM trajectories, highlighting the effectiveness of our framework in ensuring runtime learning safety under dynamic wild environments.

C. Experimental Results

1) Safety Assessment: We set quadruped's desired state to be: $v_x^d = 0.4$ m/s, $h_d = 0.3$ m, $\omega_z^d = \omega_z^{ref}$, where ω_z^{ref} is the reference angular velocity trajectory generated by the planner. Following the definition in eq. (1), we certify the robot's safety set:

$$\mathbb{S} = \{\mathbf{e} \in \mathbb{R}^{10} \mid |e_{v_x}| \leq 0.4 \text{ m/s}, |e_h| \leq 0.15 \text{ m}\}, \quad (25)$$

where $e_{v_x} \triangleq v_x - v_x^d$ denotes the error between the robot's forwarding velocity and its command, and $e_h = h - h_d$ is the error between the robot's CoM height and its desired height. The corresponding action space in eq. (2) is:

$$\mathbb{A} = \{\mathbf{a} \in \mathbb{R}^6 \mid |\mathrm{a}_v| \leq 10 \text{ m/s}^2, |\mathrm{a}_\omega| \leq 20 \text{ rad/s}^2\}, \quad (26)$$

where $\mathrm{a} = [\mathrm{a}_v,\ \mathrm{a}_\omega]^\top$, with $\mathrm{a}_v \in \mathbb{R}^3$ and $\mathrm{a}_\omega \in \mathbb{R}^3$ denoting the robot's linear and angular acceleration, respectively.

To establish the self-learning space $\mathbb{L}$ in eq. (3), we choose $\eta = 0.7$. And ω_z^{ref} is clipped within a range to ensure smooth angular velocity during navigation: $\omega_z^{ref} \in [-0.7, 0.7]$ rad/s. For HA-Teacher, we select $\alpha = 0.8$, $\kappa = 0.01$, $\chi = 0.15$, and $\tau = 10$ satisfying condition in theorem 2. By utilizing the CVX toolbox [19], we can compute $\mathbf{F}_k$ from eqs. (13) to (17) in theorem 1 for HA-Teacher's action policy (10).

The experiment also includes comparisons with the state-of-the-art safe DRL frameworks: CLF-DRL [23] and Phy-DRL [24]. We assess the safety assurance of our framework, CLF-DRL, Phy-DRL, and sole HA-Teacher, by analyzing the number of falls during runtime learning, as summarized in table I and fig. 4 (a). A demonstration video is available at: https://youtube.com/shorts/2IsZQYwjccg

The demonstration video, in conjunction with table I and training reward in fig. 5, highlights the effectiveness of our approach in ensuring runtime safety for the quadruped robot during navigation tasks in the dynamic wild environments. Additionally, the corresponding trajectories of the robot, depicted in fig. 4, further illustrate the capability of our framework in maintaining the robot states within the safety set, as outlined in eq. (26), throughout runtime learning.

2) Performance Assessment: We assess the efficiency of our framework by comparing it against different DRL agents, i.e., Phy-DRL and CLF-DRL, and sole HA-Teacher, as summarized in table I. The episodic returns are shown in fig. 5 and models are selected after training 1500, 3000, and 4500 episodes for running the task. For navigation performance, we measure *Success*–whether the robot reaches the destination, *Is Fall*–whether the robot falls, *Collision*–whether the robot collides with obstacles, *Num (wp)*–number of waypoints the robot followed and *Travel Time*. For energy efficiency, we evaluate *Avg Power*–average motor power and *Energy Consumption*–total energy used.

In table I, CLF-DRL agent struggles to develop an optimal policy for navigation task. Benefiting from its inherent structure, Phy-DRL demonstrates improved performance but still faces slow convergence and safety issues during runtime learning. HA-Teacher ensures safe navigation, but it is inefficient in travel time and energy consumption. In contrast, under the same conditions, our framework enables the HP-Student to effectively adapt to the wild environments, achieving a safe and high-performance policy after runtime learning. The demonstration video is available at: https://youtube.com/shorts/epw7sSYIiqs.

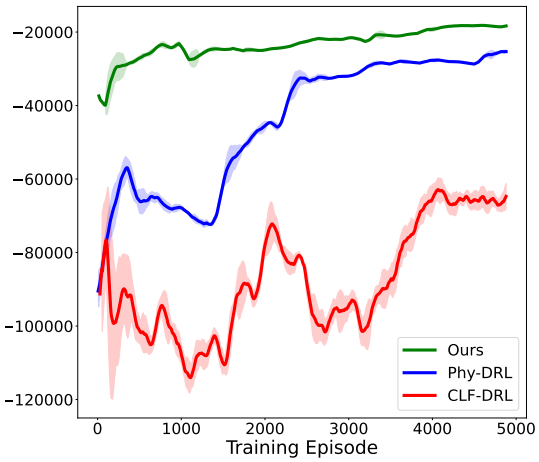


Fig. 5: Episodic Returns During Runtime Learning

Methodologies	Model – ID	Navigation Performance					Energy Efficiency	
		Success	Is Fall	Collision	Num (wp)	Travel Time (s)	Avg Power (W)	Total Energy (J)
CLF-DRL	clfdrl-ep-1500	No	Yes	Yes	0	N/A	N/A	N/A
	clfdrl-ep-3000	No	Yes	**No**	0	N/A	N/A	N/A
	clfdrl-ep-4500	No	Yes	**No**	1	N/A	N/A	N/A
Phy-DRL	phydrl-ep-1500	No	Yes	**No**	0	N/A	N/A	N/A
	phydrl-ep-3000	No	**No**	Yes	2	∞	504.3827	∞
	phydrl-ep-4500	**Yes**	**No**	**No**	**4**	56.6316	489.5142	27721.97
Ours	rtl-ep-1500	No	**No**	Yes	2	∞	491.7283	∞
	rtl-ep-3000	**Yes**	**No**	**No**	**4**	48.6417	**488.5232**	23762.59
	rtl-ep-4500	**Yes**	**No**	**No**	**4**	**45.3792**	490.9204	**22277.53**
HA-Teacher	–	**Yes**	**No**	**No**	**4**	59.2706	493.8499	29270.78

TABLE I: Comparison of Different DRL Agents, the Sole HA-Teacher, and Our Runtime Learning Framework. Safety-related metrics are highlighted in red, while key navigation metrics are marked in blue. In instances where the robot falls, other task-related metrics are labeled as N/A due to the safety-critical nature of the robot. For collision scenarios, travel time and energy consumption are represented as ∞, with power reflecting the average energy consumption prior to the collision.

V. Conclusions and Future Work

This paper presents a runtime learning framework that enables a quadruped robot to learn adaptively and safely in complex wild environments. The framework offers verifiable safety guarantees while enhancing learning performance during runtime adaptation. Experiments conducted with a Unitree Go2 robot in Isaac Gym have demonstrated the effectiveness of this runtime learning approach.

Moving forward, we will implement the proposed runtime learning framework on the real Unitree Go2 robot, enabling robot to safely learn in real-time physical environments.

References

[1] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science Robotics*, 5(47), October 2020.

[2] Ananye Agarwal, Ashish Kumar, Jitendra Malik, and Deepak Pathak. Legged locomotion in challenging terrains using egocentric vision, 2022.

[3] Peter Fankhauser, Marko Bjelonic, C. Dario Bellicoso, Takahiro Miki, and Marco Hutter. Robust rough-terrain locomotion with a quadrupedal robot. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5761–5768, 2018.

[4] Christyan Cruz Ulloa, Jaime del Cerro, and Antonio Barrientos. Mixed-reality for quadruped-robotic guidance in sar tasks. *Journal of Computational Design and Engineering*, 10(4):1479–1489, 06 2023.

[5] Jie Tan, Tingnan Zhang, Erwin Coumans, Atil Iscen, Yunfei Bai, Danijar Hafner, Steven Bohez, and Vincent Vanhoucke. Sim-to-real: Learning agile locomotion for quadruped robots, 2018.

[6] Gabriel Margolis, Ge Yang, Kartik Paigwar, Tao Chen, and Pulkit Agrawal. Rapid locomotion via reinforcement learning. In *Robotics: Science and Systems*, 2022.

[7] Yuxiang Yang, Ken Caluwaerts, Atil Iscen, Tingnan Zhang, Jie Tan, and Vikas Sindhwani. Data efficient reinforcement learning for legged robots. In *Conference on Robot Learning*, pages 1–10. PMLR, 2020.

[8] Cassandra Smith. State police urge caution as freezing rain leads to several crashes. *WCIA*.

[9] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. RMA: Rapid motor adaptation for legged robots. In *Robotics: Science and Systems*, 2021.

[10] Laura Smith, J. Chase Kew, Xue Bin Peng, Sehoon Ha, Jie Tan, and Sergey Levine. Legged robots that keep on learning: Fine-tuning locomotion policies in the real world. In *2022 International Conference on Robotics and Automation (ICRA)*, 2022.

[11] Fereshteh Sadeghi and Sergey Levine. Cad2rl: Real single-image flight without a single real image, 2017.

[12] Dung T Phan, Radu Grosu, Nils Jansen, Nicola Paoletti, Scott A Smolka, and Scott D Stoller. Neural Simplex architecture. In *NASA Formal Methods Symposium*, pages 97–114. Springer, 2020.

[13] Guillaume Brat and Ganeshmadhav Pai. Runtime assurance of aeronautical products: preliminary recommendations. *NTRS - NASA Technical Reports Server*, 2023.

[14] Joseph Sifakis and David Harel. Trustworthy autonomous system development. *ACM Transactions on Embedded Computing Systems*, 22(3):1–24, 2023.

[15] Shengduo Chen, Yaowei Sun, Dachuan Li, Qiang Wang, Qi Hao, and Joseph Sifakis. Runtime safety assurance for learning-enabled control of autonomous driving vehicles. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 8978–8984. IEEE, 2022.

[16] Zipeng Fu, Ashish Kumar, Ananye Agarwal, Haozhi Qi, Jitendra Malik, and Deepak Pathak. Coupling vision and proprioception for navigation of legged robots, 2022.

[17] Jared Di Carlo, Patrick M Wensing, Benjamin Katz, Gerardo Bledt, and Sangbae Kim. Dynamic locomotion in the mit cheetah 3 through convex model-predictive control. In *2018 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2018.

[18] Yanbing Mao, Yihao Cai, and Lui Sha. Real-DRL: Teach and learn in reality. Available at: `https://www.dropbox.com/scl/fi/bltmsj98r7ddqjlqof3r6/IROS-2025.pdf?rlkey=m15c19d4fmagm8ljy50zihnku&st=9e7x4ajc&dl=0`.

[19] Michael Grant, Stephen Boyd, and Yinyu Ye. Cvx users' guide. *online: http://www. stanford. edu/boyd/software. html*, 2009.

[20] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *4th International Conference on Learning Representations, ICLR*, 2016.

[21] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

[22] NVIDIA. Isaac gym - preview release, 2023. Available at: `https://developer.nvidia.com/isaac-gym`.

[23] Tyler Westenbroek, Fernando Castaneda, Ayush Agrawal, Shankar Sastry, and Koushil Sreenath. Lyapunov design for robust and efficient robotic reinforcement learning. *arXiv:2208.06721*, 2022.

[24] Hongpeng Cao, Yanbing Mao, Lui Sha, and Marco Caccamo. Physics-regulated deep reinforcement learning: Invariant embeddings. In *The Twelfth International Conference on Learning Representations*.

[25] Yuanzhe Geng, Erwu Liu, Rui Wang, and Yiming Liu. Deep reinforcement learning based dynamic route planning for minimizing travel time, 2020.

[26] Robert Penicka, Yunlong Song, Elia Kaufmann, and Davide Scaramuzza. Learning minimum-time flight in cluttered environments. *IEEE Robotics and Automation Letters*, 7(3):7209–7216, 2022.

[27] Yuxiang Yang, Tingnan Zhang, Erwin Coumans, Jie Tan, and Byron Boots. Fast and efficient locomotion via learned gait transitions. In *Conference on Robot Learning*, pages 773–783. PMLR, 2022.