FT-Net: Learning Failure Recovery and Fault-tolerant Locomotion For Quadruped Robots

Zeren Luo*, Erdong Xiao*, Peng Lu†

***Abstract*— Quadruped robots, in recent years, have been increasingly used in extremely harsh and dangerous conditions. Consequently, diverse severe hardware failures may occur at any time during the working cycle of the robots. In this work, we propose a fault-tolerant (FT) control pipeline based on model-free reinforcement learning – FT-Net, which is guided by a variable-height inverted pendulum model and support polygon. This pipeline allows the robot to dynamically and autonomously adapt to both partial and complete motor failures. Unlike conventional FT control methods that need to pinpoint the failed location, our controller identifies the fault implicitly with a neural network-based adaptor. Furthermore, we achieve a unified policy that is capable of switching from four-legged to three-legged walking mode when complete motor failure occurs. It is shown by both extensive simulation and hardware experiments that FT-Net learns to effectively perform recovery behaviors. The fault-tolerant locomotion can even be executed in various dynamic tasks and terrains.**

***Index Terms*— Quadruped robots, fault-tolerant control, deep reinforcement learning, self-adaptation**

MULTIMEDIA MATERIAL

Video Demo: https://youtu.be/eqfCWrgKJSg

Supplementary Material: https://github.com/arclab-hku/FT-Net_Quadruped_Robots

I. INTRODUCTION

Due to their high agility and strong stability, quadrupedal robots nowadays has been widely used for a wide variety of applications, particularly outdoor exploration. Analogous to humans and animals might easily get injured in the wild, legged robots may also encounter a variety of unexpected hardware failures during their outdoor operation, such as electric circuit aging, power shortage, and motor malfunctioning. Handling hardware failure in the real world is a crucial element for locomotion and other dynamic tasks.

However, accomplishing these tasks under motor failure presents significant challenges for quadrupedal robots. Motor failures can directly impact the robot's ability to generate sufficient torque for balancing or locomotion, leading to falling and compromised walking performance due to the altered dynamics. Maintaining balance or walking stability, and adapting the gait in real-time to compensate for the loss of motor effectiveness require sophisticated control strategies [1], [2]. Specifically, quadrupedal robots need to detect and diagnose failures promptly, estimate their impact on the system, and make rapid decisions to adjust control signals and redistribute forces among the remaining legs.

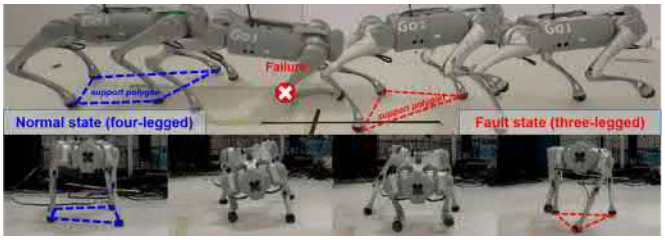

Fig. 1: The evolution of leg failure task: the failure occurs on the front right (FR) calf joint. Our proposed controller empowers the robot to dynamically and rapidly recover from the failure, and continue to walk under the fault state with optimal support polygon.

Current research in this area is still very few and conventional approaches focus much on explicit fault detection and isolation. The existing model-based approaches require very complex modeling [3], isolation algorithms [4] and real-time coordination mechanisms [5] to accommodate the varying dynamics of these tasks. Meanwhile, the learning-based approaches can simply address the locking joint failure [6], [7] (motor is immobile) or the partial motor failure [8] (motor's effectiveness is reduced). These works often make idealistic assumptions regarding failure cases and the validation on the real robot is rare.

In contrast, FT-Net aims to handle 1) the motor failure occurring in an arbitrary state, i.e. the failure is assumed to happen at any joint position and any period of the gait schedule; 2) complete motor failure rather than partial motor failure; 3) autonomous detection of the failed location. The main contributions of this work can be summarized as:

- A fault-tolerant control framework for the quadruped robots using deep reinforcement learning – **FT-Net**, which empowers the robot to handle complete failures. The adaptor learns to implicitly infer about the motor faults and avoid any extra high-cost diagnosis procedures in conventional model-based methods.
- The training process enhances the dynamic balancing properties using heuristic reward terms, which contain designs related to robot dynamics - VHIP (Variable-height Inverted Pendulum) [9] model and support polygon. Guided by these terms, the FT-Net solves beyond the sim-to-real problem in the aspect of adapting larger perturbations and ever-changing physical parameters.

* Equal contribution.

Manuscript received 15 June 2023; Accepted 25 October 2023. This letter was recommended for publication by Editor-in-Chief Asfour Tamim and Editor Desai Jaydev P. upon evaluation of the reviewers' comments. This research is supported by General Research Fund under Grant No. 17204222, and in part by the Seed Funding for Collaborative Research and General Funding Scheme-HKU-TCL Joint Research Center for Artificial Intelligence. (Corresponding author: Peng Lu. lupeng@hku.hk)

The authors are all with the Adaptive Robotic Controls Lab (ArcLab), Department of Mechanical Engineering, The University of Hong Kong, Hong Kong SAR, China. zerluo, exiao@connect.hku.hk

- The framework trains a unified control policy taking the normal locomotion, fault recovery, and fault locomotion scenarios into account. The controller has the capability to automatically switch between a quadrupedal gait pattern to a tripedal gait pattern.
- Our controller allows the robot to generalize to a variety of terrains with severe motor failures. Moreover, both the hardware and simulation experimental validation of our method are included.

II. RELATED WORK

A. Conventional Methods for Fault-tolerant Control

In order to handle the malfunctions, the robotics community commonly resorts to system identification [3], [4]. These methods are also characterized by fault detection relying on the system response that deviates significantly from the nominal state. However, system identification may involve response delays in the framework, and it is necessary to integrate additional controllers to handle the diagnosis result of damages [1], [10]. The isolated failure model such as the fault-tolerant foot trajectory and gait planning is also required to be analytically established [2], [11]. Nevertheless, it is very laborious to obtain the machine's characteristics under a wide range of failures, and deviation in these parameters will result in additional uncertainties.

In addition to enhancing the efficiency of fault estimation [12], an alternative line of conventional research tries to eliminate the demanding diagnostic procedure by improving fault tolerance and robustness of the control algorithms [3]. However, as the range of fault situations in their models grows, the modeling complexity dimensions of model uncertainty will also increase significantly, which demands higher computational budgets. Moreover, failures widen the gap between analytical models and reality, diminishing controller performance.

The occurrence of joint failure is normally accompanied by falling, or severe change of the body pose. The failure recovery procedure is necessary while the aforementioned work does not involve the special solution to this issue. A small amount of work on optimization-based methods accomplishes the recovery behavior by executing pre-defined motion sequences [13], [14], which require much effort to design the route.

B. Learning-based Methods for Fault-tolerant Control

Pioneering works on data-driven methods circumvent the complex modeling of failure uncertainty [15], [16]. The later thrive of deep reinforcement learning (Deep-RL) also aids in generating robust control policies in challenging scenarios. Prior works adopt the online [17]–[19] and offline [20] adaptation pipelines, which leverages real-world data to empower the robot adaptability in novel situations. In the hardware failure case, in contrast, system characteristics vary much more drastically than the normal sim-to-real gap, which influences the robot's behavior more significantly.

Existing research on RL-based fault-tolerant control primarily concentrates on addressing either partially weakened motors [8], [21] or locking joint failure [6], [7]. They only consider the circumstance in which the locked joint position is known, while it is far from the reality that a failure location and triggered time are agnostic to the RL agent. Some other joint failures consider less realistic scenarios, such as limb ripped-out [7], shortened limb [22]. On the other hand, some of the works are only validated in simulation regardless of the reality gap with the real-world performance remains unclear [7], [8]. RL controllers also contribute to recovery behavior, while the recovery is normally triggered under a relatively static condition with fully powered motors [23]–[25] or is assisted by the attached robot arm [26].

III. FT-NET PIPELINE

Our proposed FT-Net framework is composed of three components (shown in Fig. 2) – Section III-A: Learn to locomote: A conditioned policy on the latent representation of fault scenarios, Section III-B: Joint failure mechanism in the training environment, Section III-C: Learn to adapt: leveraging observation history to train an adaptor that implicitly addresses the failure.

A. Heuristic Fault-tolerant Learning

1) Policy Learning in Faulty Environments: A universal policy is learned to walk accounting for a wide range of faults. We generate a large number of joint failure circumstances ranging from partial to complete failure and the simulation is forwarded in these environments. The agent becomes generalizable to different failures by making use of the experience in these environments.

The problem is formulated as a Markov Decision Process (MDP), which is a framework to model a discrete-time stochastic control process. An MDP $\mathcal{M}$ is defined by the tuple $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{P}, p_0\}$ with $\mathcal{S}$ being state space, $\mathcal{A}$ being action space and p_0 being the distribution of inital state. The reward function $\mathcal{R}$ is a mapping $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$. Transition probabilities is denoted as $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$. A stationary policy is defined as a mapping from the state to the distribution over action, i.e. $\pi : \mathcal{S} \to P(\mathcal{A})$. The learning agent interacts with the environment with the action sampled from this distribution, i.e. $\boldsymbol{a}_t \sim \boldsymbol{\pi}(\boldsymbol{a}_t|\boldsymbol{s}_t)$ and receives rewards $\mathcal{R}(s_t, a_t)$ from the environment. RL problem aims to figure out the optimal policy π^* that maximize the accumulated rewards of this MDP $\mathcal{M}$ with a discount ratio γ, i.e.:

$$J_{\mathcal{M}}(\boldsymbol{\pi}) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{T-1}(\gamma^t \mathcal{R}(s_t, a_t))\right]. \tag{1}$$

State Space and Observation The robot state $\boldsymbol{s}_t$ is the combination of the observation $\boldsymbol{o}_t$ and the environments parameterization which will be later encoded by encoder $\boldsymbol{E}_{\boldsymbol{\theta}}(\cdot)$, which can be written as $\boldsymbol{s}_t = [\boldsymbol{o}_t, \boldsymbol{e}_t^{\boldsymbol{\theta}}] = [\boldsymbol{o}_t, \boldsymbol{E}_{\boldsymbol{\theta}}(\boldsymbol{\mu})]$.

The observation $\boldsymbol{o}_t \in \mathbb{R}^{49}$ is the measurements from all kinds of sensors consisting of base angular velocity $\boldsymbol{\omega}_t \in \mathbb{R}^3$, projected gravity $\boldsymbol{g}_t \in \mathbb{R}^3$, command velocities $\boldsymbol{v}_t^{cmd} \in \mathbb{R}^3$, joint positions $\boldsymbol{q}_t \in \mathbb{R}^{12}$, joint velocities $\dot{\boldsymbol{q}}_t \in \mathbb{R}^{12}$, last actions $\boldsymbol{a}_{t-1} \in \mathbb{R}^{12}$, and the foot contact boolean $\boldsymbol{c}_t \in \mathbb{R}^4$ indicated by the

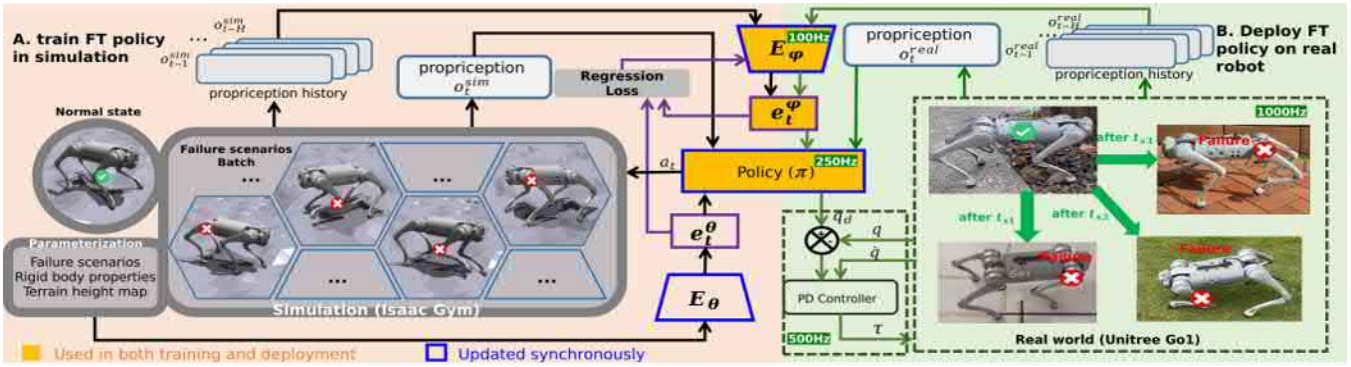

Fig. 2: Online adaptive framework of **FT-Net**. Section III-A. Learning the locomotion policy with PPO algorithm [27]. The physical parameter encoder $\boldsymbol{E}_\theta$, adaptor $\boldsymbol{E}_\varphi$, and policy networks $\boldsymbol{\pi}$ are updated synchornously. Section III-C. The online adaptor $\boldsymbol{E}_\varphi$ implicitly identifies the fault. Section IV. Multi-threading real hardware deployment, where we forward the adaptor and policy network in different frequencies.

contact force. $\boldsymbol{E}_\theta(\cdot)$ maps and reduces the high-dimensional physical information $\boldsymbol{\mu}$ to a low-dimensional vector $\boldsymbol{e}_t^\theta$. The input of $\boldsymbol{E}_\theta(\cdot)$ is the parameterization of the environments batches, consisting of the vector of motor strength $\in \mathbb{R}^{12}$, rigid body properties (COM, bodies mass, frictions) $\in \mathbb{R}^{12}$, and terrain height map $\in \mathbb{R}^{187}$.

Action Space The actions $\boldsymbol{a}_t \in \mathbb{R}^{12}$ outputted by the control policy are the target positions $\boldsymbol{q}_d$ of all joints. This command would be executed by the low-level controller $\boldsymbol{\tau} = \boldsymbol{K}_p(\boldsymbol{q}_d - \boldsymbol{q}) + \boldsymbol{K}_d(\dot{\boldsymbol{q}}_d - \dot{\boldsymbol{q}})$ both in the simulation and the hardware, in which we can specify the $\boldsymbol{K}_p$ and $\boldsymbol{K}_d$ and the target joint velocity $\dot{\boldsymbol{q}}_d$ is set as $\mathbf{0}$.

Reward Function The detailed expression of each term is shown in Table I. The heuristic reward weights tuning ranges are initially calculated based on their proportions to the major tracking terms, and the final values are determined based on experimental trials. The total reward at given state and action $\boldsymbol{s}_t$, $\boldsymbol{a}_t$, $\mathcal{R}_t(\boldsymbol{s}_t, \boldsymbol{a}_t)$ can be obtained as $\mathcal{R}_t(\boldsymbol{s}_t, \boldsymbol{a}_t) = \sum_i (w_i r_i)$.

2) Heuristics Terms Enhancing Dynamic Equilibrium: It is important to note that to facilitate the robot's dynamic equilibrium ability, especially in the failure case, we design some heuristic reward terms (marked as red in Table I).

VHIP [9] is utilized as a low-dimensional and effective representation of the dynamic motion. The VHIP depicts a quadruped robot as a concentrated point mass located at the center of mass (COM) within the system. This point mass is connected to a telescoping rod with no mass, which makes contact with the surface. The contact point of the rod aligns precisely with the robot's center of pressure (COP)[1]. In essence, it represents the specific location where the resultant ground reaction force vector $\boldsymbol{f}$ applies. For a visual depiction of the VHIP, please refer to Fig. 3. COP position falls into the convex hull of the contact position $\boldsymbol{C_i} \in \mathbb{R}^3$:

$$\boldsymbol{p}_{COP} = \sum_{c_i \in \mathbb{C}} \frac{\boldsymbol{f}_i \cdot \boldsymbol{n}}{\sum_i (\boldsymbol{f}_i \cdot \boldsymbol{n})} \boldsymbol{C_i} \tag{2}$$

According to the model, the balancing heuristic terms are designed in the following form:

[1]The comparison with Zero Moment Point (ZMP)-based VHIP is presented in the supplementary, which justifies the use of COP-based VHIP to formulate our heuristic guidance.

- VHIP angle: The angle θ between the z axis and COM–COP connection segment. This value could be calculated by the norm of the COM–COP vector itself and its projection on the z-axis.

$$\theta = \arccos \frac{\|\boldsymbol{l}_z\|}{\|\boldsymbol{l}\|} = \arccos \frac{\|\boldsymbol{p}_{COM,z}\|}{\|\boldsymbol{p}_{COM} - \boldsymbol{p}_{COP}\|} \tag{3}$$

- VHIP acceleration: When the quadruped robot has the tendency to fall, the angular acceleration of θ abruptly increases. Therefore, in addition to circumventing large θ angle, we involve the penalization of $\ddot{\theta}$ as well. By ignoring the base linear acceleration which is much smaller than the acceleration of gravity, it is trivial to have the equation of motion for VHIP

$$\ddot{\theta} = -\frac{g}{\|\boldsymbol{l}\|} \sin\theta = -\frac{g}{\|\boldsymbol{p}_{COM} - \boldsymbol{p}_{COP}\|} \sin\theta \tag{4}$$

- Support polygon – $\boldsymbol{COM}^{proj}$ distance: When the failure occurs, the ground projection of the COM ($\boldsymbol{COM}^{proj}$) exhibits large oscillations. If they are close to the edge of the support polygon of the robot, the robot would be at risk of transiting to the falling state [28].

$$\boldsymbol{o_i} = \boldsymbol{COM}^{proj} - \boldsymbol{C_i} \tag{5}$$

$$\boldsymbol{d_i} = \frac{\boldsymbol{S}_{\Delta \overline{\boldsymbol{COM}^{proj} \boldsymbol{C_i} \boldsymbol{C_j}}}}{\|\overrightarrow{\boldsymbol{C_i} \boldsymbol{C_j}}\|} = \frac{\boldsymbol{o_i} \times \boldsymbol{o_j}}{2\|\boldsymbol{o_i} - \boldsymbol{o_j}\|} \tag{6}$$

where $\{\boldsymbol{o_i}\}$ is the connection vectors from $\boldsymbol{COM}^{proj}$ to $\boldsymbol{C_i}$, and distance $\boldsymbol{d_i}$ is computed via the area of the triangle $\boldsymbol{S}_{\Delta \overline{\boldsymbol{COM}^{proj} \boldsymbol{C_i} \boldsymbol{C_j}}}$ (Fig. 3). We find out that penalizing the maximum distance among $\{\boldsymbol{d_i}\}$ results in a better policy than rewarding their minimum distance. Consequently, this augmented term is designed as the maximum distance from $\boldsymbol{COM}^{proj}$ to the support polygon, i.e. $\max\{\|\boldsymbol{d_i}\|\}$.

B. Joint Failure Mechanism

In this work, the model is able to account for the most common and realistic cases: partial failure (weakened motor) and complete failure (motor becomes totally not controllable regardless of the actions applied). We introduce a failure

TABLE I: Reward Function Terms. The positive and negative w_i respectively represent reward and penalization terms.

Term	Expression(r_i)	Weight(w_i)
x,y velocity tracking	$\exp\left\{-4(v_{xy}-v_{xy}^{cmd})^2\right\}$	1.0
yaw velocity tracking	$\exp\left\{-4(\omega_z-\omega_z^{cmd})^2\right\}$	0.5
z velocity	v_z	-2.0
pitch, yaw velocity	$(\omega_x+\omega_y)^2$	-0.01
action rate	$(a_t-a_{t-1})^2$	-0.02
joint acceleration	$\ddot{q}^2$	-2.5×10^{-7}
feet air time	$\sum_{i=0}^{4}(t_{air,f}-0.5)$	0.6
joint motion cosmetic	$(q-q^{init})^2$	-0.05(front) -0.2(rear)
VHIP angle	θ	-0.015
VHIP acceleration	$\ddot{\theta}^2$	-0.01
distance to support polygon	$\max\{\|\|\boldsymbol{d_i}\|\|\}$	-0.01

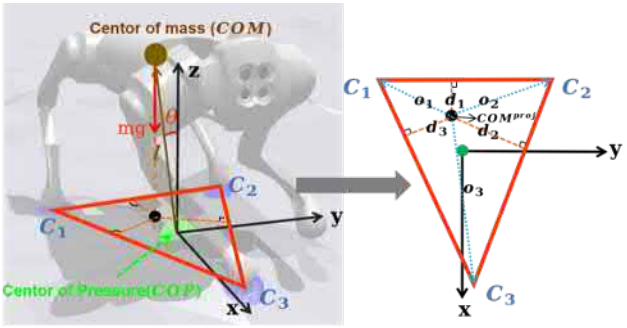

Fig. 3: A VHIP model of the robot (**Left**). The top view of the model regarding the xy plane (**Right**)

coefficient $\alpha \in [0,1]$ with 0 indicating complete failure and a larger deviation from 1 indicating more severe failure with a larger loss of effectiveness. The actual torque applied can be calculated by:

$$\boldsymbol{\tau}^{'} = \alpha\boldsymbol{\tau} = \alpha\boldsymbol{K}_p(\boldsymbol{q}_d - \boldsymbol{q}) + \alpha\boldsymbol{K}_d(\dot{\boldsymbol{q}}_d - \dot{\boldsymbol{q}}) \tag{7}$$

Thus, applying the joint failure effect is equivalent to implementing the low-level controller coefficients $\boldsymbol{K}_p^{'} = \alpha\boldsymbol{K}_p$, $\boldsymbol{K}_d^{'} = \alpha\boldsymbol{K}_d$.

We define two distributions from which we sample α: severe failure distribution $\mathcal{F}_s : \mathcal{U}(0, \boldsymbol{s}_{thres})$ and moderate failure distribution $\mathcal{F}_m : \mathcal{U}(\boldsymbol{s}_{thres}, 1)$. In our experiment setting, 0.1 is selected as the severe failure threshold $\boldsymbol{s}_{thres}$ (Go1 can hardly maintain a standing-still state when $\alpha < \boldsymbol{s}_{thres}$ in both the simulation and the real robot)

We intend to expose the learning agent more to highly severe fault scenarios to facilitate the learning of fault tolerance. Therefore, we increase the probability of encountering $\mathcal{F}_s$ in the sampling process. To achieve this, the initialization of α in each environment is determined as follows: The faulty distribution is firstly defined as a Bernoulli distribution with a probability parameter p_s, i.e. $p \sim \mathcal{B}(1, p_s)$ and then α is sampled from the following distribution:

$$\alpha \sim p\mathcal{F}_s + (1-p)\mathcal{F}_m. \tag{8}$$

The parameter p_s is introduced as a sampling weight. Its value makes the sampling probability from the high severity level $\mathcal{F}_s$ be amplified by a manually-specified factor of $p_s/s_{thres} > 1$ (we set this factor to be 3).

The failed joint J_f with coefficient α is uniformly sampled from all the 12 actuated joints, i.e. $J_f \sim \mathcal{U}\{1, 12\}$. For the benefit of sim-to-real transfer, the remaining normal motor strengths are randomized within a reasonable domain of $[0.9, 1.1]$.

In the trials, we find that the agent tends to learn an extremely aggressive movement within the simulator physics engine in the fault-tolerant task. This policy tends to fling the calf limb to the middle of the body when the calf joint is broken, while this maneuver is unsafe and hard to achieve in the real robot. A constraint $\boldsymbol{C}$ is applied regarding the joint pose limit to refrain the joint from these aggressive motions.

$$\boldsymbol{C} : q_i \in \begin{cases} [\mathcal{L}, q^{upper}] & \text{if } i \in \mathbb{C}_{\mathbb{A}} \cap s_i < \boldsymbol{s}_{thres} \\ [q^{lower}, q^{upper}] & otherwise \end{cases} \tag{9}$$

where $\mathbb{C}_{\mathbb{A}}$ denotes the aggregation of all the calf joint indices. The constraint defines that if the motor strength s_i is lower than $\boldsymbol{s}_{thres}$, the joint cannot exceed $\mathcal{L}$. $\mathcal{L}$ is specified as 2.1 after several attempts, which is the most favorable pose to initiate recovery after falling.

C. Learn to Adapt with Proprioception

Despite the controller learned in the section III-A achieves fault recovery and fault-tolerant gait, the extent and the exact location of damage are not available in a real-world deployment. A network-based adaptor is introduced that enables the agent to make the real-time estimation of the system model $\boldsymbol{e}_t^{\varphi}$. The adaptor $\boldsymbol{E}_{\varphi}(\cdot)$ is trained to approximate the function that maps the privileged information to the latent space representation. It is essentially a regression algorithm with the input feature being the N steps (we use $N = 30$) sequence of previous observations and the output label being the encoded vector $\boldsymbol{e}_t^{\theta}$. $\boldsymbol{e}_t^{\varphi}$ can be obtained as

$$\boldsymbol{e}_t^{\varphi} = \boldsymbol{E}_{\varphi}(\boldsymbol{o}_{t-1}, ..., \boldsymbol{o}_{t-N}), \tag{10}$$

It can be noted that our training procedure is distinct from [17], [18] where a teacher policy acts as the benchmark for the learning of a student policy in an offline fashion. We save this laborious procedure by optimizing the policy and the online adaptor concurrently. With the training going on,

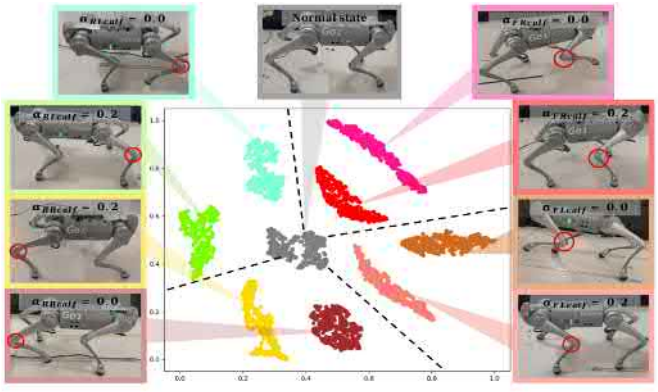

Fig. 4: $\boldsymbol{t-SNE}$ for encoded $\boldsymbol{e}_t^{\varphi}$ vector with different representations of failures. The normal state lies in the center of all situations and the failures on the other legs occupy over the surrounding area.

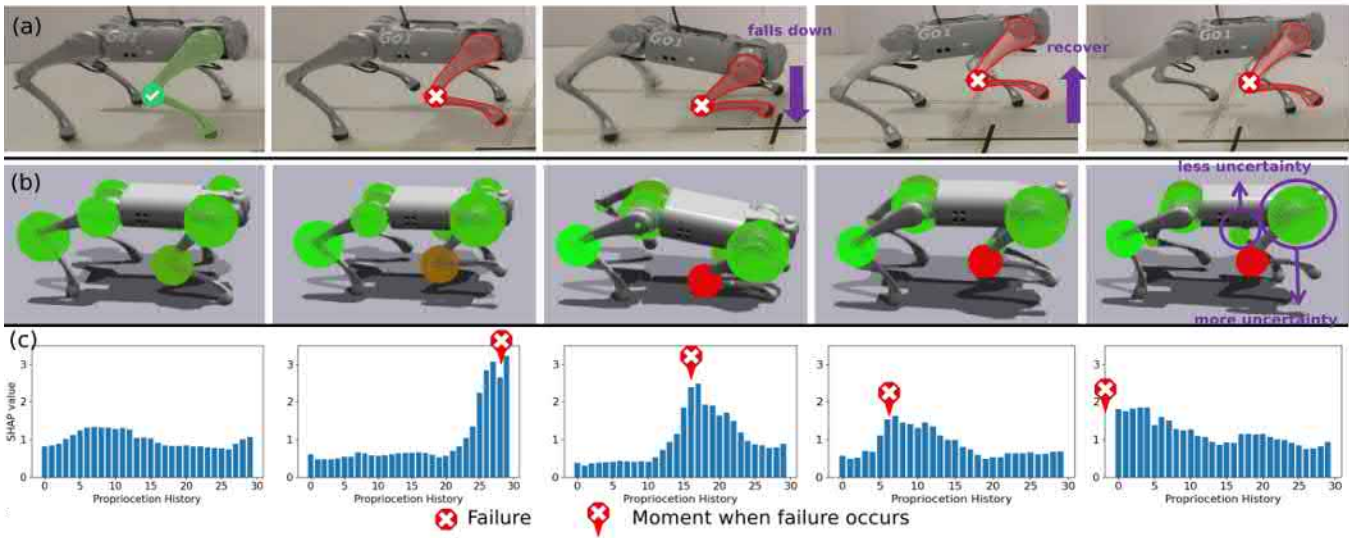

Fig. 5: (a) The failure event emerges during the motion, the failure leg is marked in a red outline. (b) The color and the radius of the ball are the mean and the standard deviation of the estimated failure coefficient α, respectively. (c) The SHAP value within a history buffer during the emergence. (* A column of pictures represents the event at the same time step)

$\boldsymbol{E}_\theta$ converges with more accurate physical representation, $\boldsymbol{E}_\varphi$ also benefits from the more reliable reference for the regression.

This training methodology also helps solve the sim-to-real problem, as $\boldsymbol{E}_\theta$ encodes ground-truth physical information, which empowers the robot to have prior knowledge regarding the physical world. During deployment, the adaptor $\boldsymbol{E}_\varphi$ estimates the real-world physical parameters. The control policy with this estimation is able to perceive the real world.

IV. EXPERIMENTAL RESULTS

The training process of FT-Net is performed on the simulator NVIDIA IsaacGym [29], [30] with the Unitree Go1 quadruped model. This simulation platform allows us to train a large number of agents in parallel. Specifically, we use 4096 Go1s for parallel training for about 2000 episodes with an episode length of 24 seconds. The random terrain is generated with the undulation $\in [-5, 5] cm$. The policy network and physical parameter encoder are both MLP with the layer sizes being [512, 256, 128, 12], [256, 128, 8], and the adaptor is a 3-layer CNN with the feature [32, 32, 9, 2], [32, 32, 5, 1]×2. We use the PPO algorithm [27] to update $\boldsymbol{\pi}$ and $\boldsymbol{E}_\theta$. The whole training process costs about 1.2 hours on one NVIDIA RTX 3080Ti Laptop GPU.

The real hardware experiments are carried out on Unitree Go1-NX quadruped with 12 actuated degrees of freedom and ˜12 kg weights. In our work, the nominal $\boldsymbol{K}_p$ is set as 30.0, 50.0, 50.0 for hip, thigh, and calf joint and $\boldsymbol{K}_d$ as 2.0 for all joints on each leg. A multi-threading deployment strategy is shown in Fig. 2, showcasing the frequency regarding the state reading from the real sensor (IMU, joint encoders, foot force sensor, etc.), the policy and the adaptor network forwards. The frequencies are determined according to the computational time of each thread, especially the forward time of two deployed networks.

A. Classification of Different Joint Failures

Two scenarios are considered to intuitively visualize the clusters and the classification of the failure latent vector $\boldsymbol{e}_t^\varphi$: (1) The complete failure (2) The partial failure with motor losing 80% of the effectiveness ($\alpha = 0.2$). We collect the data of $\boldsymbol{e}_t^\varphi$ constantly for 500 time-steps directly from the Go1 robot, then use this dataset to plot the $\boldsymbol{t-SNE}$ figure.

The $\boldsymbol{t-SNE}$ figure (Fig. 4) demonstrates the high dimensional latent vector $\boldsymbol{e}_t^\varphi$ in a 2-D graph, which illustrates that the adaptor $\boldsymbol{E}_\varphi$ allows us to differentiate diverse failure scenarios. We observe that the $\boldsymbol{e}_t^\varphi$ vectors are categorized into distinct clusters with the regions in the upper left, upper right, lower left, and lower right corresponding to different failure cases: rear left (RL), front right (FR), rear right (RR), and front left (FL), respectively. The data associated with complete and partial failures of the same leg are organized into separate clusters within the corresponding region. This is a strong indicator of the capability of the adaptor to implicitly classify the failure scenarios and it serves as the basis for a unified controller to handle different situations.

B. Attribute Analysis of the Failure Emergence

1) ***System physical variation attributes****:* FT-Net's capability to accommodate the physical variation associated with failures is examined. We evaluate the low-dimensional $\boldsymbol{e}_t^\varphi$'s capability to represent the fault scenario during the emergence of a failure event.

A network $\boldsymbol{E}_\varphi^R$ is trained to reestablish the latent information $\boldsymbol{e}_t^\varphi$ into a format compatible with the input of $\boldsymbol{E}_\theta$. The reestablish net $\boldsymbol{E}_\varphi^R$ estimates the mean $\hat{\boldsymbol{\mu}}_\theta$ and the standard deviation $\boldsymbol{\sigma}_\theta$ of the reestablished values. Given the real physical parameters $\boldsymbol{\mu}_\theta$ acquired from the simulation, it is trained by minimizing the Gaussian negative log-likelihood loss.

During fault emergence on one of the legs in Fig. 5(a)(b), the estimated mean of failure coefficient α of the failed

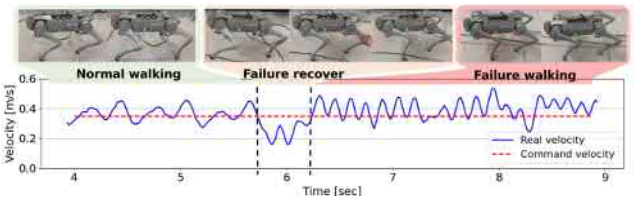

Fig. 6: Velocity tracking performance before and after the failure is triggered on the rear left (RL) leg.

joint drops progressively to 0. Gradually, the robot becomes more certain about this system's failure as the robot's state changes significantly. It is interesting to note that the robot also becomes more confident about the intact state of the joints on the other three legs as they are majorly used to support and maintain motion for the impaired robot.

Simultaneously, the uncertainties of the intact condition of other intact joints on the faulty leg increase, as they play a minor role during faulty locomotion. The above results indicate that the $\boldsymbol{E}_\varphi$ can implicitly assist the controller in perceiving the failure event.

*2) **Input contribution attributes**:* The SHAP (SHapley Additive exPlanation) [31] value of each component within the proprioception history o_i^j is calculated ($i \in \{1,...30\}$ and $j \in \{1,...,49\}$). This value $\boldsymbol{\phi}_i^j$ explains the contribution of o_i^j to the adaptor output $\boldsymbol{e}_t^\varphi$. Fig. 5(c) visualizes the sum of the SHAP values of all observation components at each time instant, i.e. $\sum_{j=1}^{D} \boldsymbol{\phi}_i^j$.

It can be seen that when the robot is in the normal state, each state in the proprioceptive history buffer contributes in a relatively uniform manner to the prediction of the $\boldsymbol{e}_t^\varphi$. With the emergence of the failure, this event causes a significant impact on $\boldsymbol{e}_t^\varphi$. The observation $\boldsymbol{o}_t$ of this pivotal event exhibits the highest SHAP value when it is pushed back to the history buffer (30 slices of history states). The controller is then prompted to adapt its behavior and make an intense recovery. In the following, the influence of emergence gradually diminishes as its main role in guiding the recovery behavior and initiating fault-tolerant walking has already been accomplished. For the theoretical details regarding the above analyses, we refer the reader to Section II in the supplementary material.

C. Recovery and Fault-tolerant Locomotion Task

We examine the robot's recovery from a static state in Fig. S11 (Fig. S* in the article refer to the figures in the supplementary materials) and also validate the performance of the controller under a dynamic context. The robot is commanded to track a constant velocity of 0.35 m/s and the leg failure is intentionally triggered during the midpoint of the trajectory.

Partial Failure ($0 < \alpha < 1$)**.** We investigate the performance when $\alpha = 0.2$, which accounts for only 20% of its original effectiveness. A slight squat is observed upon triggering the failure while the controller demonstrated a rapid adaptation process. The adjustment leads to the emergence of a novel "limping" gait schedule, characterized by a distinctive contact transition time compared to the normal state (shown in Fig. 7).

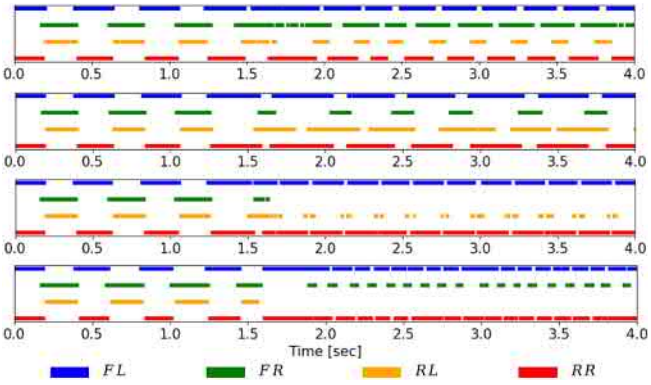

Fig. 7: Time history of the gait for four different failure cases. From top to bottom: $\alpha_{FRcalf} = 0.2$, $\alpha_{RLcalf} = 0.2$, $\alpha_{FRcalf} = 0$, $\alpha_{RLcalf} = 0$

Complete Failure ($\alpha = 0$)**.** To accurately measure the robot's actual velocity during this dynamic movement, we employed Vicon and compared it against the commanded velocity as a benchmark in the complete failure experiment. The velocity tracking results are presented in Fig. 6 and Fig. S8.

At the precise moment when the complete failure abruptly occurs at the front right leg, the robot loses its balance and collapses as the failed leg cannot support the body anymore. This behavior leads to significant changes in the robot's state (Fig. 1).

However, the robot autonomously detects this abnormality and adapts to it by modifying its actions. In particular, the rear right leg is strategically positioned as far away from the torso as possible (Fig. 1). This strategic placement enlarges the contact polygon, enhancing balance maintenance and providing the robot with a more favorable configuration for recovery. From this initial stance, the robot progressively lifts itself upright, even though this particular configuration is not conducive to walking.

Subsequently, the front left leg is positioned immediately adjacent to the body axes, maximizing the robot's mobility (Fig. 1). Due to the presence of joint friction, the swinging calf limb does not drop immediately after recovery but instead maintains a specific angle for an extended duration. As depicted in Fig. 1 and Fig. 7, the controller transitions to a novel motion pattern where the robot walks with the remaining three legs.

It is worth noting that, despite deviations between the real velocity and the constant command velocity during the failure recovery phase, the robot resumes velocity tracking once it has fully recovered (Fig. S8). The analogous analysis applies to the case when the rear left leg is damaged shown in Fig. 6.

D. Evaluation for Heuristic Terms

We evaluate heuristic guidance by firstly conducting terrain traversal experiments. Fig. S5(b)(c) reveals that the policy with the heuristic can generalize the policy even on the terrains that the robot has not seen during the training stage. As for the policy without heuristic, the robot is prone

Fig. 8: The robustness test is performed where the hardware failure is triggered and the robot is commanded to locomote on various terrain roughness.

to take more aggressive action with a longer step length. This behavior can easily cause overturn of the robot when the fault happens on uneven terrains. Even on the terrain featured in Fig. S5(a), which is used to train our policies, the incorporation of heuristic terms heavily reduces the unnecessary movement in both the feet placement and the COM behavior, as shown in Fig. S4. The other evaluation metrics are based on feasible region [32] and stability margin [28]. As for the policy with the heuristic guidance, the COM projection lies within the feasible region even over uneven terrain (Fig. S6(b)(c)), and exhibits a greater stability margin.

Hardware tests are also performed on the terrains encompass cobble, grass, and slopes, as depicted in Fig. 8. Even under normal circumstances, these terrains present certain challenges for the robot's locomotion. Remarkably, our controller achieved a 100% success rate in traversing these rough terrains under fault scenarios.

E. Comparison with Other Approaches

We conducted a comparative analysis of the proposed algorithm against several baseline approaches (Table II) including the state-of-the-art MPC (OCS2 [33]) and RL frameworks including domain randomization for the physical parameters (DRParam) [34], feeding the policy net with the non-encoded (NetParam) [35] or offline encoded (EncNetParam) [17], [18] physical parameters. For fairness, the RL-based methods are trained in the identical setup as our proposed pipeline, including the hyper-parameters, reward function, reset conditions, environment initialization, etc.

We evaluate their performance using the vertical distance between the position of the quadrupedal robot and the x-axis of the world coordinate system, after moving forward for 4 meters along its local x-axis. The results in Table II indicate that our training methodology leads to lower accumulated tracking errors and higher fault-tolerant endurance.

Our heuristic guiding terms are benchmarked with some existing methods, which explicitly guide the robot to perform recovery tasks or maintain fault tolerance: Baseline [29], GCPO [25](Encourage recovery via $\|ZMP-COM\|$), ACDR [8](Constant penalty)[2], and ours.

[2]We refer the reader to the supplementary for the corresponding details.

	Partial failure $\alpha = 0.3$	Partial failure $\alpha = 0.1$	Complete failure $\alpha = 0.0$		
	FD [m]	FD [m]	SsR(%)	DynR	FTL
OCS2	0.8	✗	✗	✗	✗
NetParam	0.15	0.21	62.5	✗	✗
DRParam	0.21	0.27	45.3	✗	✗
EncNetParam	0.08	0.18	78.1	✔	✔
FT-Net	**0.05**	**0.14**	**100.0**	✔	✔

TABLE II: Baseline comparison in Gazebo simulation. We report the mean values calculated by executing 5 trajectories for the 4 legs failure cases. The success rate of the recovery task is calculated by the success count out of 64 experiments in total. FT-Net outperforms the other methods in all tasks, particularly complete failure. (FD = Final deviation; SsR=Stand-still Recovery Success Rate; DynR=Dynamic Recovery; FTL=Fault-tolerant Locomotion)

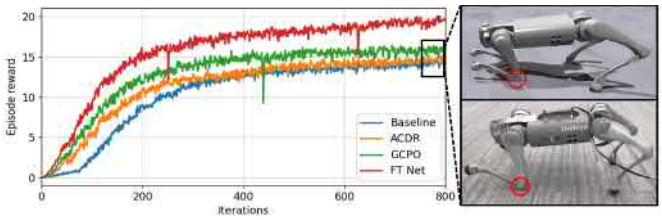

Fig. 9: (Left) Learning curves of different approaches. (Right) The inferior policy leads to the robot moving forward using the elbow on the faulty leg, in which the constant impulse potentially causes more damage.

It is shown in Fig. 9 (Left) that FT-Net outperforms all the other approaches with the highest learning curve. In particular, the baseline and the other two methods learn the policy that the robot is prone to support the body and move forward by using its elbow, and the robot's rear leg is heavily dragged during the motion, as shown in Fig. 9 (Right). This comparison strongly indicates that FT-Net can better guide the robot to recover from the failure state and yield higher fault tolerance.

V. CONCLUSIONS

This work develops a framework for the quadruped robots to overcome hardware failure during locomotion via an RL-based adaptive controller, which is trained with heuristic reward designs guided by the VHIP model and support polygon. The control policy is conditioned on a latent representation of failure scenarios. Unlike certain conventional methods, our approach eliminates the necessity for a preliminary procedure to locate the failed component. Furthermore, the neural network-based adaptor empowers the quadruped robot to dynamically adapt to unexpected failure in real-time during the deployment and implicitly identifies the failure category and its location. Notably, in cases of complete failure, the controller intelligently transits the robot from a quadrupedal mode to a tripedal mode. The experiment also demonstrates the ability to effectively address failures occurring across various terrains.

Several promising extensions of this work are 1) Generalization to various hardware platforms: This fault-tolerant controller provides a solution to the failure of other articulated robots, such as quadruped robots with distinct morphology and bipedal robots; 2) Foothold optimization:

The gait schedule, foot trajectories, and contact forces optimization could lead to a more natural failure gait. This could be resolved by integrating the model-based motion planning method [36], which predicts the foothold sequence. 3) Bio-inspired reward terms: Current framework can be improved by learning from the motion clip dataset [37] of the real injured dog, which can be collected by pose tracking techniques.

ACKNOWLEDGMENT

The authors gratefully thank Rui Huang, Zhengjie Shu, Yinzhao Dong, and Xinqi Li for the hardware support.

REFERENCES

[1] J.-M. Yang, "Kinematic constraints on fault-tolerant gaits for a locked joint failure," *Journal of Intelligent and Robotic Systems*, vol. 45, pp. 323–342, 2006.

[2] Z. Chen, Q. Xi, F. Gao, and Y. Zhao, "Fault-tolerant gait design for quadruped robots with one locked leg using the gf set theory," *Mechanism and Machine Theory*, vol. 178, p. 105069, 2022.

[3] C.-M. Lin and C.-H. Chen, "Robust fault-tolerant control for a biped robot using a recurrent cerebellar model articulation controller," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 1, pp. 110–123, 2007.

[4] M. M. Gor, P. M. Pathak, A. K. Samantaray, J. M. Yang, and S. Kwak, "Fault-tolerant control of a compliant legged quadruped robot for free swinging failure," *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, vol. 232, no. 2, pp. 161–177, 2018.

[5] V. Barasuol, J. Buchli, C. Semini, M. Frigerio, E. R. De Pieri, and D. G. Caldwell, "A reactive controller framework for quadrupedal locomotion on challenging terrain," in *2013 IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 2554–2561.

[6] D. Liu, T. Zhang, J. Yin, and S. See, "Saving the limping: Fault-tolerant quadruped locomotion via reinforcement learning," *arXiv:2210.00474*, 2022.

[7] T. Anne, J. Wilkinson, and Z. Li, "Meta-learning for fast adaptive locomotion with uncertainties in environments and robot dynamics," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 4568–4575.

[8] W. Okamoto, H. Kera, and K. Kawamoto, "Reinforcement learning with adaptive curriculum dynamics randomization for fault-tolerant robot control," *arXiv:2111.10005*, 2021.

[9] J. Liu, H. Chen, P. M. Wensing, and W. Zhang, "Instantaneous capture input for balancing the variable height inverted pendulum," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7421–7428, 2021.

[10] C. Pana, I. Resceanu, and D. Patrascu, "Fault-tolerant gaits of quadruped robot on a constant-slope terrain," in *2008 IEEE International Conference on Automation, Quality and Testing, Robotics*, vol. 1. IEEE, 2008, pp. 222–226.

[11] Y. Y. Zhao, J. H. Wang, B. W. Zhang, X. Yao, and G. H. Cao, "Gait planning and fault-tolerant control of quadruped robots," in *Third International Conference on Mechanical, Electronics, and Electrical and Automation Control (METMS 2023)*, vol. 12722. SPIE, 2023, pp. 841–846.

[12] Y. Farid, V. J. Majd, and A. Ehsani-Seresht, "Fractional-order active fault-tolerant force-position controller design for the legged robots using saturated actuator with unknown bias and gain degradation," *Mechanical Systems and Signal Processing*, vol. 104, pp. 465–486, 2018.

[13] J. Stückler, J. Schwenk, and S. Behnke, "Getting back on two feet: Reliable standing-up routines for a humanoid robot." in *IAS*. Citeseer, 2006, pp. 676–685.

[14] J. A. Castano, C. Zhou, and N. Tsagarakis, "Design a fall recovery strategy for a wheel-legged quadruped robot using stability feature space," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2019, pp. 41–46.

[15] Y. Farid, V. J. Majd, and A. Ehsani-Seresht, "Dynamic-free robust adaptive intelligent fault-tolerant controller design with prescribed performance for stable motion of quadruped robots," *Adaptive Behavior*, vol. 29, no. 3, pp. 233–252, 2021.

[16] B. Thananjeyan, A. Balakrishna, S. Nair, M. Luo, K. Srinivasan, M. Hwang, J. E. Gonzalez, J. Ibarz, C. Finn, and K. Goldberg, "Recovery rl: Safe reinforcement learning with learned recovery zones," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4915–4922, 2021.

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

[18] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," *arXiv:2107.04034*, 2021.

[19] S. Gangapurwala, M. Geisert, R. Orsolino, M. Fallon, and I. Havoutis, "Rloc: Terrain-aware legged locomotion using reinforcement learning and optimal control," *IEEE Transactions on Robotics*, vol. 38, no. 5, pp. 2908–2927, 2022.

[20] L. Smith, J. C. Kew, X. B. Peng, S. Ha, J. Tan, and S. Levine, "Legged robots that keep on learning: Fine-tuning locomotion policies in the real world," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 1593–1599.

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

[22] S. Koos, A. Cully, and J.-B. Mouret, "Fast damage recovery in robotics with the t-resilience algorithm," *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1700–1723, 2013.

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

[24] C. Yang, K. Yuan, Q. Zhu, W. Yu, and Z. Li, "Multi-expert learning of adaptive legged locomotion," *Science Robotics*, vol. 5, no. 49, p. eabb2174, 2020.

[25] S. Gangapurwala, A. Mitchell, and I. Havoutis, "Guided constrained policy optimization for dynamic quadrupedal robot locomotion," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3642–3649, 2020.

[26] Y. Ma, F. Farshidian, and M. Hutter, "Learning arm-assisted fall damage reduction and recovery for legged mobile manipulators," *arXiv:2303.05486*, 2023.

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

[28] S. Kajita and B. Espiau, "Legged robot," in *Springer handbook of robotics*. Springer Berlin/Heidelberg, Germany, 2008, pp. 361–389.

[29] 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.

[30] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, *et al.*, "Isaac gym: High performance gpu-based physics simulation for robot learning," *arXiv:2108.10470*, 2021.

[31] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds. Curran Associates, Inc., 2017, pp. 4765–4774.

[32] R. Orsolino, M. Focchi, S. Caron, G. Raiola, V. Barasuol, D. G. Caldwell, and C. Semini, "Feasible region: An actuation-aware extension of the support region," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1239–1255, 2020.

[33] F. Farshidian, M. Neunert, A. W. Winkler, G. Rey, and J. Buchli, "An efficient optimal planning and control framework for quadrupedal locomotion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 93–100.

[34] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2017, pp. 23–30.

[35] W. Yu, J. Tan, C. K. Liu, and G. Turk, "Preparing for the unknown: Learning a universal policy with online system identification," *arXiv:1702.02453*, 2017.

[36] G. B. Margolis and P. Agrawal, "Walk these ways: Tuning robot control for generalization with multiplicity of behavior," in *Conference on Robot Learning*. PMLR, 2023, pp. 22–31.

[37] X. B. Peng, Z. Ma, P. Abbeel, S. Levine, and A. Kanazawa, "Amp: Adversarial motion priors for stylized physics-based character control," *ACM Transactions on Graphics (ToG)*, vol. 40, no. 4, pp. 1–20, 2021.