Towards Fault-tolerant Quadruped Locomotion with Reinforcement Learning

Dikai Liu[1], Jianxiong Yin[1] and Simon See[1,2]

***Abstract*—Modern quadrupedal robots are skilled in navigating through challenging terrains in remote uncontrolled environments with recent advances in reinforcement learning (RL). However, survival in the wild requires not only maneuverability, but also the ability to handle potential critical hardware failures. How to grant such ability to quadrupeds with RL is rarely investigated. In this paper, we propose a novel methodology to enable fault tolerance for RL-based quadruped locomotion controller with joint teacher-student framework for fast zero-shot knowledge transfer that can be deployed to a physical robot without any fine-tuning. With no dedicated reward design for gait guidance, the designed simulation and training strategy can be easily added on top of existing RL-based controllers and generalized to unseen situations. Extensive experiments show that our fault-tolerant controller can efficiently lead a quadruped stably when it faces joint failures during locomotion.**

I. Introduction

Quadrupedal robots have demonstrated enhanced intelligence in solving a wide range of tasks with high flexibility and versatility in complex environment. They are commonly deployed in remote uncontrolled environments [1], where potential accidents could cause critical hardware failures, e.g., joint locking, free swinging, broken brackets. These failures could cause significant harm to both robots and humans, increase downtime, and shorten the service life of the robot. Therefore, it is crucial for the controller to exhibit robustness against hardware failures.

Recent advances in reinforcement learning (RL) have led to solutions in various quadruped locomotion tasks, such as traveling through rough terrains [1]–[3], jumping & falling recovery [4] and running at high speeds [5]. However, these studies typically presume *normal operating conditions*. With limited hardware failure detection (e.g., motor overheating, sensor signal loss) or protection functions (e.g., shutting down the system) on existing quadrupedal systems, the development of fault-tolerant controllers remains an open problem, even with well-studied conventional approaches [6]–[9].

RL can significantly relax the requirement of extensive prior domain knowledge through exploration and exploitation. Robotic agents are usually first trained in simulation and then transferred to the physical world [2], [3], [5]. The existence of the sim-to-real gap remains a major barrier to achieve zero-shot transfer [10]–[12]. This gap is especially crucial in the event of hardware failure, as many existing work is far from the reality for failure simulation [13]. Recent work proposed several RL-based controllers to achieve fault tolerance [14], [15], which are only trained and tested in the simulation with fixed predefined failure. Due to the huge sim-to-real gap, it is unclear how these methods will perform in the physical world and it is challenging to scale to different failure situations [14]. Some other work used specially designed reward functions to drive the policy to a dedicated gait control under failure [13], which cannot be easily implemented in every RL-based controller and generalized to other failures.

[1] NVIDIA AI Technology Centre (NVAITC); e-mail: {dikail,jianxiongy,ssee}@nvidia.com
[2] also with Coventry University and Mahindra University

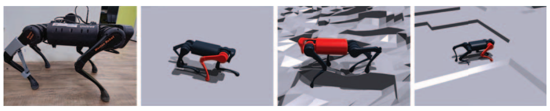


Fig. 1. Physical robot and its simulated counterpart. Unitree A1 is equipped with our joint locking mechanism. Its official URDF model is used in the Isaac Gym simulator [16] with body links of the locked joint showing in red.

Motivated by these limitations, we design a novel framework to achieve robust fault-tolerance quadruped locomotion in the physical world with the following contributions. (1) We design a simple yet efficient way to enable fault tolerance against joint locking, which can be a direct add on top of an existing RL-based controller without pre-designed gait control and additional dedicate reward design as guidance. (2) We introduced joint optimization in the teacher-student reinforcement learning paradigm to achieve fast zero-shot transfer in a single phase. When deployed in a physical quadruped, the policy can provide real-time locomotion control against possible hardware failures. Extensive experiments are conducted in both simulation and a physical Unitree AI robot (Fig. 1). Evaluations show that our method can significantly improve the robustness and hardware fault tolerance.

II. Methodology

Our proposed framework can be added on top of an existing RL-based controller to train a control policy π to guide the stable locomotion of the quadruped even when it faces critical joint locking failures. We choose RMA [3] as the baseline, a well-recognized base framework used in [5], [17], [18]. Fig. 2 presents an overview of our methodology.

A. Joint Failure Modeling in Simulation

We mainly focus on the single joint locking failure, which is also the target of recent related works [9], [14], [15]. A locked joint has limited range of motion, but can still apply torque to support the body.

An RMA-like [3] vanilla environment `BaseEnv` is first created, where no failure occurs. Then, for each virtual agent,

979-8-3503-5409-6/24/$31.00 ©2024 IEEE
DOI 10.1109/CAI59869.2024.00257

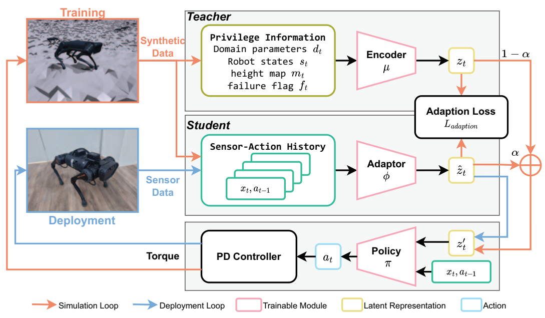


Fig. 2. Methodology overview. The teacher-student framework from [3], [5] are adopted to train the policy. During training, synthetic data are used to compute the latent representation z_t and $\hat{z}_t$, which are fused for joint optimization. The student and policy are then directly deployed with zero-shot transfer on the physical robot.

we randomly sample the failure time $T_f \sim \mathcal{U}(T^f_{min}, T^f_{max})$, the failure joint $J_f \sim \mathcal{U}\{1, \ldots, 12\}$ and the failure tolerance $\theta_{tol} \sim \mathcal{N}(0, {\theta_{max}}^2)$. The failure status is tracked by a failure flag $f_t \in [0, 12]$. Initially and after every reset, f_t is cleared as 0 to indicate a normal state. In the episode, when the agent progresses to T_f, the failure occurs and f_t is updated to reflect the joint failure $f_t = J_f$. The current position of the selected joint J_t is used as the central failure angle $\bar{\theta} = q_{J_t}$. Joint locking failure is modeled by restricting joint movement with a limited range $\theta_{allowed}$, controlled by the central position $\bar{\theta}$ and symmetric tolerance θ_{tol}, which are used to directly overwrite the joint's limit with Isaac Gym's API.

We refer to the failure environment as `FailureEnv`. Unlike previous methods [14], [15], where joint failures are predefined and fixed, we use domain randomization to generate versatile and unpredictable situations. Since joint locking directly affects joint control, the robot status and surroundings at the failure moment can greatly alter the result, online randomization can help to train a robust and generalized policy against various joint locking accidents.

B. Reinforcement Learning Architecture

We closely follow RMA [3] for observation, action, and reward design to take data from the onboard sensor and output the optimal joint position.

Observation. Data from the onboard sensors are collected to provide observations. At any time t, joint encoders, IMU and foot encoders provide noisy sensor data $x_t \in \mathbb{R}^{30}$, which consists of joint position $q \in \mathbb{R}^{12}$, joint velocity $\dot{q} \in \mathbb{R}^{12}$, gravity vector $g \in \mathbb{R}^2$ and binary foot contact $c \in \mathbb{R}^4$. The previous actions $a_{t-1} \in \mathbb{R}^{12}$ are further added to form the observation $o_t = [x_t, a_{t-1}] \in \mathbb{R}^{42}$. Historical observations of length $H = 50$ are used to capture temporal information. The privilege information contains the domain randomized parameter (payload COM, mass, motor strength, friction) $d \in \mathbb{R}^{16}$, the state of the robot (linear and angular velocity) $s \in \mathbb{R}^6$, the surrounding height map $m \in \mathbb{R}^{140}$ and the failure flag $f \in \mathbb{R}^1$ introduced in II-A.

Action. The policy outputs the target joint position $\hat{q} = a_t \in \mathbb{R}^{12}$, which is then processed by a PD controller for the desired torque $\tau = K_p(\hat{q} - q) + K_d(\hat{\dot{q}} - \dot{q})$, where K_p and K_d are the stiffness and damping gain and target velocity $\hat{\dot{q}}$ is set to 0.

Reward Function. The reward functions encourage the agent to move forward stably and smoothly with a target speed of 0.5 m/s. Penalization is given mainly for movement in other axes, such as lateral movement and yawing, large joint acceleration, power consumption, and collision with the robot body. In addition, no special reward for failure handling is designed.

C. Joint Teacher-Student Framework

Utilizing the privileged information of the robot and environment can produce better performance faster [19], [20]. The teacher-student learning paradigm [2], [3], [5] enables implicit identification of the hidden dynamics of the environment and robot from perceivable data for direct deployment.

To optimize the student adaptor ϕ, previous work [2], [3], [5] focuses on imitating the behaviors of the teacher model μ using DAgger-inspired supervised learning [21] to minimize the difference in latent representation $\mathcal{L}_{adaption} = \|z_t - \hat{z}_t\|^2$. However, even a slight difference in the latent space can cause unpredictable behavior and performance degradation, especially under joint failure. We propose to jointly optimize all modules by fusing the latent representations:

$$a_t = \pi[\alpha \hat{z}_t + (1-\alpha) z_t, o_t]$$

We further append the PPO [22] loss $\mathcal{L}_{RL}$ with $\mathcal{L}_{adaption}$, which is similar to [23], [24]:

$$\mathcal{L} = \mathcal{L}_{RL} + \beta \mathcal{L}_{adaption}$$

In the early stage, we set $\alpha = 0$ to train π solely with privilege information and minimize the adaption loss simultaneously. When reasonable control commands can be given with z_t, α gradually increases with the progression of training, and β is negatively correlated, until only $\hat{z}_t$ is used for policy making. In this way, even if we cannot obtain a perfect replica of z_t, the focus of optimization shifts to the reward benefits, and π can still be optimized.

III. Evaluation

A. Implementation and Experimental Setup

Module Implementation. Both the teacher model μ and control policy ϕ are implemented in MLP with hidden layers of $[512, 256, 128]$ and $[256, 128]$, respectively, with ELU activation. μ outputs the latent representation with length $D = 8$. Two different student models are implemented to capture temporal information, one with vanilla 1D CNNs following [3] and another with TCN [25] following [2].

Simulation. Isaac Gym and its open source library IsaacGymEnvs [16] are used to simulate massive parallel environments with rough terrains, including rough sloped terrain, smooth sloped terrain and discrete obstacles [26]. The simulation runs on two NVIDIA A6000 GPUs, each handling 4096 environments at 200Hz, which can provide more than 0.1M FPS for simulation. The controller runs at 50Hz for command.

Hardware. Unitree A1 is used as the test platform with an external NVIDIA Jetson Xavier NX for GPU acceleration to process the exported JIT model for position control.

TABLE I
NORMALIZED REWARD RETURN FOR DIFFERENT STUDENT POLICY.

Environment	[T]		[JT]		[SS]			
	600M	300M	600M		300M+300M		600M+600M	
			CNN	TCN	CNN	TCN	CNN	TCN
BaseEnv	1	0.92	0.89	0.92	0.15	0.77	0.85	**0.93**
FailureEnv	1	0.85	**0.97**	0.85	0.20	0.73	0.81	0.84

B. Teacher-Student Transfer

To better demonstrate the efficiency of our proposed joint training, the student network is trained with: the proposed joint training (**[JT]**) and separate supervised (**[SS]**) used in RMA.

- **[T]:** Teacher is trained in two configurations as oracle: **600M**; and **300M** with half of the simulated frame for use in limited frame supervised transfer.
- **[JT]:** this is trained with standard **600M** frames.
- **[SS]:** this is fully trained with **600M+600M** frames to achieve optimal performance. To compare the performance under the same total frame as **[JT]**, an additional configuration of **300M+300M** is used with teacher and supervised training stages equally divided.

The trained policies are then deployed in the simulation and the average return is shown in Table I. The results are normalized based on the corresponding teacher performance.

Despite supervised transfer showing great performance while fully trained, especially with TCN, it struggles when the total simulation step is limited. For joint training, it can achieve the same level or even outperform **[SS]** transfer with only half of the simulation step required in both environment, indicating a superior efficiency in knowledge transfer.

C. Virtual Deployment

Overall Performance. Both student policies are deployed into the same test environment where robots are spawned across different terrains and levels evenly with joint locking failure occurs randomly. Each virtual robot can run a maximum of 20 seconds after joint failure occurs. The forward velocity both before and after joint locking are tracked and the survival time of each agent is measured on average, 25% percentile (P25) and 50% percentile (P50) so that we can see how each agent handles joint locking in the worst scenarios. The result averaged over 1500 instances per terrain is shown in Table II.

Before joint failure, both agents can drive the robot forward close to the target velocity of 0.5 m/s. After failure occurs, the velocity drops in both agents, but the fault-tolerant `FailureEnv` agent maintain the velocity slightly better. Despite `BaseEnv` agent is more vulnerable to joint locking and fails within 5s for half of the instances. In contrast, the fault-tolerant `FailureEnv` agent can survival much longer with a locked joint. In smooth slope and rough slope terrains, most of the robot can even survive to the end of the journey. Due to the small physical size of the A1 robot, discrete obstacle terrain is challenging even under normal conditions [17], [18], making joint failure more deadly in this environment.

Gait Pattern Analysis. To understand how `FailureEnv` agent handles joint failure, the gait pattern of foot contact is captured during deployment around the failure moment, with F/R denoting front/rear and L/R denoting left/right. Two instances are shown in Fig. 3. With an unexpected joint locking failure, the agent can quickly adapt based on the actual situation and react accordingly. While the top instance can keep the previous gait, the bottom instance adjust the motion to drag the failure leg forward. The dynamic adjustment for different situations demonstrates the generalization ability of `FailureEnv` agent in the handling of failures without relaying on some predefined pattern.

TABLE II
AGENT PERFORMANCE WITH JOINT FAILURE IN SIMULATION.

Agent	Terrain	Avg. Forward Velocity (m/s)		Survival Time (%)		
		Before	After	Average	P25	P50
BaseEnv	Smooth Slope	0.56	0.45	51.4	6.7	36.5
	Rough Slope	0.55	0.41	44.4	4.7	20.7
	Discrete	0.54	0.41	40.8	4.4	17.5
	All	0.55	0.42	44.7	5.0	21.4
FailureEnv	Smooth Slope	0.59	0.52	69.3	20.1	100.0
	Rough Slope	0.57	0.47	59.1	11.7	81.0
	Discrete	0.55	0.44	45.8	6.6	31.0
	All	0.57	0.47	56.5	10.8	59.0

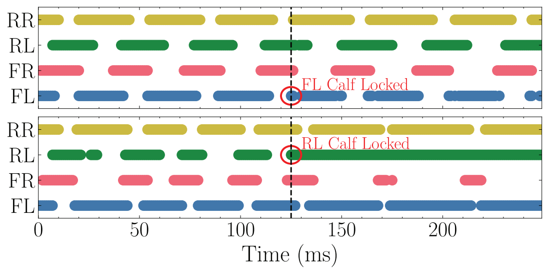


Fig. 3. Gait pattern of two instances during virtual deployment. The difference gait pattern shows that the controller can dynamically adjust to joint failure.

Multiple Joint Failure. We further evaluated the ability of `FailureEnv` agent to handle multiple joint locking failures, which is never seen during training.

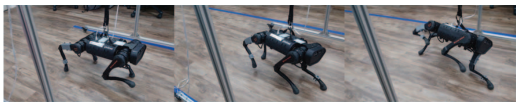
Although the `FailureEnv` agent can easily handle two joint locks simultaneously, it becomes more difficult to maintain the heading. With more joints locked, the agent begins to struggle, especially when the joints are distributed across multiple legs. When all failures occur on the same leg, even with all three joints locked, resulting in the loss of a whole leg, the agent can compensate for the loss with other legs.

D. Physical Validation

To perform physical validation, we design two strategy: *softlock* and *hardlock*. With *softlock*, we limit the output joint position to the failure range, so that failure can happen at any time and in any joint. However, some controllers cannot be easily manipulated (built-in controller), and we design a 3D-printed adjustable locking mechanism to *hardlock* the calf joint. Fig. 4 shows the snapshots of the trials.

We measure the survival time in the real world with a maximum lifetime of 20 seconds in Table III. The fault-tolerant `FailureEnv` agent can handle all the test seniors while the vanilla `BaseEnv` agent struggles on thigh and

(a) Running with joint locking by the `FailureEnv` agent


(b) Running with joint locking by the `BaseEnv` agent

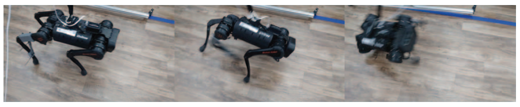

(c) Running with joint locking by built-in controller

Fig. 4. Deployment snapshots on the physical robot run by (a) fault-tolerant `FailureEnv` agent with one (top) and two (bottom) joint locked, (b) baseline `BaseEnv` agent and (c) A1's built-in controller.

TABLE III
AVERAGE SURVIVAL TIME IN PHYSICAL TESTS UNDER DIFFERENT JOINT LOCKING

Agent	Softlock			Hardlock
	Hip	Thigh	Calf	
FailureEnv	100%	100%	100%	100%
BaseEnv	100%	20%	5%	35%
Built-in	-	-	-	0%

calf joint and the robot stalls or falls directly to the ground. We further lock two joints for `FailureEnv` policy and the quadruped can still move safely, even though this situation is never seen during training.

IV. Conclusion and Future Work

In this study, we propose a novel approach for fault-tolerant RL-based quadruped locomotion. We design a joint locking failure simulation strategy with a joint training pipeline for efficient teacher-student transfer with an existing RL controller without additional reward and gait design. We demonstrate that our controller can be zero-shot transferred to physical robot and is robust under various joint locking failures.

Quadrupedal robot fault tolerance is a complex topic considering the variety robot model, use cases, and failure type. The current work can be extended with user input and high-level sensors for advanced tasks. To further improve the robustness of the locomotion controller, a unified solution can be developed for different joint failures (joint locking and free swinging). A generalized and transferable solution can be explored for different quadrupedal robotic platforms.

References

[1] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, "Learning robust perceptive locomotion for quadrupedal robots in the wild," *Science Robotics*, vol. 7, no. 62, 2022.

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

[3] A. Kumar, Z. Fu, D. Pathak, and J. Malik, "Rma: Rapid motor adaptation for legged robots," in *Robotics: Science and Systems*, 2021.

[4] H.-W. Park, P. M. Wensing, and S. Kim, "Jumping over obstacles with mit cheetah 2," *Robotics and Autonomous Systems*, vol. 136, 2021.

[5] G. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, "Rapid locomotion via reinforcement learning," in *Robotics: Science and Systems*, 2022.

[6] J.-M. Yang, "Kinematic constraints on fault-tolerant gaits for a locked joint failure," *Journal of Intelligent and Robotic Systems*, 2006.

[7] 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.

[8] M. Gor, P. M. Pathak, A. Samantaray, J.-M. Yang, and S. Kwak, "Fault accommodation in compliant quadruped robot through a moving appendage mechanism," *Mechanism and Machine Theory*, 2018.

[9] J. Cui, Z. Li, J. Qiu, and T. Li, "Fault-tolerant motion planning and generation of quadruped robots synthesised by posture optimization and whole body control," *Complex & Intelligent Systems*, 2022.

[10] 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.

[11] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *IEEE international conference on robotics and automation (ICRA)*, 2018.

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

[13] Z. Luo, E. Xiao, and P. Lu, "Ft-net: Learning failure recovery and fault-tolerant locomotion for quadruped robots," *IEEE Robotics and Automation Letters*, 2023.

[14] 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.

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

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

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

[18] H. Lai, W. Zhang, X. He, C. Yu, Z. Tian, Y. Yu, and J. Wang, "Sim-to-real transfer for quadrupedal locomotion via terrain transformer," in *IEEE International Conference on Robotics and Automation*, 2023.

[19] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq *et al.*, "Deepmind control suite," *arXiv preprint arXiv:1801.00690*, 2018.

[20] M. Laskin, A. Srinivas, and P. Abbeel, "Curl: Contrastive unsupervised representations for reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2020.

[21] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *International conference on artificial intelligence and statistics*, 2011.

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

[23] H. Wu, K. Khetarpal, and D. Precup, "Self-supervised attention-aware reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, no. 12, 2021.

[24] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath, "Learning humanoid locomotion with transformers," *arXiv preprint arXiv:2303.03381*, 2023.

[25] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[26] 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.