Contrastive Forward Prediction Reinforcement Learning for Adaptive Fault-Tolerant Legged Robots

Yangqing Fu[1] **Yang Zhang**[1] **Qiyue Yang**[1] **Liyun Yan**[1]
Zhanxiang Cao[1] **Yue Gao**[1,2†]
[1]Shanghai Jiao Tong University [2]Shanghai Innovation Institute

Abstract: In complex environments, adaptive and fault-tolerant capabilities are essential for legged robot locomotion. To address this challenge, this study proposes a reinforcement learning framework that integrates contrastive learning with forward prediction to achieve fault-tolerant locomotion for legged robots. This framework constructs a forward prediction model with contrastive learning, incorporating a comparator and a forward model. The forward model predicts the robot's subsequent state, and the comparator compares these predictions with actual states to generate critical prediction errors. These errors are systematically integrated into the controller, facilitating the continuous adjustment and refinement of control signals.Experiments on quadruped robots across different terrains and various joint damage scenarios have verified the effectiveness of our method, especially the functions of the comparator and the forward model. Furthermore, robots can adapt to locked joints without prior training, demonstrating zero-shot transfer capability. Finally, the proposed method demonstrates universal applicability to both quadruped and hexapod robots, highlighting its potential for broader applications in legged robotics.

Keywords: Legged robot locomotion, Fault tolerance control, Deep Reinforcement learning

1 Introduction

Legged robots have demonstrated remarkable performance in navigating complex and unstructured environments [1, 2]. Their ability to traverse a wide range of terrains makes them well-suited for tasks such as exploration and search-and-rescue operations, especially in scenarios where human access is limited or dangerous [3, 4]. However, despite significant progress in their mechanical design and control systems, legged robots remain susceptible to mechanical faults, particularly joint damage, which can severely degrade their locomotion capabilities [5, 6].

Joint damage can stem from various sources, including long-term wear and tear, sudden collisions, and manufacturing inconsistencies [7]. Such damage often leads to a reduction in mobility, control precision, and overall stability, significantly affecting the robot's task performance in real-world scenarios [8, 9]. Ensuring stable and reliable locomotion in the presence of joint damage remains a critical and open challenge in the field of legged robotics [10, 11].

To address fault tolerance in legged robots, model-based control methods leverage prior dynamics knowledge but require extensive tuning and struggle with unmodeled damage [9, 8]. Data-driven methods like deep reinforcement learning (DRL) enable adaptive learning [12], yet often lack generalization across diverse damage scenarios due to limited representational capacity. Imitation learning from injured agents [13] and behavior-performance mapping [14] offer alternative solutions, but face challenges in data diversity, scalability, and robustness to unseen or out-of-distribution faults.

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

In light of these limitations, this work proposes a novel fault-tolerance learning framework that explicitly incorporates predictive modeling into the control loop. The key idea is to introduce a forward predictive model and a comparator mechanism that jointly estimate future robot states based on the current state and control command. Discrepancies between the predicted and actual observed states are computed as prediction errors, which are then used to adapt and refine control signals. This feedback mechanism allows the controller to iteratively improve its fault tolerance and adapt to a wide range of joint damage conditions without relying on pre-defined damage patterns or manual intervention.

The main contributions of this work are summarized as follows: i) We propose a novel reinforcement learning-based framework for fault tolerance control, which integrates forward prediction and error-based feedback to enhance adaptability under joint damage conditions. ii) The framework improves the representation capacity of the policy and enables dynamic adjustment of control actions based on prediction errors, enhancing robustness across various terrains and fault scenarios. iii) We validate the framework in both simulation and real-world experiments on quadruped and hexapod robots, demonstrating its generalization ability across different robot morphologies. iv) The results show strong zero-shot generalization and robustness to out-of-distribution (OOD) joint damage, confirming the practicality of the method in real deployment.

2 Related Work

2.1 DRL-based Legged Robot Locomotion

DRL has demonstrated impressive capabilities in learning robust locomotion policies for legged robots in both simulation and real-world scenarios [15, 16, 17]. Early DRL methods focused on training policies in simulation with extensive environment randomization to improve sim-to-real transfer [18, 1]. These methods enabled robots to master complex walking, running, and even agile maneuvers purely from experience [19, 20].

Subsequent work has aimed at improving sample efficiency and policy generalization. Techniques such as curriculum learning [21, 22], meta-reinforcement learning [23], and domain adaptation [24] have been proposed to make DRL more practical for real-time and real-world deployment. Moreover, proprioceptive-only DRL frameworks [2, 25] have demonstrated strong generalization on unstructured terrains, highlighting the potential of minimal-sensor learning.

2.2 Fault-tolerant Control

Fault-tolerant control for legged robots has attracted growing interest due to the challenges posed by complex dynamics and unpredictable environments. Traditional approaches rely on model-based control [8, 9], where fault compensation is achieved through dynamic modeling and parameter tuning. While effective in structured scenarios, these methods struggle to generalize due to their reliance on accurate models and hand-engineered rules.

DRL offers a scalable alternative by learning robust policies directly from data. Recent works have addressed joint failures via DRL frameworks that adapt to impaired conditions [12], including multi-task learning for fault generalization [26] and resilient zero-shot adaptation [27, 28]. The UMC controller [29] further improves robustness through modular architectures and dynamic masking. However, DRL policies often lack expressive internal representations to distinguish nuanced fault patterns.

Imitation learning is another direction, where locomotion policies are guided by expert data or biological inspiration. For example, injured-animal gait data has been used to train adaptive controllers [30], and manually designed gait trajectories have enabled recovery behaviors under limb damage [31]. These approaches, though intuitive, are constrained by data quality and transferability.

Different from previous work, we enhance the fault tolerance capability of control policies from two key perspectives: contrastive learning and forward prediction. Contrastive learning is employed

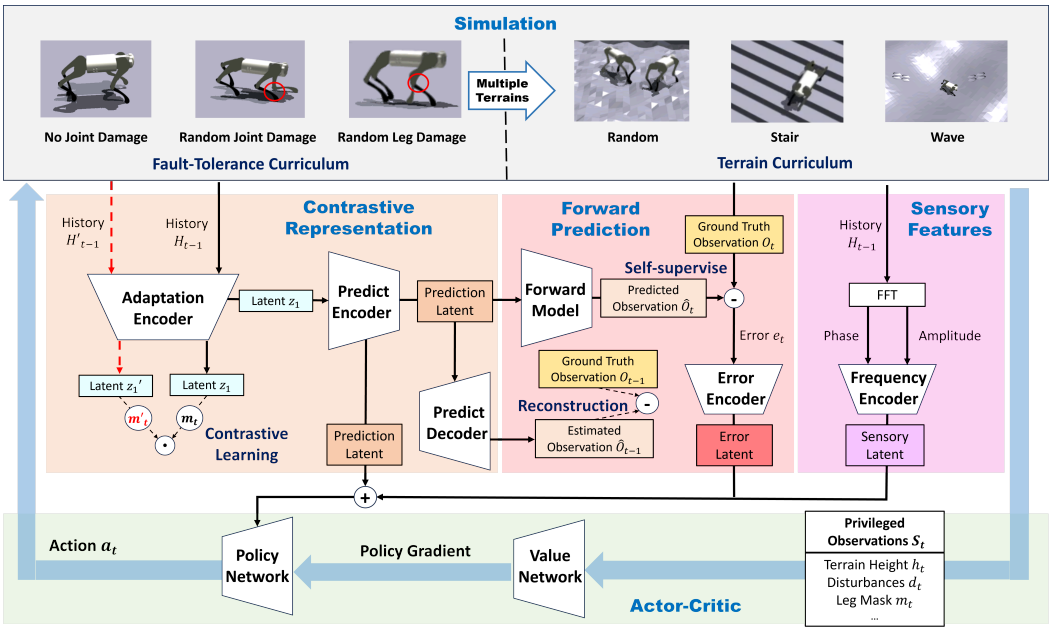


Figure 1: The specific training process of the proposed learning framework. Prediction latent, error latent, and sensory latent from three parts of the networks are calculated and utilized as input features for the policy network.

to improve the representation quality of the policy's latent space, enabling better discrimination of various joint damage conditions.

3 Method

3.1 Framework Overview

Figure 1 illustrates the overall structure of the proposed method. In the simulation environment, the robustness of the control policy is progressively enhanced through a curriculum learning strategy. The learning process begins with an intact robot model and incrementally introduces more complex conditions such as partial joint impairments and complete leg malfunctions. This staged training allows the policy to adapt steadily, avoiding sudden exposure to extreme scenarios and facilitating generalization across various fault conditions.

The system architecture primarily consists of two modules: a representation module and a prediction module. The representation module leverages contrastive learning to improve the expressiveness of the latent state encoder, which helps the agent to effectively identify and adapt to different damage conditions. The prediction module estimates future states based on historical observations H_{t-1} and latent variables z_1. By forecasting upcoming observations, the agent can prepare for future events and make proactive decisions, even under uncertain or impaired dynamics. This forward model is trained in a self-supervised manner, using the error between predicted and actual observations to iteratively refine its predictions.

This prediction error, denoted as e_t, is subsequently encoded and incorporated into the policy learning process. Serving as a key feedback signal, it reflects the accuracy of the internal model and helps guide policy updates in a direction that mitigates the impact of modeling inaccuracies or environmental disturbances.

Moreover, Fast Fourier Transform (FFT) is applied to the motion data to extract periodic features. These features, which are not easily visible in the time domain, are embedded via a dedicated en-

coder into high-dimensional latent spaces, enriching the model's perception and aiding in more informed decision-making under dynamic and degraded conditions.

3.2 Fault-tolerant Representation

To effectively handle potential failures, it is essential to learn fault-tolerant representation. Contrastive learning plays a critical role in the development of legged robots by enhancing their feature representation capabilities. This method improves the robot's ability to distinguish between various movement patterns and terrain characteristics by minimizing the distance between similar samples and maximizing the distance between dissimilar ones [32, 33]. As shown in Figure 1**c**, the simulation environment provides historical observations $H_{t-1} = [O_{t-5}, ..., O_{t-1}]$ for the **Adaptation Encoder** to map high-dimensional features into the hidden state z_1. In a batch containing various joint damage data, contrastive learning can increase the distance in the latent space between z_1 corresponding to different leg masks. The leg mask $m_t = [1, 0, ..., 1]$ is a coded vector that indicates the condition of the robot's joints, where 1 represents a healthy joint and 0 represents a damaged joint. The specific optimization objective of the loss function is expressed as:

$$\mathcal{L}_{\text{CL}} = -\frac{1}{N}\sum_{i=1}^{N}\frac{1}{\sum_{j=1}^{N}\mathbb{I}(y_i = y_j)}\sum_{j=1}^{N}\mathbb{I}(y_i = y_j)\left(\frac{\mathbf{z}_i \cdot \mathbf{z}_j^\top}{\alpha} - \log\sum_{k=1}^{N}\exp\left(\frac{\mathbf{z}_i \cdot \mathbf{z}_k^\top}{\alpha}\right)\right), \quad (1)$$

where N is the number of samples. $\mathbf{z}_i$ and $\mathbf{z}_j$ are the feature representations of samples i and j. The term $\mathbf{z}_i \cdot \mathbf{z}_j^\top$ represents the dot product between these feature vectors, and $\frac{\mathbf{z}_i \cdot \mathbf{z}_j^\top}{\alpha}$ denotes the similarity score between the samples i and j, where α is a temperature parameter. The indicator function $\mathbb{I}(y_i = y_j)$ is 1 if the samples i and j belong to the same m_t class and 0 otherwise. The normalization factor $\frac{1}{\sum_{j=1}^{N}\mathbb{I}(y_i=y_j)}$ adjusts for the number of positive samples for each sample i. The term $\frac{\mathbf{z}_i \cdot \mathbf{z}_j^\top}{\alpha}$ represents the similarity score for positive pairs (samples with the same m_t), while the term $\log\sum_{k=1}^{N}\exp\left(\frac{\mathbf{z}_i \cdot \mathbf{z}_k^\top}{\alpha}\right)$ represents the log of the sum of similarity scores between the sample i and all other samples. This reflects the overall similarity distribution.

3.3 Forward Model

In model-based reinforcement learning [34, 35], the forward model plays a crucial role in predicting the future state of the environment given the current state and an action. This model allows the agent to simulate and evaluate the outcomes of potential actions without interacting with the real environment. Different from model-based reinforcement learning, where the forward model is typically used to predict future states, our forward model primarily serves to provide error information to the **Error Encoder**. As shown in Figure 1, the prediction latent is utilized as input to **Forward Model**. Self-supervise learning is utilized to optimize this model. The loss function used to measure the discrepancy between the predicted state $\hat{O}_t$ and the actual current state O_t is commonly the Mean Squared Error (MSE):

$$\mathcal{L}_{\text{SS}} = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{d}\left(\hat{O}_{t_i}^{(j)} - O_{t_i}^{(j)}\right)^2, \quad (2)$$

where N represents the total number of samples, d represents the dimensionality of each state vector, $\hat{O}_{t_i}^{(j)}$ denotes the $j-th$ component of the predicted state vector $\hat{O}_{t_i}$ for the $i-th$ sample, $O_{t_i}^{(j)}$ denotes the $j-th$ component of the actual state vector O_{t_i} for the $i-th$ sample. **Forward Model** is designed to predict the trend in joint angle changes for the right front hip joint. Although the model accurately captures the overall trend, there remains a discrepancy between the predicted values and the true values. This discrepancy serves as more robust features for **Error Encoder** to output error latent.

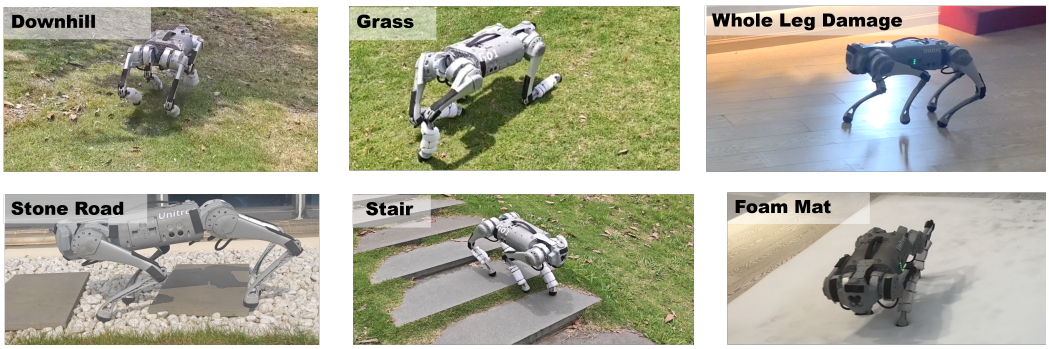


Figure 2: Snapshots of the quadruped robot's locomotion across different terrains and under varying joint damage conditions.

To extract frequency domain information from a time series H_{t-1}, we first apply the FFT to the time series data to convert it from the time domain to the frequency domain. This yields the frequency domain result $\mathbf{X}$.

$$\mathbf{X} = \mathcal{F}(H_{t-1}) \tag{3}$$

The amplitude is the magnitude of the complex FFT result, and the phase is the angle: $A = |\mathbf{X}|, \phi = \arg(\mathbf{X})$. Combine the reshaped A and ϕ into a single feature vector by concatenating them along the last dimension, resulting in the input features for **Frequency Encoder**.

3.4 Overall Optimization

A mini-batch generator is utilized to iterate over training data in batches. Each batch contains observations, actions, rewards, and other relevant information. The training loop performs optimization by computing and minimizing several loss components simultaneously. The surrogate loss $\mathcal{L}_{\text{sur}}$ is calculated to optimize the policy, and the value loss $\mathcal{L}_{\text{value}}$ measures the discrepancy between the predicted values and the target values. The entropy loss $\mathcal{L}_{\text{entropy}}$ encourages exploration by penalizing low entropy in the policy distribution. Similar to DreamWaq [36], VAE loss $\mathcal{L}_{\text{vae}}$ are also included, such as reconstruction loss, velocity loss, and KLD loss. The total loss is the sum of these components, and the model parameters are updated using their gradients:

$$\mathcal{L}_{\text{total}} = \lambda_s \cdot \mathcal{L}_{\text{sur}} + \lambda_v \cdot \mathcal{L}_{\text{value}} - \lambda_e \cdot \mathcal{L}_{\text{entropy}} + \lambda_{vae} \cdot \mathcal{L}_{\text{vae}} + \lambda_{cl} \cdot \mathcal{L}_{CL} + \lambda_{ss} \cdot \mathcal{L}_{SS}, \tag{4}$$

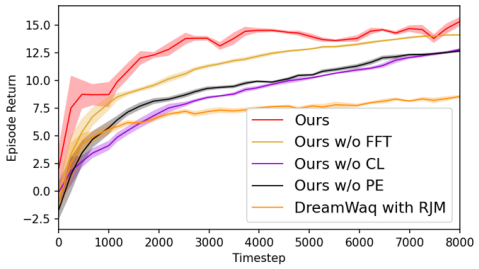
where λ is the loss function weight. Finally, gradient clipping is applied to stabilize the training.

4 Experimental Results

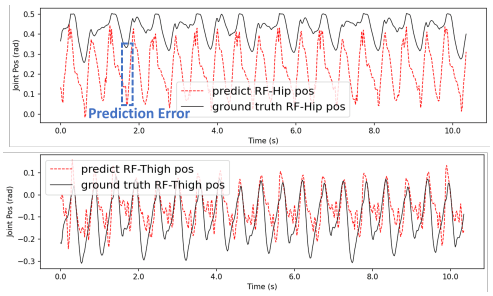
In the experimental section, the proposed learning framework is implemented on quadruped and hexapod robots. Firstly, the framework's ability to generalize across diverse terrains under different joint damages is also evaluated. Secondly, the zero-shot transfer capability is also validated by evaluating whether the framework can effectively handle locked joints without prior training. Finally, the generalization of the framework is examined by deploying it across different robot topologies to confirm its versatility and applicability to various robotic platforms.

4.1 Evaluation of effectiveness across diverse terrains

The effectiveness and adaptability of the proposed fault-tolerant controller are clearly demonstrated by its successful deployment across a diverse set of terrains, as illustrated in Figure 2. This versatility underscores the controller's ability to manage various types of joint damage, ensuring that robots can maintain stable and efficient movement in challenging and dynamic environments. The terrains used for the evaluation include stone roads, foam mats, grass, and gravel. Detailed experimental settings and additional results can be found in the supplementary video. These terrains encompass both rigid

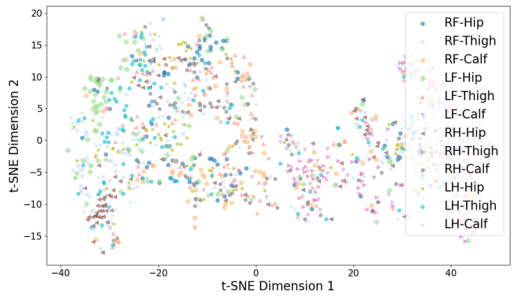


(a) Training performance with ablation (CL: contrastive learning, PE: prediction error, RJM: random joint mask).

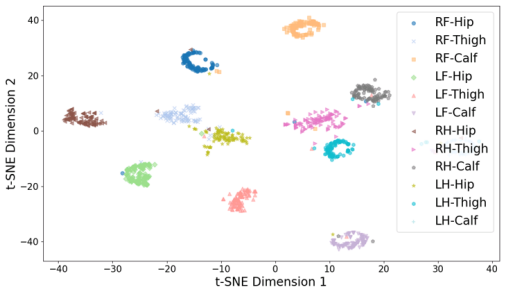


(b) Prediction error between the predicted and observed values.

Figure 3: Training performance and prediction error results.



(a) DreamWaq with RJM.



(b) Our method.

Figure 4: The t-SNE comparison results between the baseline method (DreamWaq with RJM) and our method.

surfaces, such as stone roads and gravel, as well as non-rigid, softer surfaces such as foam mats and grass, which introduce additional challenges in terms of traction, stability, and the absorption of impact forces.

Figure 3**a** presents the results of the comparison and ablation experiments. Under the same random joint mask training processing, the proposed training framework significantly outperforms DreamWaq [36]. The ablation experiment results demonstrate the influence of various components on the method's training performance. Specifically, the exclusion of contrastive learning and prediction errors leads to a marked deterioration in the method's performance. Without using prediction errors, specifically it involves providing the policy network with the direct outputs of the prediction of the forward model $\hat{O}_t$ as features, without subtracting $\hat{O}_t$ from the actual observation.

To further elaborate on the prediction performance of the proposed **Forward Model**, the prediction result is illustrated in Figure 3**b**. The **Forward Model** accurately captures the trend in joint angle changes for the right front thigh joint when its hip joint is damaged. However, due to damage, the hip joint predictions are not accurate, resulting in some prediction errors. This error serves as a valuable source of information, providing richer features for the **Error Encoder** to enhance its learning and fault-tolerant control capabilities.

Figures 4**(a,b)** describe the t-SNE analysis of DreamWaq and our method, respectively, illustrating how **Adaptation Encoder** represents the dynamics of different joint damages. Our method can clearly distinguish the dynamics of different joint damages in the latent space, enabling the policy to better adapt to various damage scenarios and allowing for more precise identification of specific faults. Contrastive learning not only improves the network's ability to represent fault-tolerant policy, but it can also be extended to improve the representation of visual input in robot learning [32, 33].

Table 1: Velocity tracking error under Zero Torque and Lock Joint damages

Joints	Zero Torque		Lock Joint	
	Dreamwaq with RJM	Ours	Dreamwaq with RJM	Ours
RF-Hip	0.337 ± 0.014	0.149 ± 0.054	0.346 ± 0.024	0.106 ± 0.084
RF-Thigh	0.436 ± 0.008	0.128 ± 0.024	0.469 ± 0.008	**0.105 ± 0.028**
RF-Calf	0.451 ± 0.008	0.172 ± 0.025	0.445 ± 0.018	0.241 ± 0.045
LF-Hip	0.390 ± 0.011	0.185 ± 0.032	0.389 ± 0.011	0.135 ± 0.041
LF-Thigh	0.354 ± 0.017	0.155 ± 0.022	0.616 ± 0.017	0.120 ± 0.027
LF-Calf	0.315 ± 0.014	0.256 ± 0.020	0.495 ± 0.034	0.256 ± 0.040
RH-Hip	0.481 ± 0.027	0.141 ± 0.028	0.423 ± 0.017	0.109 ± 0.038
RH-Thigh	0.368 ± 0.018	**0.104 ± 0.029**	0.352 ± 0.009	0.118 ± 0.039
RH-Calf	0.442 ± 0.015	0.209 ± 0.038	0.459 ± 0.019	0.290 ± 0.028
LH-Hip	0.321 ± 0.014	0.128 ± 0.020	0.362 ± 0.010	0.178 ± 0.050
LH-Thigh	0.350 ± 0.015	0.129 ± 0.031	0.489 ± 0.007	0.146 ± 0.041
LH-Calf	0.512 ± 0.010	0.213 ± 0.023	0.639 ± 0.010	0.314 ± 0.033

4.2 Zero-shot capability for locked joints

To further validate the adaptability of our method, we consider scenarios when the robot's joints are locked. Our method exhibits zero-transfer capability, seamlessly adapting from "Zero Torque" to "Lock Joint" damage types without requiring additional adjustments or retraining.

To validate the improvement of the proposed method in zero-shot transfer, we compare the velocity tracking errors under various joint damages. Table 1 shows the velocity tracking accuracy of DreamWaq with RJM compared to our proposed method for the type of zero-torque damage. The results demonstrate that our method significantly improves the accuracy of velocity tracking in various types of joint damage. Compared to other joints, damage to the calf joint has the most significant impact on robot velocity tracking. Table 1 shows that, with the type of joint lock damage, our method still achieves a higher velocity tracking accuracy. The tracking error decreased by 41.7% and 61.5% for the two types of damage, respectively.

4.3 Generalization on different robot topologies

To further validate the generalization of our method, we implemented our method on a custom-built hexapod robot, which presents an opportunity to test the method under more complex conditions due to the increased degrees of freedom (DOF) of the robot.

As shown in Figure 5, on flat terrain, the robot can maintain stable walking despite damage to two middle leg's thigh joints. However, real-world deployment brings additional challenges, primarily due to the sim-to-real gap [37], which can introduce discrepancies in the robot's performance due to factors such as sensor noise, actuator limitations, and environmental variability. These challenges increase the difficulty of implementing fault-tolerant control in the real world. Nonetheless, our method demonstrated strong robustness, enabling the real hexapod robot to maintain stable locomotion even in the presence of joint damage, thus highlighting the practical applicability and effectiveness of our fault-tolerant controller in real-world conditions.

Figure 6 shows the joint torques of the middle two legs and the thigh joint action tracking results. The results indicate that during locomotion, the output torque of the damaged thigh joints remains at zero, leading to noticeable torque fluctuations in the calf joints, likely due to ground impact. In contrast, the torque of the hip joints is less significantly affected. When the damaged thigh joint cannot produce any torque, its position becomes dependent on the other joints. The fault-tolerant controller compensates by using the motion of the other joints to stabilize and facilitate the movement of the damaged joint.

Figure 5: Snapshots of real hexapod robot experiments on flat and grass terrain.

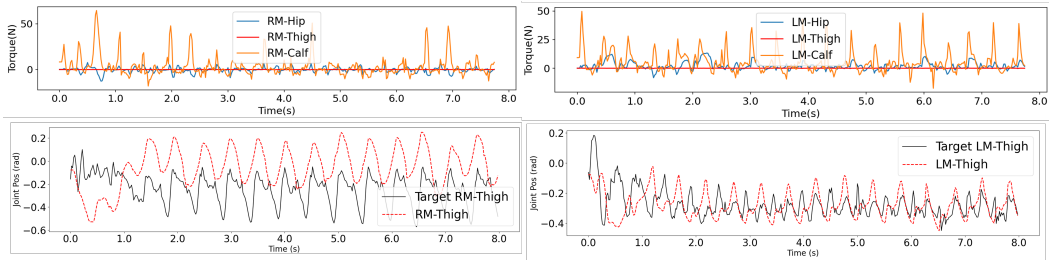


Figure 6: Torque and joint position variation under zero torque damage.

4.4 Generalization on different robot topologies

5 Limitation

The control policy relies primarily on proprioceptive inputs such as joint positions and velocities, which may restrict its perception and adaptability in complex or visually cluttered environments. Without exteroceptive sensing, such as vision or tactile feedback, the robot lacks awareness of obstacles, terrain transitions, or upcoming hazards, limiting its ability to proactively plan or adjust its gait [38]. Future work could explore the integration of visual and other sensor modalities, such as depth cameras or force sensors, to enable richer environmental perception and further enhance locomotion stability, terrain adaptability, and recovery capabilities.

6 Conclusion

This work presents a reinforcement learning-based framework to fault-tolerant control for legged robots, focusing on enhancing adaptability and robustness under joint damage conditions. By incorporating contrastive learning and prediction-error feedback into the policy architecture, the proposed method enables dynamic adjustment of control actions, allowing the robot to maintain stable locomotion across a variety of terrains and fault scenarios. Comprehensive evaluations on both quadruped and hexapod platforms demonstrate strong generalization across robot morphologies and effective zero-shot adaptation to previously unseen damage patterns. These results indicate the method provides a practical and scalable solution for fault-tolerant locomotion in complex and unpredictable environments.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Grant No. 62373242 and No. 92248303), the Shanghai Municipal Science and Technology Major Project (Grant No. 2021SHZDZX0102), and the Fundamental Research Funds for the Central Universities.

References

[1] 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*, 4(26):eaau5872, 2019.

[2] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.

[3] A. Elhadad, Y. Gao, and S. Choi. Revolutionizing aquatic robotics: Advanced biomimetic strategies for self-powered mobility across water surfaces. *Advanced Materials Technologies*, page 2400426, 2024.

[4] T. J. Buchner, T. Fukushima, A. Kazemipour, S.-D. Gravert, M. Prairie, P. Romanescu, P. Arm, Y. Zhang, X. Wang, S. L. Zhang, et al. Electrohydraulic musculoskeletal robotic leg for agile, adaptive, yet energy-efficient locomotion. *Nature Communications*, 15(1):7634, 2024.

[5] X. Wu, W. Dong, H. Lai, Y. Yu, and Y. Wen. Adaptive control strategy for quadruped robots in actuator degradation scenarios. In *Proceedings of the Fifth International Conference on Distributed Artificial Intelligence*, pages 1–13, 2023.

[6] S. Gu, F. Meng, B. Liu, Z. Zhang, N. Sun, and M. Wang. Stability control of quadruped robot based on active state adjustment. *Biomimetics*, 8(1):112, 2023.

[7] Y. Gao, B. Su, L. Jiang, and F. Gao. Multi-legged robots: progress and challenges. *National Science Review*, 10(5):nwac214, 2023.

[8] Y. Liu, X. Fan, L. Ding, J. Wang, T. Liu, and H. Gao. Fault-tolerant tripod gait planning and verification of a hexapod robot. *Applied Sciences*, 10(8):2959, 2020.

[9] 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*, 178:105069, 2022.

[10] M. Kim, U. Shin, and J.-Y. Kim. Learning quadrupedal locomotion with impaired joints using random joint masking. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9751–9757. IEEE, 2024.

[11] 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*, 8(4):2991–3003, 2022.

[12] 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.

[13] C. Rajani, K. Arndt, D. Blanco-Mulero, K. S. Luck, and V. Kyrki. Co-imitation: learning design and behaviour by imitation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 6200–6208, 2023.

[14] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503–507, 2015.

[15] Y. Yang, K. Caluwaerts, A. Iscen, T. Zhang, J. Tan, and V. Sindhwani. Data efficient reinforcement learning for legged robots. In *Conference on Robot Learning*, pages 1–10. PMLR, 2020.

[16] K. Weerakoon, A. J. Sathyamoorthy, M. Elnoor, and D. Manocha. Vapor: Legged robot navigation in unstructured outdoor environments using offline reinforcement learning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10344–10350. IEEE, 2024.

[17] G. Ji, J. Mun, H. Kim, and J. Hwangbo. Concurrent training of a control policy and a state estimator for dynamic and robust legged locomotion. *IEEE Robotics and Automation Letters*, 7(2):4630–4637, 2022.

[18] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3803–3810. IEEE, 2018.

[19] C. Zhang, N. Rudin, D. Hoeller, and M. Hutter. Learning agile locomotion on risky terrains. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11864–11871. IEEE, 2024.

[20] Z. Zhang, Q. Wei, X. Chang, L. Lang, H. Ma, and H. An. Learning agile, robust locomotion skills for quadruped robot. In *2022 International Conference on Advanced Robotics and Mechatronics (ICARM)*, pages 1011–1016. IEEE, 2022.

[21] V. Atanassov, J. Ding, J. Kober, I. Havoutis, and C. Della Santina. Curriculum-based reinforcement learning for quadrupedal jumping: A reference-free design. *IEEE Robotics & Automation Magazine*, 2024.

[22] T. Kobayashi and T. Sugino. Reinforcement learning for quadrupedal locomotion with design of continual–hierarchical curriculum. *Engineering Applications of Artificial Intelligence*, 95: 103869, 2020.

[23] X. Song, Y. Yang, K. Choromanski, K. Caluwaerts, W. Gao, C. Finn, and J. Tan. Rapidly adaptable legged robots via evolutionary meta-learning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3769–3776. IEEE, 2020.

[24] B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull. Active domain randomization. In *Conference on Robot Learning*, pages 1162–1176. PMLR, 2020.

[25] V. Tsounis, M. Alge, J. Lee, F. Farshidian, and M. Hutter. Deepgait: Planning and control of quadrupedal gaits using deep reinforcement learning. *IEEE Robotics and Automation Letters*, 5(2):3699–3706, 2020.

[26] T. Hou, J. Tu, X. Gao, Z. Dong, P. Zhai, and L. Zhang. Multi-task learning of active fault-tolerant controller for leg failures in quadruped robots. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9758–9764. IEEE, 2024.

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

[28] F. Yang, C. Yang, D. Guo, H. Liu, and F. Sun. Fault-aware robust control via adversarial reinforcement learning. In *2021 IEEE 11th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, pages 109–115. IEEE, 2021.

[29] Y. Qiu, X. Lin, J. Wang, X. Li, L. Qi, and M.-H. Yang. Umc: Unified resilient controller for legged robots with joint malfunctions. *arXiv preprint arXiv:2502.03035*, 2025.

[30] X. B. Peng, E. Coumans, T. Zhang, T.-W. Lee, J. Tan, and S. Levine. Learning agile robotic locomotion skills by imitating animals. *arXiv preprint arXiv:2004.00784*, 2020.

[31] A. Miller, S. Fahmi, M. Chignoli, and S. Kim. Reinforcement learning for legged robots: Motion imitation from model-based optimal control. *arXiv preprint arXiv:2305.10989*, 2023.

[32] R. Zheng, X. Wang, Y. Sun, S. Ma, J. Zhao, H. Xu, H. Daumé III, and F. Huang. Taco: Temporal latent action-driven contrastive loss for visual reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[33] Y. Ze, N. Hansen, Y. Chen, M. Jain, and X. Wang. Visual reinforcement learning with self-supervised 3d representations. *IEEE Robotics and Automation Letters*, 8(5):2890–2897, 2023.

[34] M. Okada and T. Taniguchi. Dreamingv2: Reinforcement learning with discrete world models without reconstruction. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 985–991. IEEE, 2022.

[35] T. Nguyen, T. M. Luu, T. Vu, and C. D. Yoo. Sample-efficient reinforcement learning representation learning with curiosity contrastive forward dynamics model. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3471–3477. IEEE, 2021.

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

[37] Y. Gao, Y. Fu, and M. Sun. Sim-to-real hierarchical planning and control system for six-legged robot. In *CAAI International Conference on Artificial Intelligence*, pages 621–625. Springer, 2022.

[38] H. Lin, H. Li, and Y. Gao. See-touch-predict: Active exploration and online perception of terrain physics with legged robots. *IEEE Robotics and Automation Letters*, 2025.

A Network design and training details

The training and inference details of our learning framework are illustrated in Figure 7. Similarly to DreamWaq, the **Prediction Encoder** predicts the current velocity $\hat{v}_t$ and outputs the latent variable z_2. Finally, the **Policy Network** generates action a_t based on current observation O_t, estimated velocity $\hat{v}_t$, and latent variables z_2, z_3, and z_4.

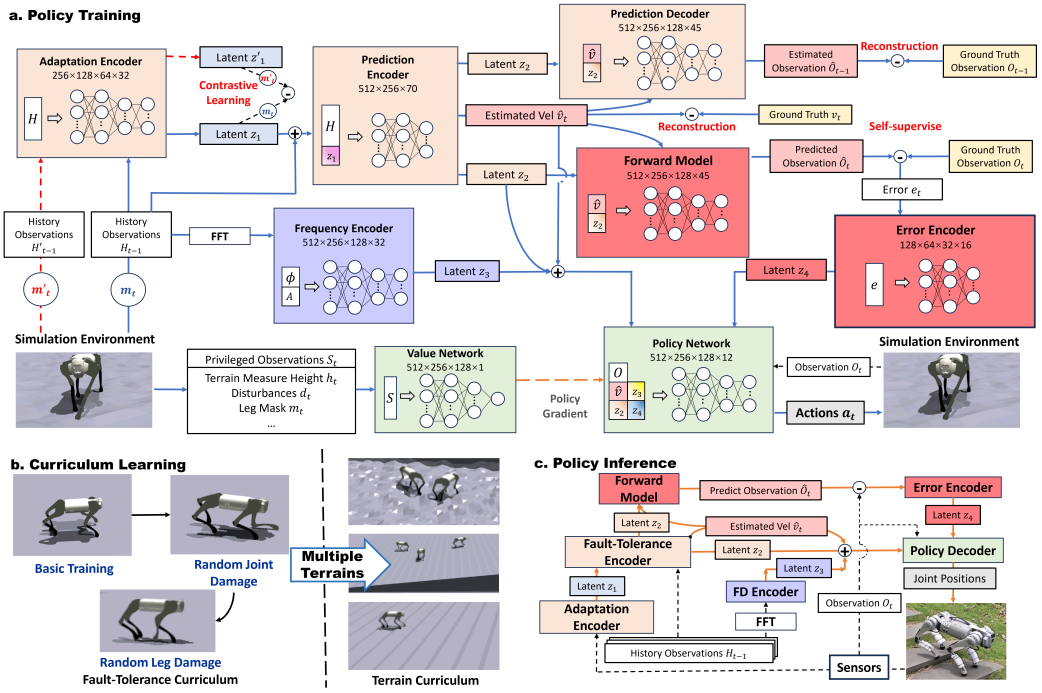


Figure 7: Training and inference details. **a**, The specific network design and parameters used in the training process. **b**, Curriculum learning during the training process. **c**, Network deployed to the real robot in the inference process.

A.1 Reward function design

In the reward function, no prior information is incorporated about leg damage models is incorporated; instead, the model's representation capability is relied upon to learn fault-tolerant control. The specific reward functions are as follows:

- **Reward of tracking linear velocity:**

$$R_{\text{tracking\_lin}} = \exp\left(-\frac{1}{\sigma^2_{\text{tracking}}}\left\|\mathbf{c}-\mathbf{v}\right\|^2\right), \tag{5}$$

  where $\mathbf{c}$ represents the linear velocity command vector, $\mathbf{v}$ represents the actual linear velocity vector, and σ_{tracking} is the scaling factor. The reward function penalizes the difference between the commanded and actual linear velocities. A smaller error results in a higher reward, which encourages the model to closely follow the commanded velocities. In this work, $\sigma_{\text{tracking}} = 0.25$.

- **Reward of tracking angular velocity:**

$$R_{\text{tracking\_ang}} = \exp\left(-\frac{(\omega_{\text{cmd}}-\omega)^2}{\sigma_{\text{tracking}}}\right), \tag{6}$$

  where ω_{cmd} is the angular velocity command, and ω is the actual angular velocity. This reward function penalizes the error between the commanded and actual angular velocities.

Similar to linear velocity tracking, the closer the actual angular velocity is to the commanded value, the higher the reward.

- **Reward for penalizing the z direction velocity:**

$$R_{\text{lin\_vel\_z}} = (v_z)^2 \tag{7}$$

where v_z is the velocity in the z direction. This reward penalizes any linear velocity along the z axis, which is typically undesirable as it may indicate instability or unintended movement in that direction.

- **Reward for penalizing the xy direction angular velocity:**

$$R_{\text{ang\_vel\_xy}} = \omega_x^2 + \omega_y^2 \tag{8}$$

where ω_x and ω_y are the angular velocities in the x and y directions, respectively. This function penalizes any angular velocity in the x and y directions, encouraging the model to maintain stability and avoid unnecessary rotation.

- **Reward for penalizing the torque:**

$$R_{\text{torque}} = \tau^2 \tag{9}$$

where τ represents the torque applied to the joints. This reward penalizes high torques, which can lead to excessive energy consumption and wear on the robot's components. The goal is to encourage smoother and more energy-efficient movements.

- **Reward for penalizing joint accelerations:**

$$R_{\text{dof\_acc}} = \sum \left(\frac{\dot{q}_t - \dot{q}_{t-1}}{\Delta t} \right)^2 \tag{10}$$

where $\dot{q}_{t-1}$ is the joint velocity at the previous timestep, and Δt is the timestep interval. This function penalizes large accelerations of the joints, promoting smoother transitions in joint velocities, and contributing to the robot's motion's overall stability and smoothness.

- **Reward for penalizing changes in actions:**

$$R_{\text{action\_rate}} = \sum \left(\mathbf{a}^{\text{last}} - \mathbf{a} \right)^2 \tag{11}$$

where $\mathbf{a}$ and $\mathbf{a}^{\text{last}}$ represent the current and previous actions, respectively. This reward penalizes rapid changes in actions between consecutive time steps, encouraging smoother and more consistent control signals.

- **Reward for penalizing hip joint position:**

$$R_{\text{hip\_pos}} = \sum_j \left(\mathbf{q}_{\text{hip},j} - \mathbf{q}_{\text{default},j} \right)^2 \tag{12}$$

where $\mathbf{q}_{\text{hip},j}$ and $\mathbf{q}_{\text{default},j}$ represent the current and default positions of the hip joint j, respectively. This reward penalizes deviations from the default or desired hip joint positions, helping to maintain a stable and balanced posture.

- **Reward of action smoothness:**

$$R_{\text{smooth}} = \sum_i \left[\left(\mathbf{q}_{t,i}^{\text{target}} - 2\mathbf{q}_{t-1,i}^{\text{target}} + \mathbf{q}_{t-2,i}^{\text{target}} \right)^2 \cdot (\mathbf{a}_{t-1,i} \neq 0) \cdot (\mathbf{a}_{t-2,i} \neq 0) \right] \tag{13}$$

where $\mathbf{q}_{t,i}^{\text{target}}$ is the target position at time t for joint i, and $\mathbf{a}_{t-1,i}$ and $\mathbf{a}_{t-2,i}$ represent the actions at previous time steps. This reward penalizes abrupt changes in the joint target positions over time, which promotes smoother and more natural joint movements.

- **Reward of gait:**

$$R_{\text{gait}} = \|\mathbf{a}_{1:3} - \mathbf{a}_{7:9}\|^2 + \|\mathbf{a}_{1:3} - \mathbf{a}_{13:15}\|^2 + \|\mathbf{a}_{4:6} - \mathbf{a}_{10:12}\|^2 \tag{14}$$

$$+ \|\mathbf{a}_{4:6} - \mathbf{a}_{16:18}\|^2 + \|\mathbf{a}_{10:12} - \mathbf{a}_{16:18}\|^2 + \|\mathbf{a}_{7:9} - \mathbf{a}_{13:15}\|^2 \tag{15}$$

This reward function, R_{gait}, is utilized to encourage the 3-3 gait pattern, where the robot's legs move in coordinated pairs. The reward penalizes deviations from synchronized movement patterns across the legs, promoting a stable and efficient gait.

The reward function weights for training the quadruped and hexapod robots are shown in Table· A.1.

Reward Function	Quadruped Robot	Hexapod Robot
$R_{\text{tracking\_lin}}$	1.0	1.0
$R_{\text{tracking\_ang}}$	0.5	0.5
$R_{\text{lin\_vel\_z}}$	-2.0	-2.0
$R_{\text{ang\_vel\_xy}}$	-0.05	-0.05
R_{ori}	-1.0	-1.0
R_{torque}	$-1e-5$	$-1e-4$
$R_{\text{dof\_acc}}$	$-2.5e-7$	$-2.5e-7$
$R_{\text{action\_rate}}$	-0.05	-0.05
$R_{\text{hip\_pos}}$	-0.1	-0.1
R_{smooth}	-0.01	-0.05
R_{gait}	0	-0.02

Table 2: Reward Function Weights

A.2 Hyperparameters of training

These hyperparameters are used to configure the training process of the reinforcement learning algorithm and the adaptation module. The num_learning_epochs and num_mini_batches define the number of training epochs and mini-batches per policy update. The learning_rate and adaptation_module_learning_rate control the step sizes for updating model and adaptation module parameters. γ is used to discount future rewards and smoothing advantage estimation, respectively. desired_kl helps in maintaining the difference between old and new policies within a target range. max_grad_norm prevents gradient explosion by clipping gradients. The specific hyperparameter values are shown in Table 3.

Hyperparameter	Value
Number of learning epochs	5
Number of mini batches	4
Learning rate	1×10^{-3}
Adaptation module learning rate	1×10^{-3}
γ	0.99
λ_s	1.0
λ_v	1.0
λ_e	0.01
λ_{vae}	1.0
λ_{cl}	1.0
λ_{ss}	1.0
Desired KL divergence	0.02
Maximum gradient norm	1.0

Table 3: Hyperparameters used in the training process

A.3 Domain randomization

In our domain randomization setup for legged robots, several parameters are adjusted to enhance the robustness and generalization of the policy:

- `rand_interval_s`: The interval in seconds for randomizing parameters.
- `friction_range`: Randomizes the surface friction, making the terrain more or less slippery.
- `restitution_range`: Changes the bounciness of contact surfaces, affecting how the robot interacts with the ground.
- `added_mass_range`: The range for added mass variability.

- `com_displacement_range`: The range for center of mass displacement.
- `randomize_motor_strength`: Varies the strength of the motors to simulate different motor performance characteristics.
- `lag_timesteps`: The number of timesteps to introduce as delay.
- `randomize_Kp_factor`: Randomizes the proportional gain (`Kp`) of the PD controller.
- `Kp_factor_range`: The range for `Kp` variability.
- `randomize_Kd_factor`: Randomizes the derivative gain (`Kd`) of the PD controller.
- `Kd_factor_range`: The range for `Kd` variability.

Parameter	Value
Randomization interval (s)	15
Friction range	[0.3, 3.0]
Restitution range	[0.0, 0.4]
Added mass range	[-1.0, 2.0]
COM displacement range	[-0.15, 0.15]
Motor strength range	[0.9, 1.1]
Lag timesteps	6
Kp factor range	[0.9, 1.1]
Kd factor range	[0.9, 1.1]

Table 4: Domain randomization parameters

A.4 More Comparisons

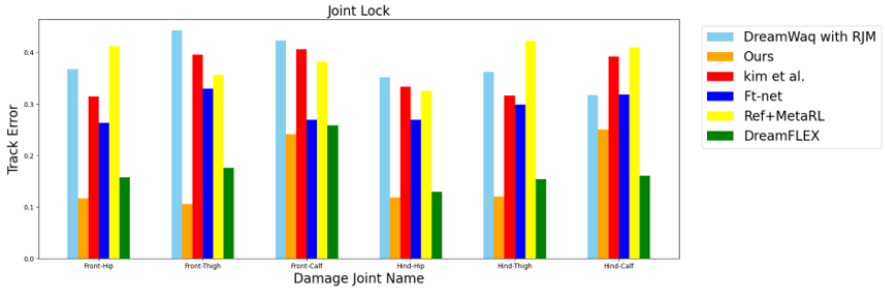


Figure 8: Comparison under Joint Lock Conditions.

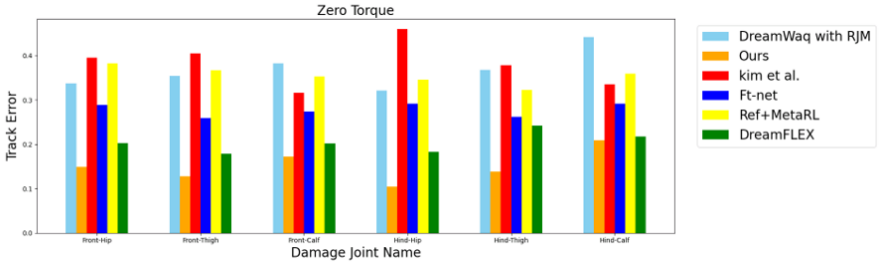


Figure 9: Comparison under Zero Torque Conditions.

A.5 Terrain training

In our terrain randomization setup for legged robots, various terrain types are defined with specific parameters to enhance the robustness and adaptability of the policy:

- `plane_terrain`: `weight` = 1.0, `height` = 0.0. This represents a flat terrain with a specified height.

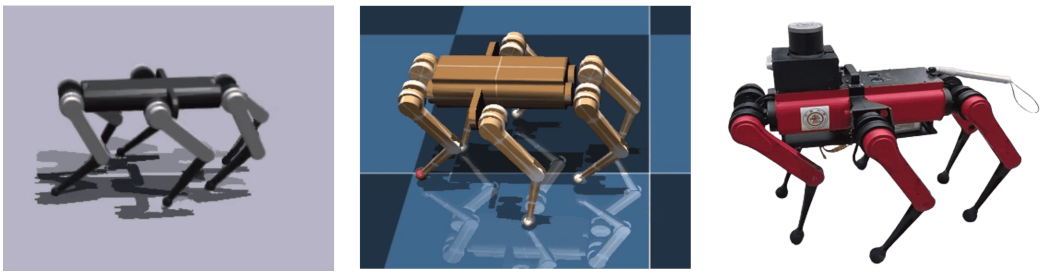


Figure 10: Custom-built hexapod robot in Isaac Gym, MuJoCo, and the real world.

- `random_uniform_terrain`: `weight` = 3.0, `min_height` = -0.12, `max_height` = 0.12, `step` = 0.01, `downsampled_scale` = 0.15. This represents a terrain with random uniform height variations within the specified range.
- `sloped_terrain`: `weight` = 1.0, `slope` = 0.5. This represents a sloped terrain with a specified slope angle.
- `pyramid_sloped_terrain`: `weight` = 3.0, `slope` = -0.5, `platform_size` = 1.5. This represents a pyramid-shaped sloped terrain with a specified slope and platform size.
- `wave_terrain`: `weight` = 2.0, `num_waves` = 4, `amplitude` = 0.05. This represents a wavy terrain with a specified number of waves and amplitude.
- `stairs_terrain`: `weight` = 1.0, `step_height` = 0.05, `step_width` = 0.5. This represents a terrain with stairs having specified step height and width.
- `pyramid_stairs_terrain`: `weight` = 3.0, `step_width` = 0.4, `step_height` = -0.05, `platform_size` = 3.0. This represents a pyramid-shaped stairs terrain with specified step width, step height, and platform size.
- `stepping_stones_terrain`: `weight` = 1.0, `stone_size` = 1.5, `stone_distance` = 0.1, `max_height` = 0.0, `platform_size` = 4.0, `depth` = -10. This represents a terrain with stepping stones of specified size, distance, height, platform size, and depth.
- `discrete_obstacles_terrain`: `weight` = 1.0, `max_height` = 0.2, `min_size` = 1.0, `max_size` = 2.0, `num_rects` = 20, `platform_size` = 1.0. This represents a terrain with discrete obstacles of specified size, height, and platform size.

B Hexapod experiment

In this section, we provide a detailed introduction to the hexapod robot used in our experiments, including the hardware specifications and some parameter tuning details. The appearance of the robot is shown in Figure 10. Although the real robot is equipped with a LiDAR, it was not used in the experiments.

B.1 Hexapod hardware

Table 5 provides a comprehensive overview of the hardware specifications for the hexapod robot used in our experiments. Each row represents a different component of the robot, detailing its position, mass, inertia, and joint range. The base of the robot is positioned at (0, 0, 0.4) meters and has a mass of 18 kg. Each leg consists of a hip, leg, and foot, with their respective positions, masses, and inertias specified. All joints have a range of motion from -3.14 to 3.14 radians. This detailed breakdown helps in understanding the physical configuration and mechanical properties of the robot, which are crucial for both simulation and real-world applications.

Body Part	Position (pos)	Mass (kg)	Inertia (kg·m^2)
Base	(0, 0, 0.4)	18.0	(0.05507, 0.30204, 0.34116)
RB Hip	(-0.33, -0.053, 0)	0.601	(0.00031, 0.000574, 0.00031)
RB Leg	(0, 0, 0)	0.798	(0.00212, 0.002277, 0.000497)
RB Foot	(0, -0.08025, -0.249)	0.390	(0.00033, 0.00033, 0.00000715)
LB Hip	(-0.33, 0.053, 0)	0.601	(0.00031, 0.000574, 0.00031)
LB Leg	(0, 0, 0)	0.798	(0.00212, 0.002277, 0.000497)
LB Foot	(0, 0.08025, -0.249)	0.390	(0.00033, 0.00033, 0.00000715)
LM Hip	(0, 0.19025, 0)	0.601	(0.00031, 0.000574, 0.00031)
LM Leg	(0, 0, 0)	0.798	(0.00212, 0.002277, 0.000497)
LM Foot	(0, 0.08025, -0.249)	0.390	(0.00033, 0.00033, 0.00000715)
LF Hip	(0.33, 0.053, 0)	0.601	(0.00031, 0.000574, 0.00031)
LF Leg	(0, 0, 0)	0.798	(0.00212, 0.002277, 0.000497)
LF Foot	(0, 0.08025, -0.249)	0.390	(0.00033, 0.00033, 0.00000715)
RF Hip	(0.33, -0.053, 0)	0.601	(0.00031, 0.000574, 0.00031)
RF Leg	(0, 0, 0)	0.798	(0.00212, 0.002277, 0.000497)
RF Foot	(0, -0.08025, -0.249)	0.390	(0.00033, 0.00033, 0.00000715)
RM Hip	(0, -0.19025, 0)	0.601	(0.00031, 0.000574, 0.00031)
RM Leg	(0, 0, 0)	0.798	(0.00212, 0.002277, 0.000497)
RM Foot	(0, -0.08025, -0.249)	0.390	(0.00033, 0.00033, 0.00000715)

Table 5: Hexapod Robot Hardware Specifications

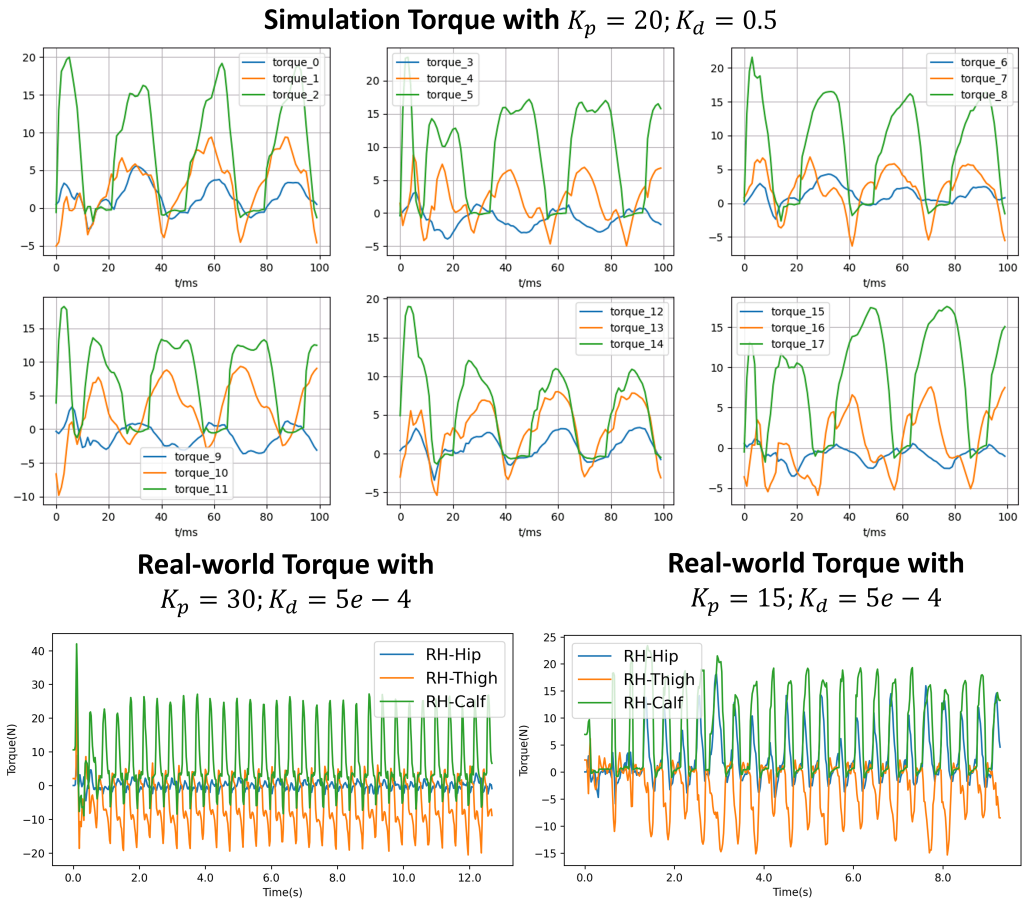


Figure 11: Sim-to-real result of joint torque with different K_p and K_d parameters.

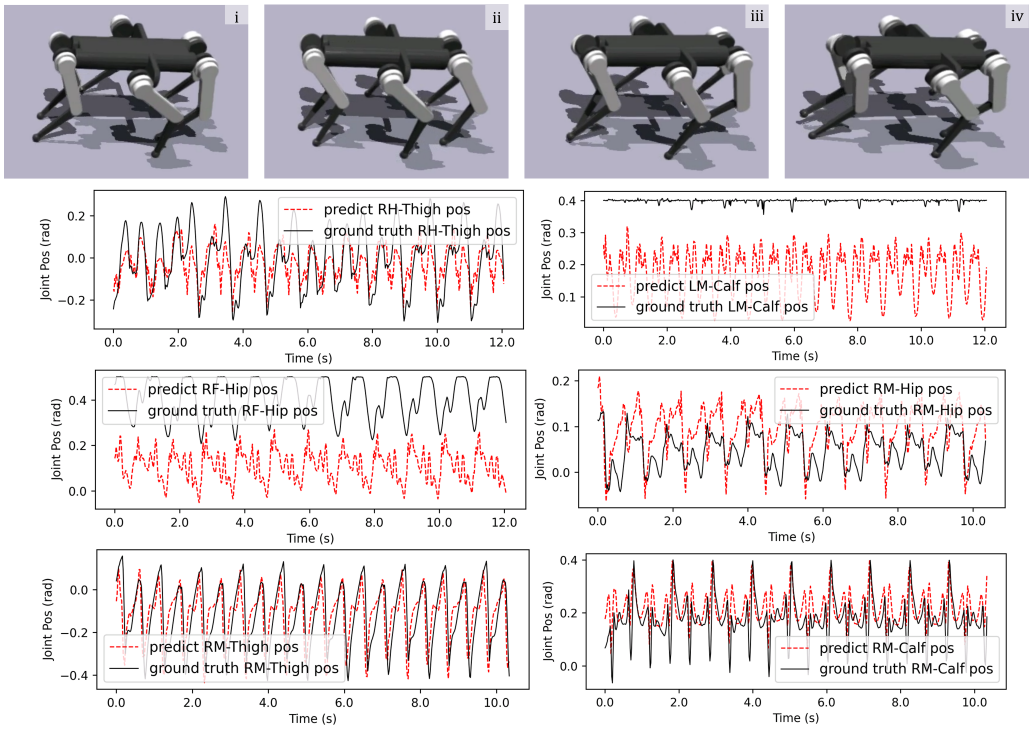


Figure 12: Custom-built hexapod robot in Isaac Gym, MuJoCo, and the real world.

B.2 Sim-to-real transfer for hexapod robot

In the process of sim-to-real transfer for a hexapod robot using reinforcement learning strategies, tuning the motor parameters K_p (proportional gain) and K_d (derivative gain) significantly affects the robot's performance. This tuning is crucial because it influences how the robot's joints respond to control signals, impacting stability, responsiveness, and overall movement quality. In the training environment, the motor parameters are initially set with $K_p = 20$ and $K_d = 0.5$. These settings result in joint output torques stabilizing around 20 N, providing a balance between responsiveness and stability that is well-suited for the simulated conditions. The proportional gain K_p controls how forcefully the motors react to positional errors, while the derivative gain K_d helps dampen oscillations by responding to the rate of change of the error. However, when these settings are transferred directly to the real robot, discrepancies often arise due to differences between the simulated model and the actual hardware. As shown in Figure 11, if K_p is increased to 30 in the real robot, the output torque correspondingly increases to approximately 30 N. This higher torque can lead to more aggressive and less controlled movements, as the robot reacts more forcefully to positional errors. This discrepancy results in a mismatch in performance between the simulation and the real world, where the robot may exhibit instability or overshoot desired positions. In contrast, when the real robot's K_p is lowered to 15, the output torque decreases, and the robot's movements become less aggressive and more controlled. In this scenario, the real robot exhibits a movement behavior that closely mirrors the simulated environment. The reduced proportional gain means the motors respond less forcefully to errors, resulting in smoother and more stable movements, which aligns better with the conditions the reinforcement learning model was trained under.

B.3 Forward prediction of hexapod robot

Figure 12 demonstrates that the **Forward Model** is also effective in the hexapod robot. We conducted a robustness evaluation by damaging three joints of the hexapod robot, specifically the RH-Thigh, LM-Calf, and RF-HIP. After training within our framework, the robot was able to adapt and

continue walking stably despite these significant impairments. This showcases the robustness and adaptability of the learning framework, as the robot could compensate for the damaged joints and maintain functional locomotion. As illustrated in Figure 12, the **Forward Model** performs best in forecasting the state of the damaged thigh joint, demonstrating the highest accuracy. Conversely, the model exhibits larger prediction errors for the damaged calf and hip joints. As with the quadruped robot, the **Forward Model** accurately forecasts the position changes of healthy joints.