A Residual Meta-Reinforcement Learning Method for Training Fault-Tolerant Policies for Quadruped Robots

Ci Chen
State Key Laboratory of Industrial Control and Technology
Zhejiang University
Hangzhou, China
chenci107@zju.edu.cn

Chao Li
DeepRobotics Company
Hangzhou, China
lichao@deeprobotics.cn

Rong Xiong
State Key Laboratory of Industrial Control and Technology
Zhejiang University
Hangzhou, China
rxiong@zju.edu.cn

Hongbo Gao
School of Information Science and Technology
University of Science and Technology of China
Hefei, China
ghb48@ustc.edu.cn

Yue Wang*
State Key Laboratory of Industrial Control and Technology
Zhejiang University
Hangzhou, China
wangyue@iipc.zju.edu.cn

2023 IEEE International Conference on Unmanned Systems (ICUS) | 979-8-3503-1630-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICUS58632.2023.10318421

***Abstract*—Motor locking is a common issue in quadruped robots that can have serious consequences if the robot continues executing its original commands. However, the static stability of the quadruped allows for the flexibility to adjust the robot's control policy so that it can maintain movement along a predetermined trajectory. In this paper, we introduce a residual meta reinforcement learning method comprising a trajectory generator and a meta-reinforcement learning corrector. The trajectory generator generates a reference joint position, while the corrector utilizes contextual reasoning to determine the appropriate action in the event of a motor locking. This action is employed to rectify the reference joint position, resulting in a fault-tolerant control strategy for the robot. We conducted comprehensive simulation experiments to validate our proposed algorithm, which demonstrates that the robot can still follow the predefined trajectory, even in the presence of a motor locking. Moreover, our proposed approach outperforms all baseline algorithms.**

Index Terms*—*Quadruped Robots*, *Reinforcement Learning*, *Fault Tolerance

I. Introduction

Quadruped robots are the most extensively utilized type of legged robots, renowned for their exceptional mobility in unstructured environments [1]. They hold the potential to replace humans in complex and extreme scenarios such as nuclear power plants and rescue sites. However, in extreme environments, it becomes challenging for humans to access the scene and perform repairs on the quadruped robot in the event of a malfunction. Hence, studying fault-tolerant control for quadruped robots becomes crucial under such circumstances [2]. One prevalent form of failure is motor locking, which can lead to unpredictable behavior if the quadruped robot continues following its initial instructions.

While there have been numerous studies on the motion control of quadruped robots, most of them have focused on enhancing their agility performance, with only a few examining fault-tolerant control. For instance, [3] and [4] propose a gait planning method that ensures the effectiveness of fault-tolerant gaits while avoiding deadlocks caused by kinematic constraints. Reference [5] suggests a method that involves adjusting the robot's center of mass position and combining gait coordination information to achieve fault tolerance performance. However, this method requires additional mechanisms. Additionally, [2] proposes a fault-tolerant technique that uses an equivalent geometric model to reconstruct the failed leg's workspace when a joint becomes locked. Furthermore, it introduces a non-linear approximation formula to optimize body posture and standing height, enabling compatibility with Whole-Body Control (WBC) for achieving steady and continuous forward movement. However, these methods entail a significant amount of expert knowledge and involve a laborious adjustment process. Moreover, different control strategies must be designed based on the specific motor damage situation.

The progress in reinforcement learning algorithms has enabled the development of robot fault-tolerant strategies utilizing these algorithms. For instance, approaches like [6]–[8] employ model-based reinforcement learning methods that utilize well-designed neural networks to predict the robot's next state. Subsequently, optimization algorithms (such as the random shooting method or cross-entropy optimization algorithm) are used to select the next action. However, these methods have only been validated on Ant tasks within Gym environments, and their applicability to the physical world

*Corresponding Author

This work was supported in part by the National Key R&D Program of China under Grant 2021ZD0114500, and in part by the National Natural Science Foundation of China under Grant U2013601.

remains uncertain.

In this paper, we introduce a residual meta-reinforcement learning technique that offers fault-tolerance policies for quadruped robots in motor locked situations. This approach combines a reference action generator and a corrector utilizing the meta-reinforcement learning method. The former offers a motion paradigm for the quadruped, while the latter captures locked joint information through the context to produce optimal strategies, which are used to correct the reference action in the joint space to obtain optimized joint positions. Our contributions include the following aspects:

(1) We propose a novel fault-tolerant strategy for quadruped robots in motor locked situations, which can offer optimized strategies specifically tailored to different types of motor locked scenarios.
(2) We conduct comprehensive experiments in a simulated environment, and the results of the experiments demonstrate that the proposed method surpasses the performance of the baseline algorithms.

II. Methodology

A. Problem Statement

The motion of quadrupeds can be described as a Markov Decision Process (MDP), typically expressed as the tuple $\langle S, A, r, p, \gamma \rangle$. Here, S represents the state space, A indicates the action space, r denotes the reward function, $p(\, s_{t+1} |\, s_t, a)$ signifies the transition function, and γ represents the reward discount factor. Within the framework of reinforcement learning, the RL algorithm aims to discover an optimal policy π^* so that for each state s_t, an optimal action a_t is chosen that maximizes the cumulative discounted rewards.

$$\pi^* = \arg\max \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \gamma^t r_t \right] \tag{1}$$

We utilize residual reinforcement learning to address the problem. Specifically, the action a_t consists of two components: the joint position derived from the reference action generator $\pi_G(a_t|s_t)$ and an additional component acquired through meta reinforcement learning $\pi_\theta(a_t|s_t, z)$.

$$a_t = \pi_G(a_t|s_t) + \pi_\theta(a_t|s_t, z) \tag{2}$$

Fig. 1 illustrates the overall framework. During the training phase, training is conducted for both the context encoder and policy network. The context encoder generates a latent vector z that is derived from the context information, serving as input for training the policy network. During the adaptation phase, the robot first executes a short trajectory to obtain an appropriate z through the context encoder. This derived z is subsequently utilized as part of the input for the policy network to determine $\pi_\theta(a_t|s_t, z)$. Meanwhile, the reference action generator can output $\pi_G(a_t|s_t)$ given the reference leg. The final target joint position of the robot is the result of adding these two components together.

B. Reference Action Generator

The legs of the quadruped robot go through a repetitive motion while it moves. We define one full cycle as T_{stride}, which consists of two phases: the swing phase T_{sw} and the stance phase T_{st}. More precisely, $T_{stride} = T_{sw} + T_{st}$. We designate an empirical value of 0.2s for T_{sw}, and set $T_{st} = \frac{2L_{span}}{v_d}$, where v_d stands for the predetermined speed (set at $0.5\mathrm{m/s}$), and L_{span} represents half of the stride length.

The phase generation of the reference action generator relies on ground contact information obtained from the non-injured leg. If the motor in the front right (FR) leg, hind left (HL) leg, and hind right (HR) is immobilized, we consider the front left (FL) leg to be the reference leg. On the flip side, if the motor in the FL leg becomes immobilized, we designate the FR leg as the reference leg. A touch-down event is classified as when the force exerted in the z-direction surpasses a predetermined threshold, indicating that the reference leg has made contact with the ground. The occurrence of this event is indicated by a boolean value TD, which is set to one when the reference leg is touching the ground and zero during the swing phase. The phase of each leg can be obtained from (3-5).

$$t_{ref}^{elapse} = \begin{cases} t - t_{ref}^{TD} & if\ 0 < t_{ref}^{elapse} < T_{stride} \\ T_{stride} & if\ t_{ref}^{elapse} > T_{stride} \end{cases} \tag{3}$$

$$t_{ref}^{TD} = t \quad if \quad TD = 1 \tag{4}$$

$$t_i = t_{ref}^{elapse} - \Delta S_{ref,i} T_{stride} \tag{5}$$

where t_{ref}^{TD} represents the moment when the reference leg touches down on the ground, t_{ref}^{elapse} denotes the time elapsed since t_{ref}^{TD}, and t_i corresponds to the individual clock for each leg, with $i \in \{FL, FR, HL, HR\}$. In this context, $\Delta S_{ref,i}$ indicates the phase difference of leg i relative to the reference leg. We utilize the trot gait in this paper. Therefore, when FL is chosen as the reference leg, the value is shown in (6). Conversely, when FR is selected as the reference leg, the value is shown in (7). The time for each leg is normalized by mapping t_i to $S_i(t)$. Specifically, during the swing phase, $0 \le S_i(t) \le 1$, whereas during the stance phase, $1 \le S_i(t) \le 2$. More information can be found in (8).

$$\Delta S_{trot}^{FL} = \begin{bmatrix} \Delta S_{ref,FL} \\ \Delta S_{ref,FR} \\ \Delta S_{ref,HL} \\ \Delta S_{ref,HR} \end{bmatrix} = \begin{bmatrix} 0.0 \\ 0.5 \\ 0.5 \\ 0.0 \end{bmatrix} \tag{6}$$

$$\Delta S_{trot}^{FR} = \begin{bmatrix} \Delta S_{ref,FL} \\ \Delta S_{ref,FR} \\ \Delta S_{ref,HL} \\ \Delta S_{ref,HR} \end{bmatrix} = \begin{bmatrix} 0.5 \\ 0.0 \\ 0.0 \\ 0.5 \end{bmatrix} \tag{7}$$

$$S_i(t) = \begin{cases} \frac{t_i + T_{stride}}{T_{st}} & -T_{stride} < t_i < -T_{sw} \\ \frac{t_i + T_{sw}}{T_{sw}} & -T_{sw} < t_i < 0 \\ \frac{t_i}{T_{st}} & 0 < t_i < T_{st} \\ \frac{t_i - T_{st}}{T_{sw}} & T_{st} < 0 < T_{stride} \end{cases} \tag{8}$$

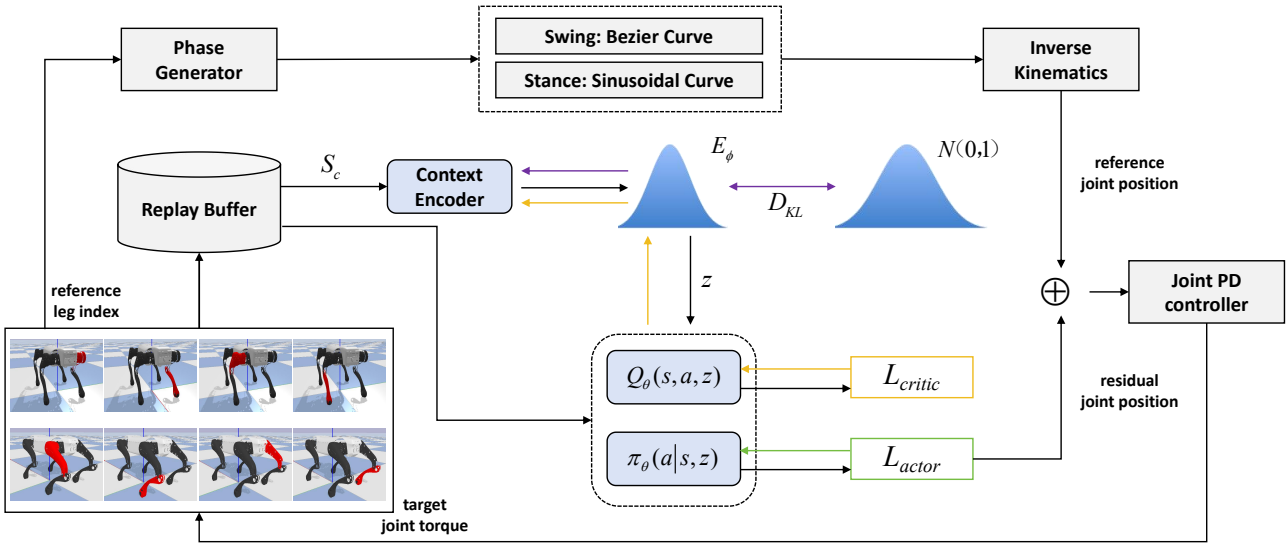

Fig. 1. Framework of the proposed method.

After identifying the phase, we use it to create the foot trajectory for the robot. For the swing phase of the leg, we employ a Bezier curve [9] as the reference trajectory. On the other hand, for the stance phase, we integrate a sinusoidal curve as the reference trajectory.

$$p_i = \begin{cases} p_i^{sw}(S_i(t)) = \sum_{k=0}^{n} c_k B_k^n(S_i(t)) & 0 \le S_i(t) \le 1 \\ \left[\begin{array}{c} p_{i,x}^{st}(S_i(t)) = L_{span}(1-2S_i(t)) \\ p_{i,y}^{st}(S_i(t)) = \sigma \cos\left(\frac{\pi}{2L_{span}} p_{i,x}^{st}(S_i(t))\right) \end{array}\right] & 1 \le S_i(t) \le 2 \end{cases} \tag{9}$$

where c_k represents the k-th control point and $B_k^n(S_i(t))$ symbolizes the Bernstein polynomial of degree n. p_i^{sw} refers to the robot's foot position during the swing phase, while p_i^{st} represents the foot's position during the stance phase, with σ acting as the amplitude variable. By employing inverse kinematics, we can ascertain the reference joint position of the robot.

$$\pi_G(a_t\,|s_t) = IK(p_i) \tag{10}$$

The term s_t represents the information about the robot's foot contact, which is used to determine if a touchdown (TD) event has happened.

C. Meta Reinforcement Learning

When motors get locked, it results in changes to the robot's dynamics model, leading to diverse transition functions. While Randomization [10] can tackle this issue by training a generalized model across a range of scenarios, it might result in the policy losing optimality. Therefore, in this paper, we favor a context-based meta-reinforcement learning approach [11] in an effort to develop a policy capable of delivering superior strategies under various motor locked situations.

We leverage the context c to infer the latent vector z, which acts to discern the current task $\mathcal{T}$. Here, $c_n^{\mathcal{T}} = (s_n, a_n, r_n, s'_n)$ represents a state transition in task $\mathcal{T}$, with $c_{1:N}^{\mathcal{T}}$ encapsulating all prior experiences. To procure z, we train a context encoder denoted by E_ϕ. In light of the fact that a fully observed Markov Decision Process (MDP) should respect permutation invariance, that is to say, for task inference, access to one tuple is necessary regardless of the chronological sequence of tuple observation. Keeping this paradigm in mind, we represent the latent vector as follows:

$$z \sim \mathcal{N}(\Pi_{n=1}^{N} E_\phi^\mu(c_n), \Pi_{n=1}^{N} E_\phi^\sigma(c_n)) \tag{11}$$

where $\mathcal{N}$ denotes the normal distribution. During the training phase, we infer z from the training task to characterize the specific task distribution and learn to deduce a new task based on prior experiences. During the adaptation stage, we initially collect experience to infer z for the present task, then we utilize z to extract the policy appropriate for that task. To enhance data utilization, we integrate the off-policy Reinforcement Learning (RL) method, Soft Actor-Critic (SAC) [12], with the context encoder and consider the latent vector z as part of the state. The loss function for each network is outlined as follows:

$$L_{encoder} = L_{critic} + \beta D_{KL}(\mathcal{N}(\prod_{i=1}^{N} E_\phi^\mu(c_n), \prod_{i=1}^{N} E_\phi^\sigma(c_n)) \Big\| \mathcal{N}(0,1)) \tag{12}$$

$$\mathcal{L}_{critic} = \mathbb{E}_{\substack{(s,a,r,s')\sim B \\ z\sim E_\phi(z|c)}} \left[\frac{1}{2}\left(Q_\theta(s,a,z) - (r + \bar{V}(s',\bar{z}))\right)\right]^2 \tag{13}$$

$$\mathcal{L}_{actor} = \mathbb{E}_{\substack{s\sim B, a\sim\pi_\theta \\ z\sim E_\phi(z|c)}} \left[\alpha \log \pi_\theta(a\,|s,\bar{z}) - \min_{i=1,2} Q_{\theta_i}(s,a,\bar{z})\right] \tag{14}$$

where $\bar{V}$ signifies the target value network and $\bar{z}$ indicates the absence of gradient computations being run through it. As shown in Fig. 1, between the replay buffer and the context encoder, there is a sampler referred to as S_c. It utilizes the recently collected samples in the replay buffer to train the context encoder and achieve an approximate on-policy training effect. Conducting an on-policy during the adaptation stage ensures the alignment of the two distributions. When training the actor and critic networks, sampling is performed directly from the replay buffer, implementing an off-policy training approach that enhances sample efficiency.

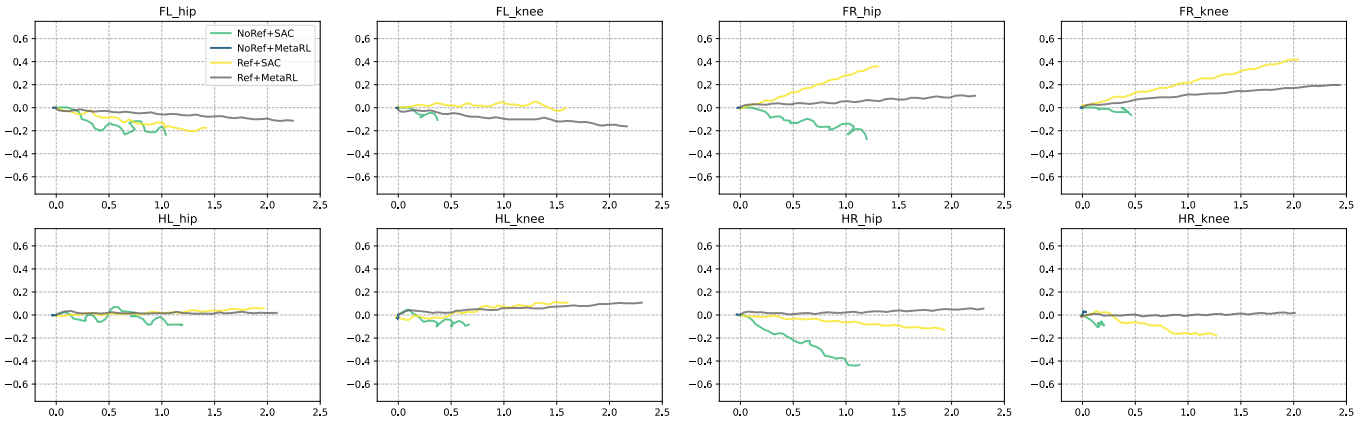

Fig. 2. The trajectory of the robot in the ablation experiment, begins at (0,0), with a movement command to move right.

III. Experiment

A. Experimental Setup

1) Observation-action space and reward function: The state, denoted by $s_t \in \mathbb{R}^{37}$, consists of a number of components. These components include the linear velocity and pose of the base, as well as the angular velocity. Furthermore, the state also includes the angle and angular velocity for each joint, along with four binary values that indicate whether each foot is in contact with the ground.

The action $a_t \in \mathbb{R}^{12}$ corresponds to the desired angles for the 12 motors. Notably, the angle of the locked motor persists as constant, irrespective of the network's output. Following this, the joint angles are translated into torques via a Proportional-Derivative (PD) controller prior to being relayed to the robot.

Our reward function includes six components, outlined as follows: (1) Offering rewards for the robot's body to move in a specific direction. (2) Imposing penalties for roll and pitch rotations of the robot's body. (3) Encouraging the movement of the robot's feet in the desired direction. (4) Applying a cost penalty to the motors. (5) Penalizing contact between the robot's body (excluding the feet) and the ground. (6) Promoting contact between the feet and the ground.

2) Training details: During the training phase, we leverage Pybullet as the simulation environment, integrating URDF sourced from DeepRobotics. We use PyTorch to build the network and utilize an NVIDIA 3080ti to accelerate training. Since the robot does not need to turn, the rotation angles of the four abduction joints are negligible, taking into account the forward movement task. Consequently, if these joints become locked, the impact on the task is minimal. Hence, we have chosen the four hip joints and four knee joints as probable motors that could experience locking. In the event of a hip joint lock, we set its angle to -0.9rad, whereas for a locked knee joint, we position it at 1.8rad. The network structure is presented in Tab.I, while the values of certain hyperparameters are displayed in Tab.II.

B. Ablation Experiment

This section includes four ablation experiments that were designed to access the impact of each component in the proposed method. The setting of each experiment is outlined as follows:

TABLE I
Network Architecture

Module	Inputs	Hidden Layers	Outputs
Context Encoder(MLP)	s_t, a_t, r_t	[200,200,200]	z
Actor Network(MLP)	s_t, z	[300,300,300]	a_{t+1}
Critic Network(MLP)	s_t, a_t, z	[300,300,300]	Q value

TABLE II
Hyperparameter Settings

Hyperparameter	Value
Non-linearly	ReLU
Discount factor	0.99
Batch Size	256
Context encoder learning rate	3e-4
Actor/Critic learning rate	3e-4
Optimizer	Adam
KL loss weight	1

NoRef+SAC: Without using the reference action generator. The robot's initial default joint angle J_{init}, is set to $([0, -0.9, 1.8] \times 4)$, and the action space is set to $([\pm 0.2, \pm 0.7, \pm 0.7] \times 4)$. The final angles of each joint J_{final}, are obtained by adding J_{act} to J_{init}. The action J_{act} is chosen by SOTA off-policy method SAC [12] and the joints are randomly selected to be locked during the training process.

NoRef+MetaRL: Without using the reference action generator. The action space and the final joint angle are designed to be the same as NoRef+SAC. The meta RL algorithm selects the action J_{act}. The training set is {FL_hip, FL_knee, FR_hip, FR_knee}, and the testing set is {HL_hip, HL_knee, HR_hip, HR_knee}.

Ref+SAC: Utilizing the proposed reference action generator. The SAC algorithm selects actions, while the locked joints are randomly chosen during the training process.

Ref+MetaRL: The proposed method. The training set is {FL_hip, FL_knee, FR_hip, FR_knee}, and the testing set is {HL_hip, HL_knee, HR_hip, HR_knee}.

TABLE III
ANALYSIS OF COMPARISON EXPERIMENT

Criterion	Method \ Joint	FL_hip	FL_knee	FR_hip	FR_knee	HL_hip	HL_knee	HR_hip	HR_knee
x-distance	***GrBAL***	0.4571±0.0744	0.8672±0.0387	0.4741±0.2050	0.8257±0.1397	0.5017±0.1336	0.6565±0.2007	0.1730±0.1431	0.7847±0.0514
	ReBAL	0.4311±0.0985	0.8320±0.0772	0.4796±0.1081	0.8148±0.0890	0.4513±0.0590	0.7873±0.1354	0.2959±0.0367	0.9173±0.0509
	Ref+MetaRL(Ours)	**2.1608±0.0954**	**2.2581±0.0573**	**2.2314±0.0296**	**2.3482±0.1067**	**2.1246±0.0286**	**2.2728±0.0749**	**2.2822±0.0558**	**2.1976±0.2224**
$cos(\theta)$	***GrBAL***	0.8731±0.0833	0.9217±0.0430	0.8504±0.1104	0.9633±0.0436	0.8367±0.1251	0.9229±0.0653	0.3480±0.2709	0.9313±0.0582
	ReBAL	0.9117±0.0872	0.8994±0.0970	0.8289±0.1311	0.9483±0.0232	0.8876±0.0413	0.9328±0.0610	0.5465±0.1023	0.9779±0.0181
	Ref+MetaRL(Ours)	**0.9974±0.0021**	**0.9956±0.0013**	**0.9982±0.0004**	**0.9964±0.0007**	**0.9990±0.0012**	**0.9988±0.0002**	**0.9999±0.0001**	**0.9983±0.0015**
Yaw	***GrBAL***	0.6360±0.1084	0.2720±0.0297	0.3893±0.2189	0.1848±0.1070	0.4775±0.1548	0.3557±0.1888	0.9289±0.2700	0.2986±0.1707
	ReBAL	0.4995±0.0678	0.2283±0.0827	0.7275±0.1171	0.1973±0.0779	0.4000±0.0749	0.3189±0.1725	0.7002±0.0171	0.2316±0.1421
	Ref+MetaRL(Ours)	**0.0493±0.0031**	**0.0588±0.0032**	**0.0523±0.0029**	**0.0475±0.0035**	**0.1003±0.0.0238**	**0.0350±0.0020**	**0.0757±0.0036**	**0.0535±0.0077**

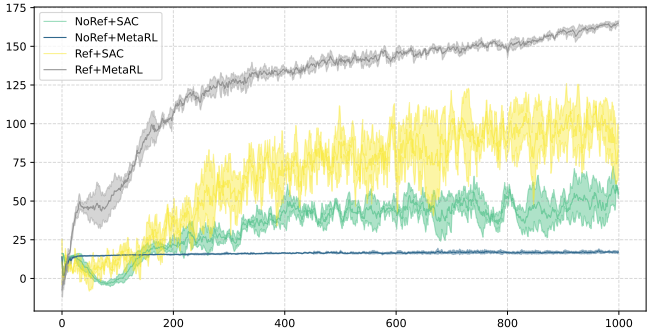

Fig. 3. The rewards comparison of ablation experiments. The x-axis indicates the number of epochs, and the y-axis represents the cumulative rewards.

In Fig. 3, we display the rewards achieved by four different methods, each tested under three unique seeds. Furthermore, in order to provide a more perceptible representation of the robot's trajectory, we visually present the robot's trajectory using a single random seed in Fig. 2. Upon analyzing the data from Fig. 3, it is evident that the proposed method produced the highest rewards. Ref+SAC and NoRef+SAC closely follow while NoRef+MetaRL returns comparatively lower rewards. Fig. 2 highlights that only the proposed method and Ref+SAC successfully move the robot in all situations, with the proposed method demonstrating a straighter trajectory.

The superiority of meta-RL is highlighted by comparing the proposed method with Ref+SAC. Our proposed method applies contextual reasoning to derive latent vectors better suited to varied joint-locked situations, in contrast to Ref+SAC, which opts for a broader generalization across joint-locked situations, albeit at the cost of performance. It's pertinent to note that the training process of our method only involved the task in the training set, with no encounters with the task in the testing set, unlike Ref+SAC which factored in all situations. Despite this, the proposed method yields better trajectories for the task in the testing set compared to Ref+SAC, demonstrating the method's noteworthy performance in unknown environments.

NoRef+MetaRL's relatively low rewards suggest that meta-RL's effectiveness is contingent on the design of the action space. Fig. 2 illustrates that the robot remains stationary with minimal movement when NoRef+MetaRL is used. In our proposed method, the use of a reference action generator to provide priors facilitates a speedier accumulation of positive samples in the replay buffer, which reduces sparse rewards and significantly enhances training performance.

C. Comparison with Baselines

In this section, we evaluate the proposed approach by comparing it to two model-based reinforcement learning methods. The baselines are introduced first.

Gradient-Based Adaptive Learner (GrBAL, [6]): GrBAL employs gradient-based meta-learning, specifically MAML (Model-Agnostic Meta-Learning), to optimize an adaptive meta-objective for training a dynamics model, enabling online adaptation.

Recurrence-Based Adaptive Learner (ReBAL, [6]): The model-based Meta-RL approach, which bears resemblances to GrBAL in multiple aspects, utilizes a recurrent model. This approach acquires its own update rule by employing an internal gating structure.

Tab.III presents the results of our comparative experiments. The performance is assessed using three distinct evaluation metrics. Firstly, we measure the distances covered by the robot in the x-direction, where a greater distance indicates better performance. The second criterion involves calculating the cosine of the angle between two vectors: one formed from the endpoint to the starting point, and the other being (1,0). A cosine value closer to one infers that the robot's trajectory is more finely aligned with the target trajectory. In the end, we examine the average yaw angle of the robot throughout the entire trajectory, where lower values indicate a posture that closely resembles the target posture. According to these three criteria, our method shows the best performance.

We speculate that the reason for this outcome lies in the fact that the baseline methods are based on dynamic models, which perform better on tasks with fewer joints and higher stability, such as Ant. However, the quadruped robot used in this paper has a larger number of joints and lower stability, making the dynamics model less predictable and resulting in lower performance. On the other hand, the proposed approach, combining meta-reinforcement learning with residual learning, does not require the establishment of a dynamic model and can better handle such challenges.

To provide a clear understanding of the proposed algorithm's effectiveness, we have included Fig. 4, showcasing the

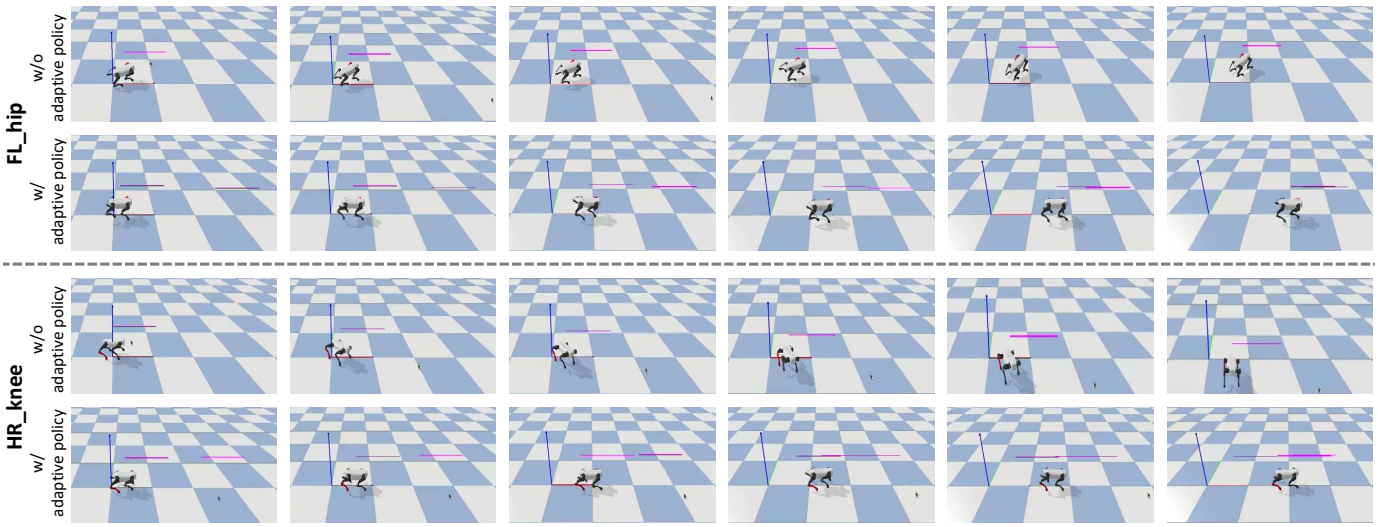

Fig. 4. Snapshot of the robot under different joint locking situations.

snapshot of the robot using both the proposed method and the traditional approach. By comparing the two, we can assess the effectiveness of the proposed method.

IV. Conclusions

In this paper, we propose a fault-tolerant control method for the robot in the event of joint locking, utilizing a residual meta-reinforcement learning algorithm. The approach consists of two parts: a reference trajectory generator and a joint space action corrector. Simulation experiments have verified that the proposed method can enable the robot to follow a predetermined trajectory when a joint is locked. However, in this paper, only the task of walking forward is considered, while the quadruped robot has omnidirectional motion capability. Therefore, in future work, we will further develop the omnidirectional motion performance of the quadruped robot under joint stuck conditions. Additionally, we will gradually address the sim2real problem that arises when deploying the reinforcement learning algorithm on a physical robot and achieve verification on the physical robot.

Acknowledgment

This work was supported in part by the National Key R&D Program of China under Grant 2021ZD0114500, and in part by the National Natural Science Foundation of China under Grant U2013601.

References

[1] 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.

[2] 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*, vol. 8, no. 4, pp. 2991–3003, 2022.

[3] 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.

[4] 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.

[5] 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*, vol. 121, pp. 228–244, 2018.

[6] A. Nagabandi, I. Clavera, S. Liu, R. S. Fearing, P. Abbeel, S. Levine, and C. Finn, "Learning to adapt in dynamic, real-world environments through meta-reinforcement learning," *arXiv preprint arXiv:1803.11347*, 2018.

[7] K. Lee, Y. Seo, S. Lee, H. Lee, and J. Shin, "Context-aware dynamics model for generalization in model-based reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 5757–5766.

[8] Y. Seo, K. Lee, I. Clavera Gilaberte, T. Kurutach, J. Shin, and P. Abbeel, "Trajectory-wise multiple choice learning for dynamics generalization in reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 33.

[9] G. G. Lorentz, *Bernstein polynomials*. American Mathematical Soc., 2013.

[10] 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.

[11] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen, "Efficient off-policy meta-reinforcement learning via probabilistic context variables," in *International conference on machine learning*. PMLR, 2019, pp. 5331–5340.

[12] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International conference on machine learning*. PMLR, 2018, pp. 1861–1870.