Proceedings of The 2022 IEEE International
Conference on Real-time Computing and Robotics
July 17-22, 2022, Guiyang, China

Fast Adaptation Dynamics Model for Robot's Damage Recovery

Ci Chen[1], Dongqi Wang[1], Jiyu Yu[1], Pingyu Xiang[1], Haojian Lu[1], Yue Wang[1], Rong Xiong[1]

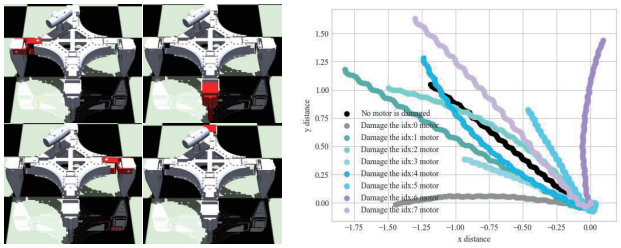
2022 IEEE International Conference on Real-time Computing and Robotics (RCAR) | 978-1-6654-6983-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/RCAR54675.2022.9872230

***Abstract*— In the process of operating, robots will inevitably encounter damage due to external or internal factors, such as motors blockage. For the legged robot, when the motors of joints are failing, if other motors still act according to the original instructions, it will cause the robot to deviate from the predetermined trajectory, which is unacceptable for legged robots. Inspired by the fact that the model trained by supervised learning on the training set can be generalized to the testing set, our goal is to obtain a dynamic model that can be generalized to all kinds of motor damage situations. It can predict what state will be reached in the next step when an action is applied in the current state. With this dynamics model, we use the Monte Carlo particles to optimize the feasible actions in a model predictive control (MPC) fashion and achieve the expected goal (such as making the robot walk in a straight line). The comparison experiment adopt two meta-learning model and vanilla dynamics model approaches, the results show that the proposed method is superior to the three baselines, which proves the effectiveness of the proposed method.**

I. INTRODUCTION

Legged robots have great potential in search and rescue, disaster response, and routing inspection. However, one of the factors limiting their widespread adoption in complex environments is their fragility. Emergency braking of non-fatal damage will undoubtedly degrade the robot's efficiency. Especially in the field of search and rescue, it's not convenient for humans to access in many cases, so it's impossible to repair the damaged robots. Nevertheless, if the robot still executes the original control command, unpredictable behavior may be produced, which may cause unacceptable damage to itself or the external environment. For the legged robot, when the motor is damaged, if other motors don't make adaptive action adjustments, it will cause the robot to deviate from the predetermined trajectory and even cause the robot to overturn, as is shown in Fig. 1.

The use of robots to assist the handicapped has made long-term progress [1], [2], but the research on how to restore robots after an injury is relatively less compared with the former. Traditional robot damage recovery consists of two steps, first self-diagnosis is performed, and then the most appropriate alternative strategy based on the diagnosis results is selected, which is pre-designed [3], [4]. However, alternative policies are difficult to design because the probability of damage increases exponentially with the complexity of the robot, making it hard to consider all possible scenarios beforehand. Furthermore, this approach often fails, due to the results of the self-diagnosis are not necessarily correct, or there are no alternative policies designed to meet this situation in advance.



(a) (b)

Fig. 1: (a) Examples of motor damage in a legged robot, the damaged motor is marked in red. (b) The trajectory of the robot when one motor is damaged and the original forward motion command is still applied. The black dots indicate the trajectory of the robot without motor damage, other colors represent the trajectory of the robot under different motor damage situations. The starting point of the robot movement is at (0,0).

The recent research about damage robots recovery can be roughly divided into two types, the first is the trial-and-error method, and the second is the learning-based fast parameters adaptive method. [5] used the robot's actuation-sensing relationship to indirectly infer its morphology structure and used this self-model to generate forward motion. When part of the leg was removed, the self-model was adjusted to generate new gaits. [6] proposed the T-resilience method, which can be divided into two parts. The first part includes self-modeling and controller evaluation, and the second part is responsible for transferring the controller to real-world robots. The performance of the robots in the real world will be used to update the robots' self-modeling in the first part. [7] proposed an intelligence trial-and-error method, which enables the robots to store previous experience knowledge in the form of a behavior-representation space map, and uses the Bayesian optimization method combined with the map to guide the damaged robot to try different types of behaviors. However, these methods also have drawbacks. In the early stage of trial and error, due to the algorithm is not yet perfect, the robot may perform some behaviors that are harmful to the robot's body or the outside world.

In the second learning-based parameters adaptation approach, In the second learning-based parameters adaptation approach, [8] proposed a shared module strategy (SMP), which treats each actuator of the robot as an agent, and

[1]Ci Chen, Dongqi Wang, Jiyu Yu, Pingyu Xiang, Haojian Lu, Yue Wang and Rong Xiong are with the State Key Laboratory of Industrial Control and Technology, and Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou, 310027, China. Haojian Lu and Yue Wang are the corresponding authors. luhaojian@zju.edu.cn, wangyue@iipc.zju.edu.cn.

there are bottom-up and top-down message transmission mechanisms between agents. The results show that a single strategy can provide control policies for robots with different morphological structures. Therefore, when the robot breaks its leg, resulting in a change in morphology structure, the previous strategy can still be applied. However, this method needs to obtain the speed and pose of each limb, they are difficult to obtain in the actual environment, which limits it to be used only in the simulation environment. [9] proposed a fast parameter adaptive method combining meta-learning and model predictive control method. However, in the new environment, further training is needed to fine-tune the network parameters to better adapt to the new environment. [10] proposed policy dynamic value function (PD-VF) . The key idea is to obtain a value function based on policy and dynamics embedding by using the supervised learning method. When testing, the strategy that can maximize the rewards through less interaction can be inferred. However, [11] takes this method as a comparative experiment, and finds its performance is poor, which may be caused by the inaccurate estimation of value function conditioned on policy and environment embeddings.

In this paper, a robot damage adaptive recovery method based on model-based reinforcement learning is proposed. It mainly consists of two parts: the first part is a probabilistic ensemble dynamics model based on future trajectory enhancement and damage information addition. Compared with the deterministic dynamics model, the probabilistic model can better describe the inherent randomness of the dynamics system. Ensemble dynamics model can effectively alleviate the inaccuracy caused by less training data. Furthermore, we enhance the future trajectory of the training data and use a 0-1 vector to encode the damage information of the robot in the state of the robot. Experiments show that these two methods can effectively improve the generalization effect of the algorithm. After the dynamics model is well trained, given the current state and action, the next state can be predicted. The second part is based on the idea of the model predictive control (MPC) method. By using the dynamics model obtained in the previous step, the planning algorithm based on Monte Carlo particles can obtain the optimal action trajectory in the future and select the first action to be executed by the robot. Furthermore, our method allows changing the reward function online and therefore is able to modify the behavior of the robots.

The remainder of the paper is organized as follows: Section II is a brief summary of the related works. Section III describes the proposed method in detail. Section IV shows the experimental results. Finally, we conclude and suggest future work in SectionV.

II. Related Works

A. Model-based Reinforcement Learning

By learning a forward dynamics model to approximate the state transition dynamics of the environment, model-based reinforcement learning (MBRL) methods can achieve better sampling efficiency than model-free reinforcement learning (MFRL) methods. The learned dynamics model can be used as a simulator of MFRL [12], [13], provide priors for the algorithm [14], [15], or be utilized to predict future trajectory in an MPC fashion [16], [17]. The main challenges are how to obtain an accurate dynamics model and how to make the learned model more generalizable. To this end, many scholars have carried out a series of works.

[18] applied the MBRL method to the physical quadruped robot. It only takes 4.5 minutes to collect data on the physical robot, and after being trained, the quadruped robot can walk smoothly on the ground. To make the method more suitable for real-time control, the two loops are decoupled by parallelizing the planning and control frequency. In addition, trajectory generators [19] are used instead of directly learning the position control signal of the motor for obtaining smooth gaits. [20] proposed a new method called probabilistic ensembles with trajectory sampling (PETS), which uses the ensemble models for prediction and the probabilistic dynamic models to output the distribution, which can isolate two types of uncertainties in MBRL: aleatoric and epistemic. Besides, it can greatly improve sampling efficiency compared to model-free methods. Our method is also based on the idea of this method. [21] is proposed aimed at service robots, which need to ensure safety in the process of interacting with people. To achieve this goal, a probabilistic latent variable model is used to enable fast inference of the posterior environment transition distribution given contextual data, and then uncertainty-aware trajectory sampling is used to evaluate safety constraints. [22] proposed an improved model-based meta RL approach. In order to achieve online model adaptation, based on the idea of meta-learning, this method learns different latent vectors in each training scenario. Experiments show that the learned model can be applied to scenarios such as changes in ground friction and external forces applied to the robot.

B. Legged Robot Control

[24] design a terrain estimation method based on generalized least square by fusing the body, leg, and contact information. Based on virtual model control (VMC) and quadratic program (QP), the optimal foot force adapted to the terrain is obtained. [25] propose a hierarchical learning method used to realize an 18 DOF spider robot reach any target. In the lower level, the model-free reinforcement learning method is used to train the motion primitives. In the upper level, the model predictive control method is utilized to sequence these skills. This approach improves the sampling efficiency and generalization of real-world robots. [26] also adopts the hierarchical framework, in which the upper level policy issues instructions in the latent space, and can specify the execution time of this latent command, the lower level policy uses this latent command and the information obtained by the robot's sensors to control actuators. The advantage of this approach is that it enables the upper controller to run at a lower frequency than the lower controller.

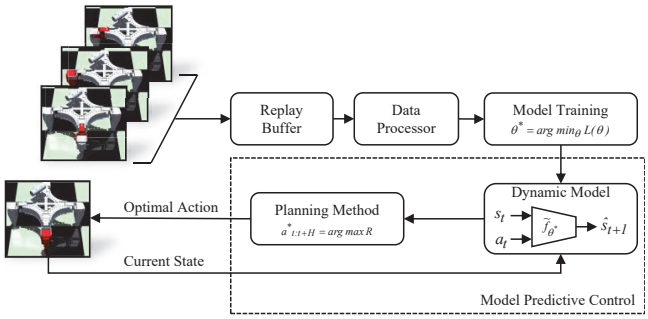


Fig. 2: Framework of proposed method.

III. Methodology

A. Problem Statement

The process of a robot interacting with environment is a standard reinforcement learning problem, which can be modeled as a Markov decision process defined by a tuple $(S, A, p, r, \gamma, \rho_0)$. Here, S represents the state space, A denotes the action space, $p(s_{t+1}|s_t, a_t)$ is the state transition probability of applying action a_t under state s_t, $r(s_t, a_t)$ indicates the reward value given by the environment when action a_t is applied under state s_t, $\gamma \in (0, 1]$ is the discounted factor used to weight the reward, and ρ_0 represents the initial state distribution. Under the framework of RL, the RL algorithm tries to find a strategy π so that in each corresponding state, there is an optimal action that maximizes the cumulative discounted reward.

The proposed method is a model-based RL algorithm, and the goal is to learn a robust dynamic model $\tilde{f}_\theta$ with good generalization. By means of supervised learning, the motor of the robot is damaged randomly firstly, then the damaged robot is used to interact with the environment to get the training set $D = \{(s_t, a_t, s_{t+1})\}$. These data are used to train the dynamic model, when the dynamic model is well trained, it can generalize to situations that are not included in the training set (in the context of this paper, it refers to motor damage situations that are not included in the training set). This trained model can be used to approximate the real dynamics model f, when given the current state-action repair (s_t, a_t), the next state s_{t+1} can be predicted. The predicted s_{t+1} can be used to guide the subsequent planning method to select the optimal action. Therefore, even in the case of motor damage not included in the training set, the robot can still move forward as expected.

B. Dynamics Model

In the model-based RL method, when using neural network for dynamic modeling, it can be divided into deterministic neural network and probabilistic neural network. The former is relatively simple to complete, but it is easy to overfit when there are few samples. Compared with the former, the latter increase the ability to describe aleartoric uncertainty (the inherent system stochasticity). For the latter, the neural network receive s_t and a_t as input and output a distribution of s_{t+1}. The most common distribution is the Gaussian distribution $\tilde{f}_\theta = \Pr(s_{t+1}|s_t, a_t) = N(\mu_\theta(s_t, a_t), \Sigma_\theta(s_t, a_t))$, and the loss is:

$$L(\theta) = E_{(s_t, a_t, s_{t+1}) \sim D}[-\log \tilde{f}_\theta(s_{t+1}|, s_t, a_t)] \quad (1)$$

In the proposed method, the damage state of the robot is represented by a 0-1 vector C, in which 0 indicates that the motor is damaged, and 1 represents the motor isn't damaged. We concatenate this vector C with original state s_t and rewrite it as $s_t = CONCAT(s_t, c)$. Furthermore, we use the future trajectory to enhance the training data of the model, record the future trajectory as $\tau_{t,M} = \{(s_t, a_t), ..., (s_{t+M}, a_{t+M})\}$, so the modified loss of probabilistic model is:

$$L(\theta) = E_{\tau_{t,M} \sim D}[-\frac{1}{M} \sum_{i=t}^{t+M-1} \log \tilde{f}_\theta(s_{i+1}|, s_i, a_i)] \quad (2)$$

In the process of using neural networks to establish dynamics models, there is not only aleatoric uncertainty but also epistemic uncertainly, which is caused by fewer samples and insufficient estimation. To alleviate the epistemic uncertainty, the ensemble models are adopted, which include B bootstrap models. Using θ_B to refer to the parameters of bth model $\tilde{f}_{\theta_b}$, then the final predictive parameters' value is the average value of ensemble model's parameters, i.e. $\tilde{f}_\theta = \frac{1}{B} \sum_{b=1}^{B} \tilde{f}_{\theta_b}$. Therefore, the final updating formula of the dynamic models' parameters is:

$$\theta \leftarrow \theta - \alpha \nabla_\theta \frac{1}{B} \sum_{i=1}^{B} L_i(\theta) \quad (3)$$

C. Planning Method

We adopt the model predictive control (MPC) method to plan the actions. These methods can be roughly divided into three types according to how they represent the state distribution: deterministic, parametric, and particle methods. [27] shows the particle methods are competitive in terms of computability and accuracy and don't require strong assumptions about the distribution, so we choose particle-based methods in our approach. This method evolves a group of Monte Carlo particles and takes the distribution of these particles as the distribution after selecting an action. When using the ensemble dynamic model, for each particle, only one ensemble model is used from the beginning to the end. Assuming there are P particles and the prediction horizon is H, then the evaluation function of the action is written as:

$$R = \sum_{i=t}^{t+H} \frac{1}{P} \sum_{p=1}^{P} r(s_i^p, a_i) \quad (4)$$

With the evaluation function in hand, we employ the Cross-Entropy Method (CEM) to optimize the next action. Randomize the initial sample distribution firstly, then verify the action effects by (4). Next, extract the parameters with better effect in the previous part (for example, the top 20%), readjust the parameters of the distribution (mean and standard deviation) according to the newly extracted parameters, and optimize them continuously. The goal of this

method is to minimize the cross-entropy between the data distribution obtained by Monte Carlo particles and the real data distribution and try to make the sampling distribution of the particles the same as the real situation. After several rounds of iteration, the controller selects the first action of the optimized action sequence as the action to be applied to the robot. The overall flow of the proposed method is summarized in Fig. 2 and Algorithm 1.

Algorithm 1 Framework of proposed method

Initialize parameters of forward dynamics model θ;
Initialize replay buffer D;
for each iteration **do**
  // COLLECTING TRAINING SAMPLES
  for t = 1 to PathLength **do**
    Execute a^* from planning part (or random policy for the first iteration) to collect $\{(s_t, a_t, s_{t+1})\}$;
    Update $D \leftarrow D \cup \{(s_t, a_t, s_{t+1})\}$;
  end for
  Process data for future trajectory enhancement;
  // UPDATE DYNAMICS MODEL
  for b = 1 to B **do**
    Sample $\tau_{t,M} \sim D$;
    Pass through dynamics model and calculate the loss function according to E.q.(2);
  end for
  Update θ according to E.q.(3);
  // PLANNING
  for p = 1 to P **do**
    for $i = t$ to $t + H$ **do**
      Calculate evaluation function according to E.q.(4);
      Update $CEM(\cdot)$ distribution;
    end for
  end for
  Return first action a^* from optimal actions $a^*_{t:t+H}$.
end for

IV. Experiments

A. Simulation Setup

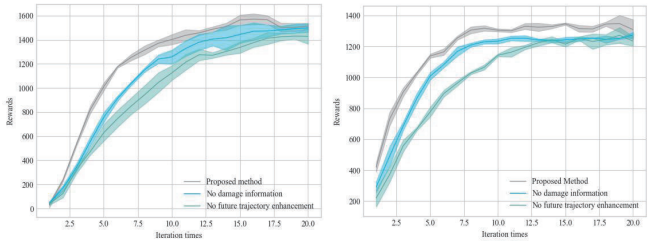
The proposed method and comparative experiment are trained in a virtual environment created by Mujoco [28]. It was chosen because it simulates contact forces more realistically than other simulators, which is important for legged robots. The training is performed using Tensorflow [29] on four NVIDIA Geforce GTX 2080Ti GPUs. In addition, to speed up training, we use MPI4PY to build a parallel environment to speed up the training process[1]. The robot consists of a body, four thighs, and four calves, it has a total of eight degrees of freedom. The 3D model of each component is modeled with SolidWorks software [30]. After modeling, the .STL file and the physical parameters such as the mass the moment of inertia of the component are exported. Using that information, a .XML file can be made to describe the entire robot. The state includes the position, posture, and line velocity of the body link, as well as the position and velocity of each joint. The reward is designed as the speed of the robot in the x-axis direction. The specific parameters in the training process are shown in Table I.

[1]https://github.com/mpi4py/mpi4py

TABLE I: Parameters Setting

Param	Value
learning rate	0.001
batch size	256
path length in one epoch	1000
train epochs in one iteration	20
total iteration	20
valid split ratio	0.1
hidden size	(200, 200, 200, 200)
nonlinearity	swish
ensemble size	5
particles	20
candidates (CEM)	200
horizon (CEM)	30

B. Ablation Experiment



(a) Training Set (b) Testing Set

Fig. 3: The comparison of the reward curves of the proposed method, the method without future trajectory enhancement, and the method without damage information under the training set and testing set.

To verify the effect of future trajectory enhancement and the addition of damage information in the proposed method, we design ablation experiments. The results of the training set are shown in Fig. 3 (a), and the result of the testing set is shown in Fig. 3 (b). The experiment results are obtained from three groups of different random seeds, where the solid lines represent the mean value and the shaded parts represent the variance of results. It can be seen that on the training set, although the final training results of the three achieve almost the same value, the proposed method converges significantly faster than the methods without future trajectory enhancement and damage information. On the testing set, the rewards of the proposed method are higher than that of the two comparison methods, indicating that the future trajectory enhancement and the addition of damage information will indeed strengthen the generalization performance of the algorithm. We conduct a T-test on the results of the testing set, and the results are shown in Table II. The P-value on the testing set between the proposed method and the method without future trajectory enhancement is

9.9452×10^{-5}, which is less than the threshold value of 0.05, the P-value between the proposed method and the method without damage information is 8.7666×10^{-5}, which is also less than the threshold, proving that there is a significant difference between the proposed method and other methods, i.e. the rewards of the proposed method is statistically higher than that of the comparison methods.

TABLE II: T-test for the last five rewards of ablation experiment

Methods	Proposed method v.s. No future enhancement	Proposed method v.s. No damage information
P value	9.9452×10^{-5}	8.7666×10^{-5}

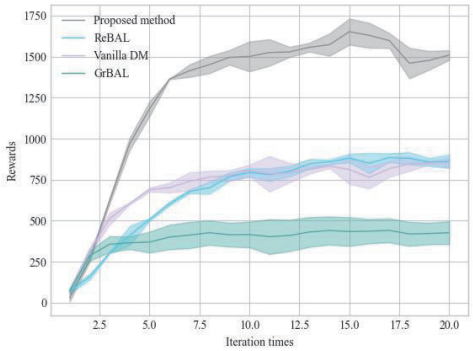


Fig. 4: Comparison with the proposed method and three baselines.

C. Comparison with baselines

In this part, we evaluate and compare the proposed method to three baselines. It is noted that we run three times using different seeds for each method to gauge consistency. Three baselines are introduced firstly.

Gradient-Based Adaptive Learner (GrBAL): [2] GrBAL uses gradient-based meta-learning to perform online adaptation. In particular, it uses MAML to train a dynamics model by optimizing an adaptation meta-objective.

Recurrence-Based Adaptive Learner (ReBAL): Model-based meta-RL method is similar to GrBAL. However, it utilizes a recurrent model to learn its own update rule (i.e., through its internal gating structure).

Vanilla Dynamics Model (Vanilla DM): Dynamics model trained to minimize the standard one-step forward prediction loss.

The comparative results of the proposed method and the three baselines are shown in Figure 4. It can be seen that the rewards of the proposed method are significantly higher than that of the three comparison methods. After convergence, the average rewards of the proposed method are about 1600, the average rewards of ReBAL and Vanilla DM are about 900, and the average rewards of GrBAL are only about 400. The results demonstrate the effectiveness of the proposed method.

[2]We used a reference implementation publicly available at https:github.comiclaveralearning_to_adapt.

D. Result Visualization

In order to visualize the results of the proposed method, the model obtained in the last iteration of training is used to verify the motion performance of the robot under three conditions of damage to the 7^{th}, 0^{th}, 0^{th} and 5^{th} motors, respectively. The 7^{th} motor was damaged in the training set, and the 0^{th} and 5^{th} motor was not damaged in that. Under three different random seeds, the motion trajectories of the robots are shown in Fig. 5. It can be seen that at the same time, the robot in the training set moves the farthest, followed by the case of one motor being damaged, and the robot moves the shortest distance when the two motors are damaged. Although the performance of the testing set is not as good as that of the training set, the final distance is around 35m in the test set, which is 87.5% of the final distance reached by the robot in the training set. Furthermore, the proposed method can still make the robot move along the x-axis in the case of two motors broken, which has never appeared in the training set. All these prove the strong generalization ability of the dynamic model in the proposed method, which is also the reason why the proposed method can make the robot keep the original motion trajectory under the robot damage conditions.

V. CONCLUSIONS

This paper proposes a model-based reinforcement learning method for robot damage recovery, which mainly consists of two parts: an ensemble-based dynamics model enhanced by future trajectory and damage information, and a Monte Carlo particle-based planning method. The introduction of the future trajectory and damage information improves the generalization of the dynamic model. The combination of the two parts enables the proposed method to keep the original motion trajectory of the robot even when one or two motors are damaged. Furthermore, the proposed method can also be extended to other legged robots which has different configurations from the mentioned robot. In the future, we will apply this method to the real robot environment. However, due to the existence of sensor noise, there will be a gap between the simulation and the real world, which is also a problem we need to solve.

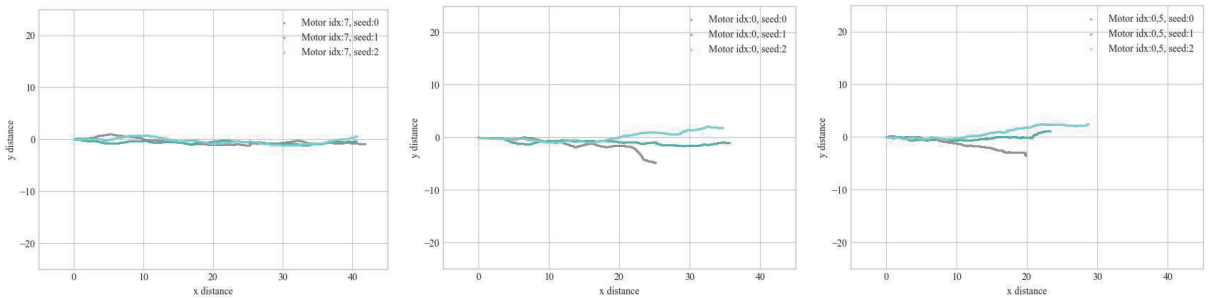
VI. ACKNOWLEDGEMENT

This work was supported by the National Nature Science Foundation of China under Grant (62173293), the Fundamental Research Funds for the Zhejiang Provincial Universities (2021XZZX021), Science and Technology on Space Intelligent Control Laboratory (2021-JCJQ-LB-010-13), and Zhejiang Provincial Natural Science Foundation of China (LD22E050007).

REFERENCES

[1] T. Fukuda, "Cyborg and bionic systems: Signposting the future," *Cyborg and Bionic Systems*, vol. 2020, 2020.

[2] D. Xu and Q. Wang, "Noninvasive human-prosthesis interfaces for locomotion intent recognition: A review," *Cyborg and Bionic Systems*, vol. 2021, 2021.



(a) Training Set (b) Testing Set (one motor damaged) (c) Testing Set (two motor damaged)

Fig. 5: The motion trajectory of the damaged robot under three sets of random seeds. (a) represents the training set results, (b) represents the results of the testing set, in which case one motor of the robot is damaged, (c) represents the testing set results, and two motors of the robot are damaged at this time. It can be seen that in the same time step, the robots in the training set have traveled the farthest distance, in which all three groups have reached a position of $40m$, and the path is relatively straightforward. When one motor is damaged, the robot's movement is shortened, two groups reach the position of $35m$, one group only reaches the position of $25m$. When two motors were damaged, the average distance reached by the three groups is between 20-30m.

[3] V. Verma, G. Gordon, R. Simmons, and S. Thrun, "Real-time fault diagnosis [robot fault diagnosis]," *IEEE Robotics & Automation Magazine*, vol. 11, no. 2, pp. 56–66, 2004.

[4] W. G. Fenton, T. M. McGinnity, and L. P. Maguire, "Fault diagnosis of electronic systems using intelligent techniques: A review," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 31, no. 3, pp. 269–281, 2001.

[5] J. Bongard, V. Zykov, and H. Lipson, "Resilient machines through continuous self-modeling," *Science*, vol. 314, no. 5802, pp. 1118–1121, 2006.

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

[7] C. Antoine, C. Jeff, T. Danesh, and M. Jean-Baptiste, "Robots that can adapt like animals," *Nature*, vol. 521, no. 7553, pp. 503–507, 2015.

[8] W. Huang, I. Mordatch, and D. Pathak, "One policy to control them all: Shared modular policies for agent-agnostic control," in *International Conference on Machine Learning*. PMLR, 2020, pp. 4455–4464.

[9] 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.

[10] R. Raileanu, M. Goldstein, A. Szlam, and R. Fergus, "Fast adaptation to new environments via policy-dynamics value functions," in *International Conference on Machine Learning*. PMLR, 2020, pp. 7920–7931.

[11] 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)*. IEEE, 2021, pp. 109–115.

[12] M. Janner, J. Fu, M. Zhang, and S. Levine, "When to trust your model: Model-based policy optimization," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[13] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel, "Model-ensemble trust-region policy optimization," *arXiv preprint arXiv:1802.10592*, 2018.

[14] Y. Du and K. Narasimhan, "Task-agnostic dynamics priors for deep reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2019, pp. 1696–1705.

[15] W. Whitney, R. Agarwal, K. Cho, and A. Gupta, "Dynamics-aware embeddings," *arXiv preprint arXiv:1908.09357*, 2019.

[16] I. Lenz, R. A. Knepper, and A. Saxena, "Deepmpc: Learning deep latent features for model predictive control." in *Robotics: Science and Systems*, vol. 10. Rome, Italy, 2015.

[17] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and stabilization of complex behaviors through online trajectory optimization," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 4906–4913.

[18] 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*. PMLR, 2020, pp. 1–10.

[19] A. Iscen, K. Caluwaerts, J. Tan, T. Zhang, E. Coumans, V. Sindhwani, and V. Vanhoucke, "Policies modulating trajectory generators," in *Conference on Robot Learning*. PMLR, 2018, pp. 916–926.

[20] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," *Advances in neural information processing systems*, vol. 31, 2018.

[21] B. Chen, Z. Liu, J. Zhu, M. Xu, W. Ding, L. Li, and D. Zhao, "Context-aware safe reinforcement learning for non-stationary environments," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 10 689–10 695.

[22] T. Anne, J. Wilkinson, and Z. Li, "Meta-reinforcement learning for adaptive motor control in changing robot dynamics and environments," *arXiv preprint arXiv:2101.07599*, 2021.

[23] A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, and P. Battaglia, "Graph networks as learnable physics engines for inference and control," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4470–4479.

[24] L. Wang, L. Meng, R. Kang, B. Liu, S. Gu, Z. Zhang, F. Meng, and A. Ming, "Design and dynamic locomotion control of quadruped robot with perception-less terrain adaptation," *Cyborg and Bionic Systems*, vol. 2022, 2022.

[25] T. Li, N. Lambert, R. Calandra, F. Meier, and A. Rai, "Learning generalizable locomotion skills with hierarchical reinforcement learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 413–419.

[26] D. Jain, A. Iscen, and K. Caluwaerts, "Hierarchical reinforcement learning for quadruped locomotion," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 7551–7557.

[27] A. G. Kupcsik, M. P. Deisenroth, J. Peters, and G. Neumann, "Data-efficient generalization of robot skills with contextual policy search," in *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[28] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 5026–5033.

[29] J. V. Dillon, I. Langmore, D. Tran, E. Brevdo, S. Vasudevan, D. Moore, B. Patton, A. Alemi, M. Hoffman, and R. A. Saurous, "Tensorflow distributions," *arXiv preprint arXiv:1711.10604*, 2017.

[30] D. S. SolidWorks, "Solidworks®," *Version Solidworks*, 2005.