Autonomous Locomotion of a Rat Robot Based on Model-free Reinforcement Learning

Zitao Zhang[1], Yuhong Huang[2], Zijian Zhao[1], Zhenshan Bing[2], Chenglin Cai[3], Alois Knoll[2] and Kai Huang[1,*]

***Abstract*— The rat robot is a soft compact quadrupedal robot with the same size as real rats. It is difficult for such robots to learn effective motions on complex terrain owing to their underactuated nature and limited sensors. This paper proposes a novel approach for the rat robot to learn adaptive motion on rugged terrain based on reinforcement learning. The training architecture is designed for the rat robot's nonlinear control structure. In order to improve perceptual efficiency, we gather and compress perception information based on sensor data observations in time clusters during robot walking. Our proposed method demonstrates excellent exploration of complex effector space and nonlinear dynamics of the rat robot to adapt to challenging terrain. We evaluate the efficacy of our approach on a varied set of scenarios, which include various obstacles and terrain undulations and physical validation is performed. Our results show that our approach effectively achieves efficient motions on complex terrains designed for small-sized robots and outperforms other benchmark algorithms.**

I. Introduction

Rat robot is a kind of bionic quadrupedal robot with the same size as real rats, which has a compact structure together with flexible rope-driven tendons. Locomotion of this kind of compact bionic robot is worthy investigating, for not only better understanding the nature but also special robot applications. Legged robots show greater competitiveness than their wheeled counterparts in unstructured complex environments [1]. Compared with larger quadruped robots, small-sized robots [2] exhibit increased flexibility, making them well-suited for navigating constricted spaces due to their smaller size, lower weight, and reduced cost. Fig. 1 shows the rat robot developed on the basis of our previous work [3], [4].

Motion abilities of the rat robot require drugging with state-of-art methods. Fig. 2 shows the end-effector space of legs based on our previous kinematics calculation [5], [6] , which needs huge efforts for conventional methods to fully explore. Meanwhile, deep reinforcement learning (DRL) has gained attention in legged locomotion recently. Conventional methods acquire manual effort to model both the robotics system and the target task scenario, often depending on the experience of designers. In contrast, reinforcement learning is a viable approach for generating robot motions for complex terrain by analyzing the robot status during walking [7]. However, DRL for small-sized quadrupedal robots like rat robots has not been widely studied.

*Corresponding author, email: huangk36@mail.sysu.edu.cn

[1]Authors from the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China.

[2]Authors from the Department of Informatics, Technical University of Munich, Munich, Germany.

[3]Authors from 01.AI, Beijing, China.

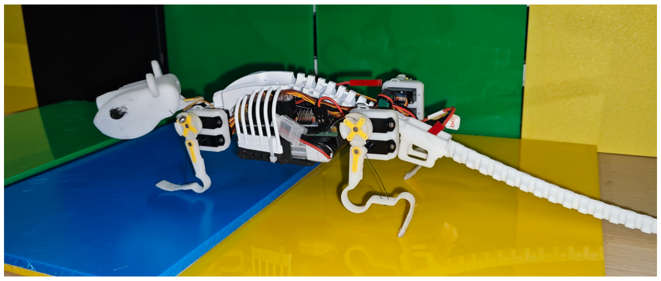
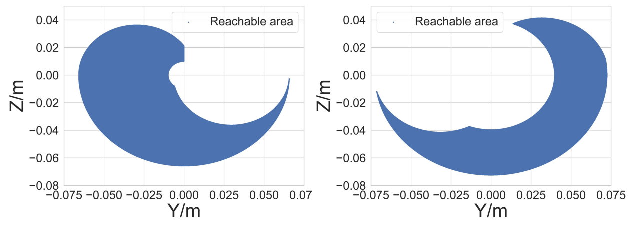


Fig. 1. Rat robot, a small-sized quadruped robot with soft rope-driven legs.



(a) Front Leg (b) Hind Leg

Fig. 2. Reachable range of legs in the end-effector space. The Cartesian Y-Z plane undertakes the motion space of the rope-tendon driven leg.

Locomotion based on reinforcement learning is challenging because of nonlinear robotic dynamics and weak feedback from limited sensors. On one hand, the soft structure and rope-driven tendons theoretically poss an infinite number of digrees of freedom(DOF) [8]. It contributes to complex time-serious relationship between control signals and environmental feedback. On the other hand, due to limitations in size, weight, area, and power (SWAP), conventional sensors are frequently unfeasible for small-sized robots [9], which is critical for common-sized robotics locomotion. Light-weight sensors acquire for additional processing to attain comparable levels of perception to higher-quality counterparts [10]. Therefore, actions and observations among timesteps are supposed to be considered together. And further process of time-serious sensor data is significant to motion learning.

In this paper, we propose a new method to conduct autonomous locomotion of the rat robot to overcome complex terrain based on reinforcement learning. This approach is aimed at overcoming the limit of light-weight sensors and structure challenge on the rat robot to better explore motion potential of small-sized soft robots. The normalized action is generated with a policy network to cover certain timesteps in order to obtain stronger physical interaction. Based on the observations from light-weight sensors of the robot, we design a time cluster to filter

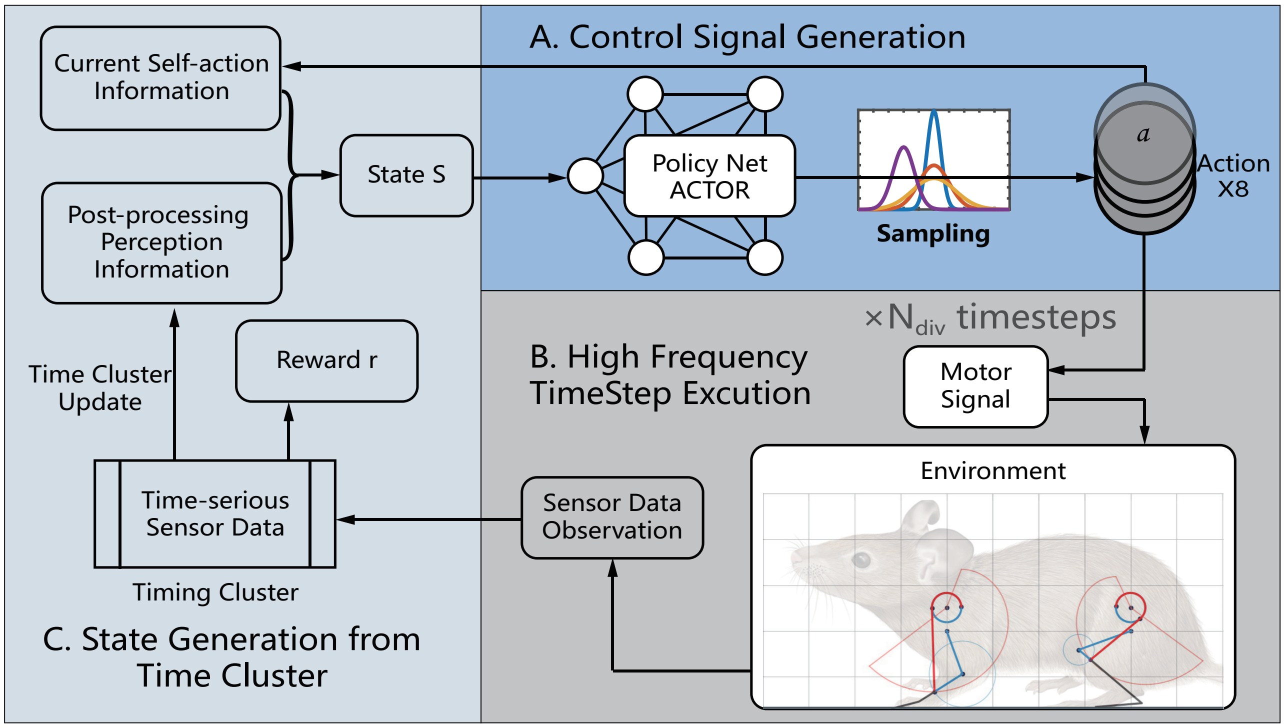

A. Control Signal Generation
Current Self-action Information
Post-processing Perception Information
State S
Policy Net ACTOR
Sampling
a
Action X8
$\times N_{div}$ timesteps
B. High Frequency TimeStep Excution
Motor Signal
Environment
Reward r
Time Cluster Update
Time-serious Sensor Data
Sensor Data Observation
Timing Cluster
C. State Generation from Time Cluster

Fig. 3. Architecture: Part A implements a reinforcement learning method to generate action signals. Part B is the execution of motor signals within a high frequency time-serious piece. Part C processes the sensor data observations and generates the reward and the state of a markov node.

noise of time-series observations during interaction with the environment. The state is generated based on the processed observations. Besides, we design the reward function with three factors designed for adapting to the robot motion scenarios. Experiments are conducted on four challenging scenarios and the results prove the validation of our method. Main contributions of this work are summarized as follows:

- We design the action of the rat robot generated by a policy network covering several timesteps. Complex end effector space and nonlinear dynamics are explored with reinforcement learning to adapt to challenging terrain.
- We propose a time cluster updating method to process sensor data observations among high-frequency actions. Effective states are generated to reinforce perceptual feedback, contributing to improved training efficiency.

II. Related Work

DRL utilizes the feature representation ability of deep learning to strengthen the decision-making ability of agents, contributing to outstanding end-to-end control performance [11]. Despite achievements that reinforcement learning has made, there are still gaps between robotic control tasks and benchmark problems in reinforcement learning [12]. Recent new methods tend to combine model-free and model-based approaches [13], leveraging predetermined models of system dynamics to guide model-free learning. Lee et al. [7] presented a proprioceptive feedback method that improved model robustness. Haojie et al. [14] made use of preset motion as reference for policy learning. An optimization model based on the designed evolutionary trajectory generator was proposed to adjust the shape of final motion trajectory. Sanket et al. [15] developed a probabilistic MPC method that considers model uncertainty in long-term predictions, leading to improved accuracy. These methods offer a promising direction for model-free methods by incorporating additional information.

However, most approaches are designed for full-sized robots such as the MIT Cheetah [16] and ANYmal [17]. One our previous work [18] proposed a hierarchical model-based method to train adaptive locomotion of a rat robot. Small-sized robots, facing constraints such as limited sensor information, low-powered micro-controllers, and computing resources [2], pose unique challenges to DRL. Many current approaches rely on expensive graphics cards and high-perception tactile sensors to build an altitude map of the environment, which may not be applicable or effective for small-sized robots.

III. Architecture Overview

Fig. 3 shows the architecture of the proposed method. First of all, we analyze the structure of the rat robot and give the definition of action in locomotion, which is a 8-Dimension vector in the part A Control Signal Generation. In our architecture the actions are sampled from a policy distribution, generated by a policy network. With the rat robot's equipment limits taken into account, the sensor data observation within a timestep is available. Secondly, on the basis of control signal execution with several timesteps in the part B High Frequency Timestep Execution, a time cluster is designed to filter noise and conduct post-process to improve perceptual efficiency. A integrated state is then established as the part C shows. Besides, we design the reward function

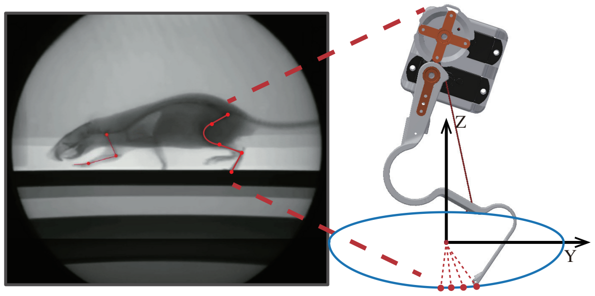


Fig. 4. Structure of the bionic leg, consisting of an elastic limb, a driving tendon and two actuating motors. End point of the limb moves in the 2-Dimension end-effector space, driven by two actuators. For the flexible leg driven by dual motors, the Cartesian Y-Z plane undertakes the end-effector motion space. Schematic diagram of the leg movements is from [4].

in Section IV-C. With markov nodes generated including actions, states and rewards, learning iterations are performed to update the policy and then train the robot to overcome rugged terrain.

IV. Methodology

A. Action and Observation Space of the Rat Robot Platform

The rat robot based on our previous work is a type of small-sized bio-inspired robot, designed with soft actuated components that differ from dog-size quadrupedal robots. The bio-inspired legs of our robot are designed to closely mimic the body structure and movement pattern of natural mice, while the flexible structures pose challenges to traditional quadrupedal motion control methods, particularly when reinforcement learning is implemented. As Fig. 4 shows, each leg of the rat robot is driven by rope-tendon driven servos on upper side with the range of 180 degrees for each, activated by PWM signals. The direct control signal for one timestep can be denoted as $\mathbf{q} \in \mathbb{R}^8_{clip}, \mathbb{R}^8_{clip} := [-\frac{\pi}{2}, \frac{\pi}{2}]$. We design the action as a set of normalized servo angles conducted in several timesteps, which id defined as

$$\mathbf{a}_t = \{\frac{2}{\pi}\mathbf{q}\}^{N_{step}\times 8} \tag{1}$$

where N_{step} is the number of timesteps that one single action is conducted in. Besides, we get the normalized range $\mathbf{a}_t \in \mathbb{R}^8_{clip}, \mathbb{R}^8_{clip} := [-1, 1]$.

As part A of Fig. 3 shows, we implement a policy network to generate actions. The Control policy is parameterized as a Gaussian distribution $\mathcal{N}(\mu, \sigma)$ with a diagonal covariance matrix:

$$\pi_\theta(a \mid s_t) = \mathcal{N}(a \mid \mu_\theta(s_t), \sigma_\theta). \tag{2}$$

where θ is the parameter of the policy and s_t is the state in the markov node, which is illustrated in Section IV-B. A deep neural network (DNN) structure is employed to generate the mean $\mu_\theta(s_t)$ of the distribution. Additionally, the standard deviation σ_θ is generated by a separate and independent network layer, which facilitates exploration during training. The complete algorithm is outlined in Algorithm 1.

Observation of the rat robot is determined by the sensors. In addition to structural peculiarities, limited sensor equipment affects perception. The main sensor of the rat robot is an IMU at its body center, which provides 3-axis acceleration and 3-axis angular velocity $o_{IMU} = \{\mathbf{a}, \mathbf{g}\}$. With calculation of the integrated library, the three-axis velocity and quaternion can be obtained $o'_{IMU} = \{\mathbf{v}, \mathbf{Q}\}$.

Algorithm 1 Model-free Motion Learning with Rat Robot

```
Require: $S_0$, State with initial environment
    $\beta$, experience replay buffer
    $N_{step}$, the number of time steps a time cluster contains
    $\epsilon \sim \mathcal{N}(0,1)$, Gaussian noise for exploration
 1: Initiate the policy
 2: while Not Converged do
 3:     Get normalized action signal $a_t \leftarrow \pi_\theta(s) + \epsilon$
 4:     for $i = 1$ to $N_{step}$ do    ▷ Generate one time cluster
 5:         Conduct motor control $(q_1, q_2)$ for each leg
 6:         Get sensor data of one single timestep
 7:         Append sensor data into Timing Pool
 8:     end for
 9:     Generate State $s$ of current time cluster
10:     Append the transition $(s_t, a_t, r, s_{t+1})$ into $\beta$
11:     if $\beta$ is full then
12:         Update Policy
13:         Reset $\beta$
14:     end if
15: end while
```

Taking into account our development progress on the physical robot, we assume that 4 foot pressure sensors are available. In reality, the accuracy of pressure readings from foot sensors is low and unstable due to the limitations of physical device performance. Without considering the dynamical characteristics, we focus on the state of whether each leg is in contact with the ground or not. Therefore, the pressure sensor reading is converted to a Boolean value as $c_{F,k} = \{0, 1\}$, where $k = 1, 2, 3, 4$ refers to four legs. To summarize, the observation space of the rat robot can be defined as

$$o_t = \{\mathbf{v}, \mathbf{Q}, c_{F,k}\} \tag{3}$$

B. State Generation with Time Cluster Updating

On the basis of the observation space, we design a post-process of sensor data observations among timesteps, which is aimed at reducing the dimensionality of complex environment observations. A data pool of length N_{step} is set to store sensor data observations among N_{step} timesteps, referred to as a "time cluster". As part B of Fig. 3 shows, an RL step comprises a segment of servo signal conduction within N_{step} timesteps. Part C describes the process of state generation, as detailed by Fig. 5.

During the execution of actions within a time cluster, sensor data is collected for further process to obtain valid sensory information with filtered noise. We use the mean filter to process the sensor information. The processing of the three-axis velocity information is as follows:

$$\mathbf{V}^i = \sum_{k=0}^{N_{step}} \frac{\mathbf{v}^i_k}{N_{step}} \tag{4}$$

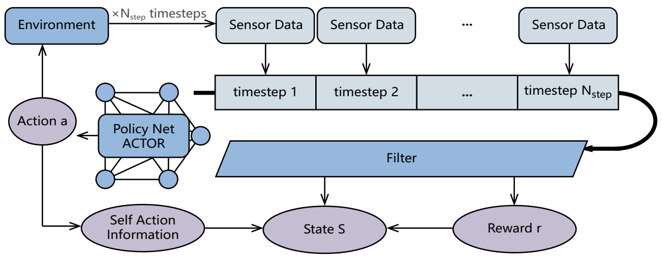


Fig. 5. Process of State Node Generation within a Time Cluster

where i from 1 to 3 corresponds to three directions, and v_k^i is the kth sensor data of velocity within a time cluster. In a continuous run, the computation at the end of a time cluster needs to be reduced in order to balance the computation time within each timestep for real-world execution. We modify Equation 4 into an incremental form in order to update the amount of state information in real time:

$$\mathbf{V}_k^i = \frac{k-1}{k}\mathbf{V}_{k-1}^i + \frac{1}{k}\mathbf{v}_k^i. \tag{5}$$

The processing of the angular velocity follows the same approach. Note that the selection of filter types is not the focus of our research, where other filters can be used as well. As for the foot pressure sensor, $c_{F,k} = 0$ only when sensor readings are all 0 in a time cluster.

After generation of a Timing Cluster, the processed perceptual information together with the information of its own action in the current RL step form the State s in the Markov node

$$s_t = \left(a_t, r, \vec{V_{vel}}, \vec{V_{gyro}}, \vec{V_Q}, \mathbf{c}_F\right) \tag{6}$$

where r is the reward value, $\vec{V_{vel}} \in \mathbb{R}^3$ is the filtered velocity, $\vec{V_{gyro}} \in \mathbb{R}^3$ is the filtered angular velocity, and $\vec{V_Q} \in \mathbb{R}^4$ is the filtered quaternion. Acceleration is not included because experimental experiences show that the calculated velocity performs better.

C. Reward Function

In order to traverse challenging terrain among several scenarios and avoid getting trapped, the reward function is designed as follows:

$$r = r_{forward} + r_{trap} + r_{energy} \tag{7}$$

which consists of three items considering different factors. $r_{forward}$ is the main part of the reward function aimed to drive the robot to move ahead, which is defined as

$$r_{forward} = \begin{cases} W_{vel}\vec{V_{vel}} \cdot \vec{u_{dir}} & \vec{V_{vel}} \cdot \vec{u_{dir}} < T_{vel} \\ -W_{vel} & else \end{cases} \tag{8}$$

where W_{vel} is the weighting factor, $\vec{u_{dir}}$ is a unit vector for direction. T_{vel} is the threshold of velocity because a overlarge velocity correspond to the robots falling down in an unstable situation. Second factor r_{trap} punishes overturning and causes terminating one episode, which is defined as

$$r_{trap} = \begin{cases} 0 & n_{air} < 10\ (about 0.5s) \\ -K_{trap} & else \end{cases} \tag{9}$$

where n_{air} is the number of sustained states when all four feet are off the ground at the same time. n_{air} is calculated as follows:

$$n_{air} = \begin{cases} 0 & \Sigma_{k=1}^4 c_{F,k} > 0 \\ n_{air}+1 & else \end{cases} \tag{10}$$

The last factor r_{energy} is

$$r_{energy} = W_e \Sigma_{i=1}^8 |a_{i,t} - a_{i,t-1}| \tag{11}$$

where W_e is the weighting factor. It is designed to improve coherence between movements and reduce drift.

V. Experiments

This section provides a detailed description of the experimental setup employed to assess the performance of the proposed method. The motion of the rat robot is analyzed to demonstrate the effectiveness of our proposed method. Subsequently, we elaborate on the efficiency of the algorithm.

A. Experimental Setup

Four terrain scenarios are constructed for the purpose of training the rat robot to navigate, as illustrated in Fig. 6. The size of the terrain scenarios is designed to simulate the real-world environment encountered by small robots. Table I shows the hyper parameters in the experiment.

The rat robot is a small quadrupedal robot that has dimensions of 40 cm × 25 cm and features 8 degrees of freedom for control. Explorable end effector space of each leg is shown in Fig. 2. Using the physical platform utilized in our prior research, as depicted in Fig. 1, a digital twin is created.

The simulations are carried out using the MuJoCo robot simulation platform [19], operating on a Ubuntu 20.04 system. The training machine is a server equipped with dual Intel Xeon Gold 6234 Processors. Each training session is allocated one physical core.

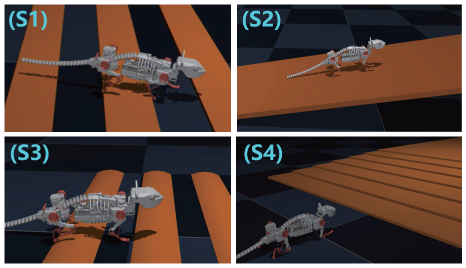


Fig. 6. Four different testing scenarios: (S1)Planks: gallop over planks with gaps of 10 cm width. (S2)Uphill: a slope 10-degree. (S3)Logs: a pile of logs. (S4)Stairs: small stairs between platforms of different heights.

TABLE I
Hyper-Parameters

Parameter	Symbol	Value
Length of a time cluster	N_{step}	5
Batch Size	N_B	2048
Maximum timestep of one episode	E_{ts}	2000
Weight of $r_{forward}$	W_{vel}	5.0
Weight of r_{trap}	K_{trap}	5.0
Weight of r_{energy}	W_e	0.1

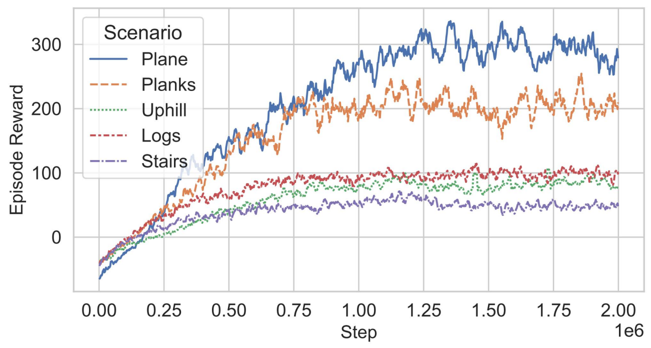


Fig. 7. Training curves on 5 simulation tasks.

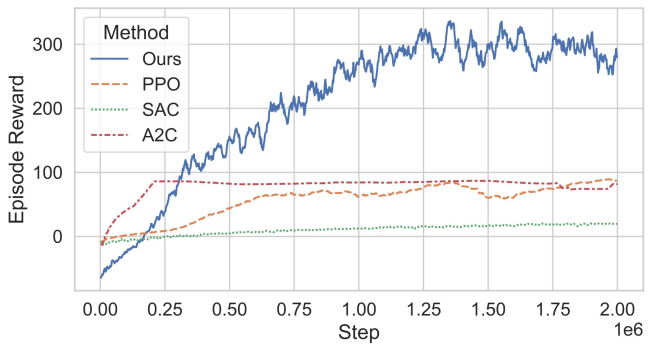


Fig. 8. Training curves of different methods on the plane.

TABLE II
PERFORMANCE IN FIVE SCENARIOS*

Scenario name	Success rate (%)	Episode reward
Plane	–	332.9
(S1)Planks	72	210.3
(S2)Uphill	80	98.6
(S3)Logs	78	102.1
(S4)Stairs	86	99.4

* Experiment of each scenario is repeated for 50 times.

B. *Performance in Different Scenarios*

The rat robot successfully completed all four tasks with policies learned from scratch. Fig. 7 shows the training curves in the four scenarios together with the basic plane scenario. All five policy agents converge successfully with different episode rewards. That results from blockage of rugged terrain in different scenarios. The landscape in the basic plane scenario is simple and relatively easy to pass through, resulting in a larger converged episode reward. The scenario of stairs seem to be most difficult so the episode reward is lower than others. It consists of both rugged terrain on the horizontal plane and height difference of slope, which is in some way a combination of (S1) and (S2). Validation experiments are performed by using trained policies to drive the robot and render the screen for observation. Results are consistent with the training curves.

Table II shows the performance metrics of different scenarios. We made 50 attempts in each scenario with respective trained policies. Performance of (S1), (S3) and (S4) is consistent with their training curves. It is interesting that although the converged episode reward of (S2) is not the smallest of five scenarios, success rate of (S2) performs relatively small. Such phenomenon indicates that height variation is more challenging for the rat robot's movement, needing to be further studied.

C. *Comparison With benchmarks*

In order to validate the proposed method of this work, three classical benchmark algorithms are presented here for comparison. The proposed architecture is based on the policy optimization reinforcement learning. Considering that, we pick there model-free reinforcement learning algorithms to be compared with our method: A2C, PPO and SAC. To make fair comparisons, the benchmark algorithms use the same action and reward design as our method, but the core technique of the time cluster design is not implemented. As a result, the uncompressed timesteps make the episode steps of benchmark algorithms N_{step} times the length of our method. In comparison, episode rewards of them are divided by N_{step} inversely. The implementation of these benchmark algorithms is based on the Stable-baselines3 Platform [20].

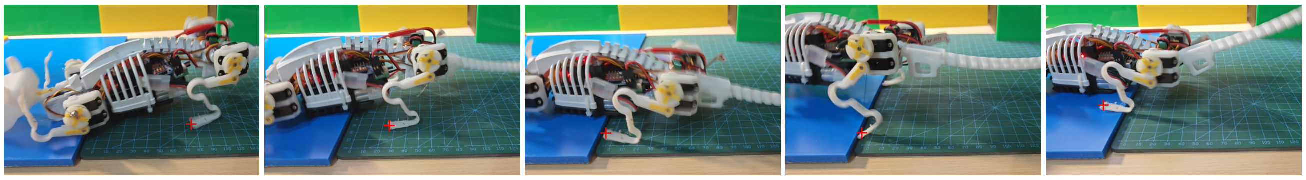
As shown in Fig. 8, the proposed method significantly outperforms the benchmark approaches just on the plane. All the policy has to learn the parameters from scratch. Although benchmark algorithms are also able to converge, the over low converged episode rewards on the plane suggest that they are unable to generate forward motion to drive the robot. That is because they all fall into the local optimum, failing to fully explore the action space. Scenario rendering shows that three benchmark algorithms just make the robot struggle in place. In four challenging scenarios the conclusions are consistent. That is because general reinforcement learning methods cannot learn from the weak sensor feedback within a simple short timestep. In conclusion, the compression of states is necessary for improving training efficiency.

D. *Validation in the real world*

To validate the effectiveness of our approach, we implement experiments on the real physical robot. In a scenario where 5 mm stairs are set ahead to get through, tests on the rat robot are performed with both our approach and regular control method of trotting gait. Our approach helps the robot succeed in running across the the terrain while the basic trotting gait fails. Fig. 9 shows the montage of our robot running across the staircase. For Further discussion, we compare the performance of our method with that of a basic trotting gait and our previous work of a hierarchical DRL control method [18]. As Fig. 10 shows, the proposed approach performs the fastest speed. Due to a more direct control policy compared with our previous hierarchical DRL control method, the execution time for motion calculation has been reduced while maintaining the ability to get through unstructured terrain.

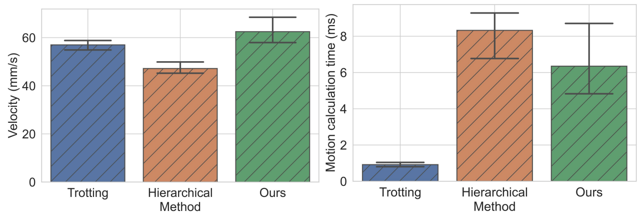
VI. CONCLUSIONS

This paper proposes a reinforcement learning method to generate motion of the rat robot for unstructured terrain. We design a reinforcement learning architecture to explore the nonlinear kinematics and dynamics of the small, flexible rat



(a) t = 0s (b) t = 2s (c) t = 4s (d) t = 6s (e) t = 8s

Fig. 9. Montage of the physical robot running across a 5 mm high staircase.



(a) Velocity (b) Motion calculation time

Fig. 10. Performance comparison in a plane in the real world. The trotting gait is set to the maximum frequency for normal forward motion. Method Hierarchical refers to our previous work [18].

robot. A time cluster is designed to execute post-processing of time-series data during the robot's interaction with the environment. The effective state is generated based on the compressed observations and self-action status, improving the training efficiency. A multi-factor reward function is formulated to adapt to complex terrain. We design experiments with four rugged terrain scenarios to validate the performance of the proposed method and it is compared with benchmark algorithms to prove effectiveness of the time cluster. Experiments show that the proposed method makes success in all challenging scenarios and outperforms other benchmark algorithms.

VII. Acknowledgments

This work has received funding from Pazhou Laboratory Grants with No. PZL2021KF0020, Fundamental Research Funds for the Central Universities, Sun Yat-sen University with No. 23qnpy30/67000-31610023, and Guangxi Key R & D Program with No. GuikeAB24010324.

References

[1] C. Yao, G. Shi, Y. Ge, Z. Zhu, and Z. Jia, "Predict the physics-informed terrain properties over deformable soils using sensorized foot for quadruped robots," in *2023 International Conference on Advanced Robotics and Mechatronics (ICARM)*, 2023, pp. 330–335.

[2] S. M. Neuman, B. Plancher, B. P. Duisterhof, S. Krishnan, C. Banbury, M. Mazumder, S. Prakash, J. Jabbour, A. Faust, G. C. de Croon, and V. J. Reddi, "Tiny robot learning: Challenges and directions for machine learning in resource-constrained robots," in *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, 2022, pp. 296–299.

[3] P. Lucas, S. Oota, J. Conradt, and A. Knoll, "Development of the neurorobotic mouse," in *2019 IEEE International Conference on Cyborg and Bionic Systems (CBS)*. IEEE, 2019, pp. 299–304.

[4] Z. Bing, A. Rohregger, F. Walter, Y. Huang, P. Lucas, F. O. Morin, K. Huang, and A. Knoll, "Lateral flexion of a compliant spine improves motor performance in a bioinspired mouse robot," *Science Robotics*, vol. 8, no. 85, p. eadg7165, 2023.

[5] Y. Huang, Z. Bing, F. Walter, A. Rohregger, Z. Zhang, K. Huang, F. O. Morin, and A. Knoll, "Enhanced quadruped locomotion of a rat robot based on the lateral flexion of a soft actuated spine," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 2622–2627.

[6] Y. Huang, Z. Bing, Z. Zhang, K. Huang, F. O. Morin, and A. Knoll, "Smooth stride length change of rat robot with a compliant actuated spine based on cpg controller," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

[7] 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, Oct. 2020, publisher: American Association for the Advancement of Science.

[8] D. Rus and M. T. Tolley, "Design, fabrication and control of soft robots," *Nature*, vol. 521, no. 7553, pp. 467–475, May 2015. [Online]. Available: https://doi.org/10.1038/nature14543

[9] B. P. Duisterhof, S. Krishnan, J. J. Cruz, C. R. Banbury, W. Fu, A. Faust, G. C. H. E. de Croon, and V. Janapa Reddi, "Tiny robot learning (tinyrl) for source seeking on a nano quadcopter," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 7242–7248.

[10] P. Fankhauser, M. Bjelonic, C. Dario Bellicoso, T. Miki, and M. Hutter, "Robust rough-terrain locomotion with a quadrupedal robot," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 5761–5768.

[11] X. Wang, S. Wang, X. Liang, D. Zhao, J. Huang, X. Xu, B. Dai, and Q. Miao, "Deep reinforcement learning: A survey," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2022.

[12] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[13] H. Nguyen and H. La, "Review of deep reinforcement learning for robot manipulation," in *2019 Third IEEE International Conference on Robotic Computing (IRC)*. IEEE, 2019, pp. 590–595.

[14] H. Shi, B. Zhou, H. Zeng, F. Wang, Y. Dong, J. Li, K. Wang, H. Tian, and M. Q.-H. Meng, "Reinforcement Learning With Evolutionary Trajectory Generator: A General Approach for Quadrupedal Locomotion," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3085–3092, Apr. 2022, conference Name: IEEE Robotics and Automation Letters.

[15] S. Kamthe and M. Deisenroth, "Data-efficient reinforcement learning with probabilistic model predictive control," in *International conference on artificial intelligence and statistics*. PMLR, 2018, pp. 1701–1710.

[16] D. J. Hyun, S. Seok, J. Lee, and S. Kim, "High speed trot-running: Implementation of a hierarchical controller using proprioceptive impedance control on the mit cheetah," *The International Journal of Robotics Research*, vol. 33, no. 11, pp. 1417–1445, 2014.

[17] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, R. Diethelm, S. Bachmann, A. Melzer, and M. Hoepflinger, "Anymal - a highly mobile and dynamic quadrupedal robot," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016, pp. 38–44.

[18] Z. Zhang, Y. Huang, Z. Zhao, Z. Bing, A. Knoll, and K. Huang, "A hierarchical reinforcement learning approach for adaptive quadruped locomotion of a rat robot," in *2023 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2023, pp. 1–6.

[19] 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*, 2012, pp. 5026–5033.

[20] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, "Stable-baselines3: Reliable reinforcement learning implementations," *J. Mach. Learn. Res.*, vol. 22, no. 1, jan 2021.