Autonomous Locomotion of a Rat Robot Based on Reinforcement Learning

Zitao Zhang[1], Yuhong Huang[2], Zijian Zhao[1], Zhenshan Bing[2], and Kai Huang[1]

[1] Sun Yat-Sen University, Guangzhou, China,
zhangzt9@mail2.sysu.edu.cn
[2] Technical University of Munich, Munich, German

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. We gather and analyze perception information based on changes in time slices to monitor environmental changes during robot walking. Our proposed framework demonstrates a significant reduction in training convergence time, from millions to hundreds of thousands, compared to commonly used reinforcement learning methods. We evaluate the efficacy of our approach on a varied set of simulated terrain scenarios, which include various obstacles and terrain undulations. Our results show that our approach effectively achieves efficient motions on complex terrains designed for small-sized robots.

Keywords: robotics, autonomous control, reinforcement learning

1 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. Leg structure of quadrupedal robots make them more competitive than wheeled robots in rugged terrains and complex environments. Furthermore, compared with larger dog-sized quadrupedal robots such as ANYmal[2], small robots[3] 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 digital twin of the rat robot we use in this research, based on our previous work[1].

Motion potential of the rat robot acquires for drugging with state-of-art methods. Fig. 2 shows the end-effector space of legs, which needs huge efforts for conventional methods to fully explore. Meanwhile, deep reinforcement learning has gained attention in legged locomotion of quadrupedal robots in recent years.

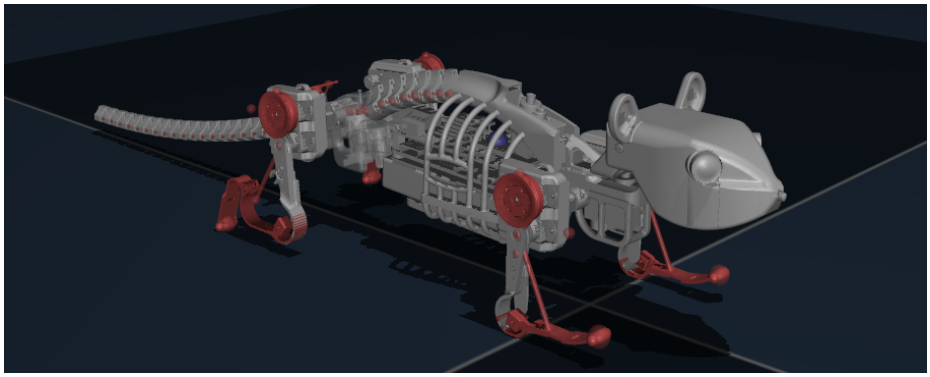
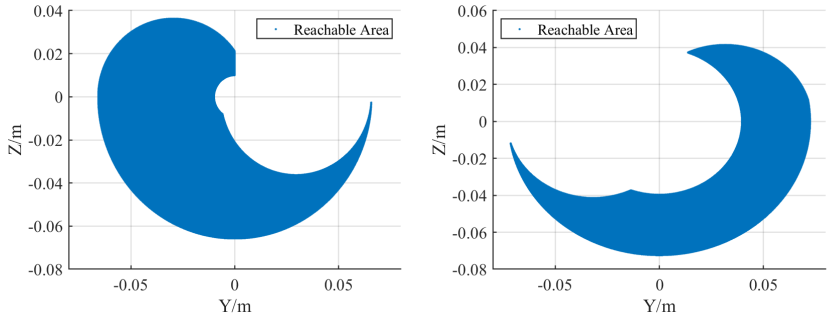


Fig. 1. The rat robot is designed with soft actuated components and supports more flexible robot motions.



(a) Front Leg (b) Hind Leg

Fig. 2. Reachable range of legs. For the flexible leg driven by dual motors, the motion space is located in the Y-Z plane of its own Cartesian coordinate system.

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 gaits for complex terrain by analyzing the robot status during walking. However, reinforcement learning for small-sized quadrupedal robots like rat robots has not been widely studied.

Locomotion based on reinforcement learning is however challenging because of nonlinear 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)[4]. It contributes to complex time-serious relationship between control signals and environmental feedback. For example, a 60 Hz motor signal takes dozens of executions to allow the robot to achieve a tiny movement of lifting a front claw. On the other hand, due to limitations in size, weight, area, and power (SWAP), conventional sensors are frequently unfeasible for small-

sized robots[5], which is critical for common-size robotics locomotion. Light-weight sensors acquire for additional processing to attain comparable levels of perception to higher-quality counterparts[6]. 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-size 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 noise of time-series observations during interaction with the environment. The state is generated based on the processed observations. Besides, we deign the reward function with three factors are designed to adapts 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 that covers a period of timesteps. Complex end effector space and nonlinear dynamics is explored with reinforcement learning to adapt to challenging terrain.
- We propose a time cluster updating method to process sensor data observations among high-frequency action conduction. Effective states generated from compromised sensor data observations reinforce perceptual feedback, contributing to improved training efficiency.

2 Related Work

Reinforcement learning, as an area of machine learning, aims to let an agent learn to get maximum long-term rewards with actions in the environment. Today deep reinforcement learning utilize the feature representation ability of deep learning to strengthen the decision-making ability of agents, contributing to outstanding end-to-end control performance[7]. Despite achievements that reinforcement learning has made, there are still gaps between robotic control tasks and benchmark problems in reinforcement learning[8]. For instance, continues states and actions makes information extraction more difficult. Besides, the inherent execution of physical systems results in frequent delays to be solved.

Generally, reinforcement learning methods can be categorized into model-free and model-based approaches [9]. Model-free methods are relatively simple to implement and allow for end-to-end control by sampling from the environment and generating control signals. Xie et al. [10] developed a novel method that can modify reward function simultaneously with training, while Lee et al. presented a proprioceptive feedback method that improved model robustness. However, these methods can be computationally intensive and time-consuming. Faced with this

problem, Hwangbo et al. [11] proposed a framework that combines simulation and real environments.

On the other hand, model-based methods require the construction of a virtual environment to facilitate on-policy learning. Although they can accelerate convergence speed, they often require significant computation. As a result, many new methods combine model-free and model-based approaches, leveraging predetermined models of system dynamics to guide model-free learning. These methods offer a promising direction for model-free methods by incorporating additional information. For instance, Haojie et al. [12] proposed an evolutionary trajectory generator-based model that optimizes the output trajectory's shape for a given task, providing diversified motion priors to guide policy learning. Sanket et al. [13] developed a probabilistic MPC method that considers model uncertainty in long-term predictions, leading to improved accuracy.

However, most reinforcement learning approaches have been designed for full-sized robots such as the MIT Cheetah [14], Anymal [2], and BigDog [15]. Small-sized robots, facing constraints such as limited sensor information, low-powered micro-controllers, and computing resources, pose unique challenges to reinforcement learning. 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.

3 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 in Section 4.3. 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.

.

4 Methodology

4.1 Action and Observation Space of the Rat Robot Platform

The rat robot based on our previous work is a type of small-scale 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

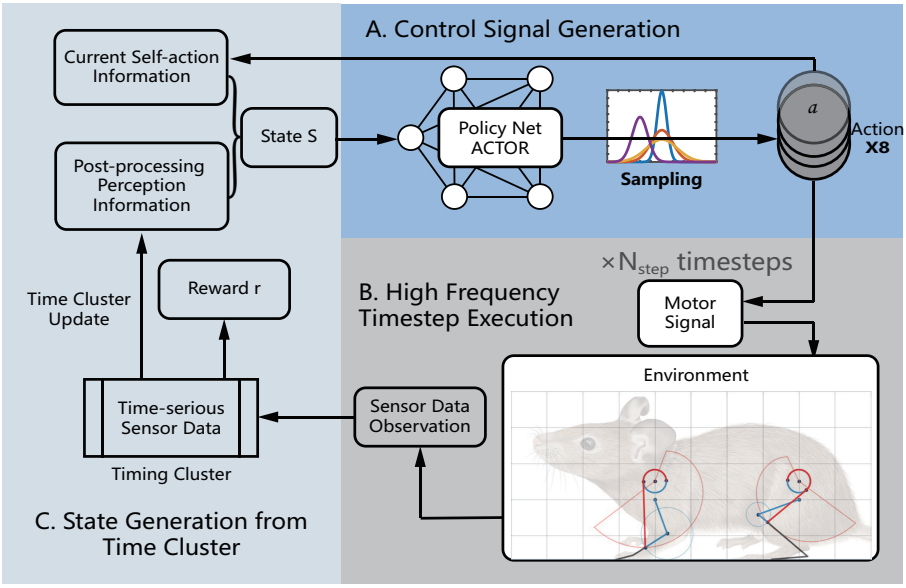


Fig. 3. Architecture: Part A implements a reinforcement learning method to generate action signals. Part B is the execution of motor signals within a high frequency time-serious piece. Part C processes the sensor data observations and generates the reward and the state of a markov node.

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 tendon ropes with two PWM servos on the hip. Each servo has a rotation range of 180 degrees. As a result, the direct control signal in a single time step is 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 with a diagonal covariance matrix.

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

A deep neural network (DNN) structure is employed to generate the mean $\mu_\theta\left(s_t\right)$ 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.

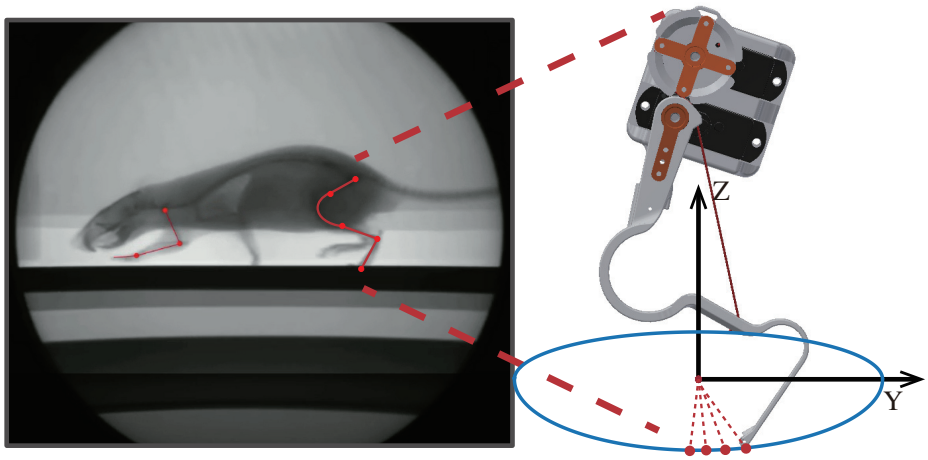


Fig. 4. Structure of the bionic leg, consisting of an elastic limb, a driving tendon and two actuating motors. End point of the limb moves in the 2-Dimension end-effector space, driven by two actuators. For the flexible leg driven by dual motors, the end-effector space is located in the Y-Z plane of its own Cartesian coordinate system. Schematic diagram of the leg movements is cited from [16].

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 $o'_{IMU} = \{\mathbf{v}, \mathbf{Q}\}$.

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

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

4.2 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".

Algorithm 1 RL with Rat Robot

Require:S_0,State with initial environment
β, experience replay buffer
α, learning rate
N_{step}, the number of time steps a time cluster contains
$\epsilon \sim \mathcal{N}(0, 1)$, Gaussian noise for exploration

Initiate the policy
while Not Converged **do**
  Get normalized action signal $a_t \leftarrow \pi_\theta(s) + \epsilon$
  for $i = 1$ to N_{step} **do** ▷ Generate One-Step time cluster
    Conduct motor control (q_1, q_2) for each leg
    Get sensor data observation of one single timestep
    Append sensor data into Timing Pool
  end for
  Generate State S of current time cluster
  Append the transition (s_t, a_t, r, s_{t+1}) into β
  if β is full **then**
    Update Policy
    Reset β
  end if
end while

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:

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

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:

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

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.

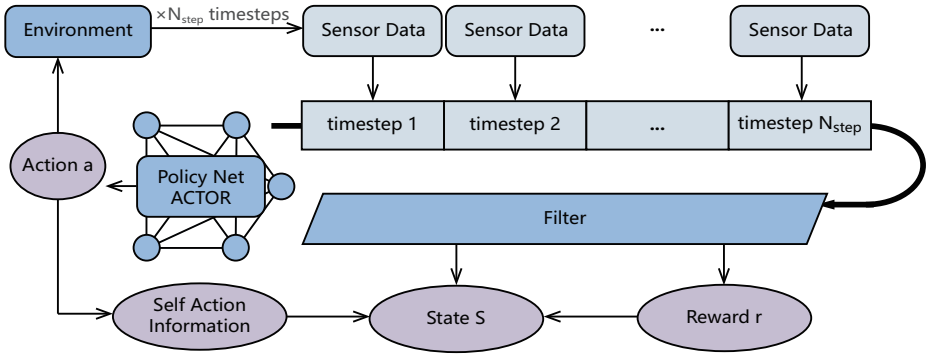


Fig. 5. Process of State Node Generation within a Time Cluster

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.

4.3 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.

5 EXPERIMENTS

This section provides a detailed description of the experimental setup employed to assess the performance of the proposed framework. 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.

5.1 Experimental Setup

Four terrain scenarios were constructed for the purpose of training the rat robot to navigate, as illustrated in Figure 6. The size of the terrain scenarios is designed to simulate the real-world environment encountered by small robots. Table 1 shows the hyper parameters in the experiment.

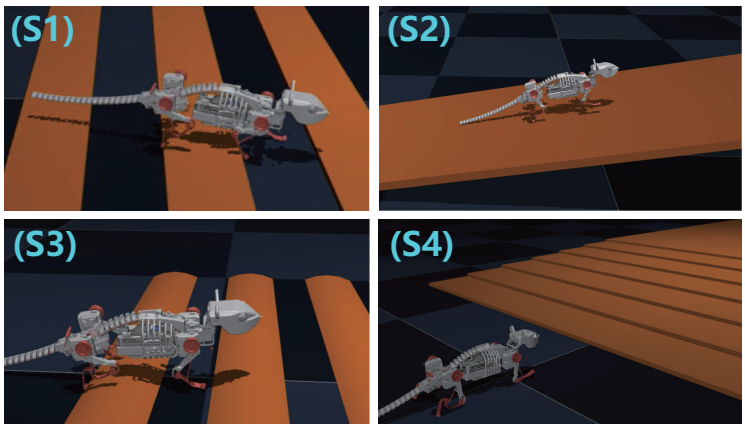


Fig. 6. The proposed approach can adapt the rat robot to all 4 test scenarios. (S1)Planks: Gallop over planks with wide gap(width=10 cm).(S2)Uphill: a 10-degree slope. (S3)Logs: Upon a pile of logs. (S4)Stairs: Micro stairs between two platforms.

Table 1. 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

The rat robot is a small quadrupedal robot that has dimensions of 40 cm × 25 cm and features 8 degrees of freedom for control. Each leg of the robot has a step length of 9 cm and a foot clearance of 1.5 cm. Using the physical platform utilized in our prior research, as depicted in Figure 1, a digital twin was created.

The simulations are carried out using the MuJoCo robot simulation platform[17](Version 2.1.0), operating on a Ubuntu 18.04 system. The training machine is a server equipped with dual Intel Xeon Gold 6234 Processors and 256GB of memory. Each training session is allocated one physical core and requires less than 2GB of memory.

5.2 Performance in Different Scenarios

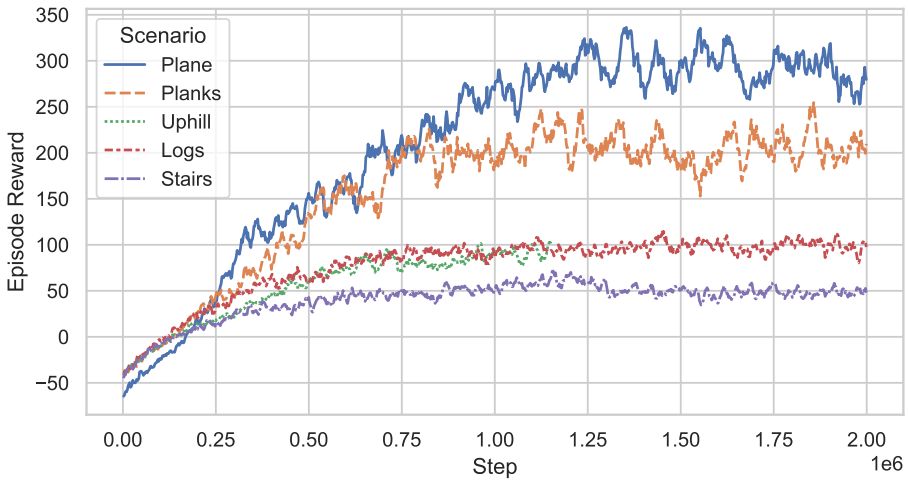


Fig. 7. Training curves on 5 simulation tasks.

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 agent converge successfully with different episode rewards. That results from terrain characteristics 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 show that the rat robot passes through all scenarios successfully.

Table 2 shows the performance metrics of four 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 four scenarios, average velocity in (S2) performs relatively small. Such phenomenon indicates that height variation is more challenging for the rat robot's movement, needing to be further studied. Besides, speed performance of (S4) is also worse than (S1) and (S3), but the success rate is higher.

Table 2. Performance in four scenarios. 50 validation experiments are completed for every scenario.

Scenario name	Sucess rate(%)	Episode reward	Average Velocity(cm/s)
(S1)Planks	72	210.3	3.24
(S2)Uphill	80	98.6	1.62
(S3)Logs	78	102.1	3.18
(S4)Stairs	86	99.4	2.26

5.3 Comparison With benchmarks

In order to objectively evaluate the effectiveness of the proposed method, we compare it with three representative benchmark algorithms. 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 uses 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[18].

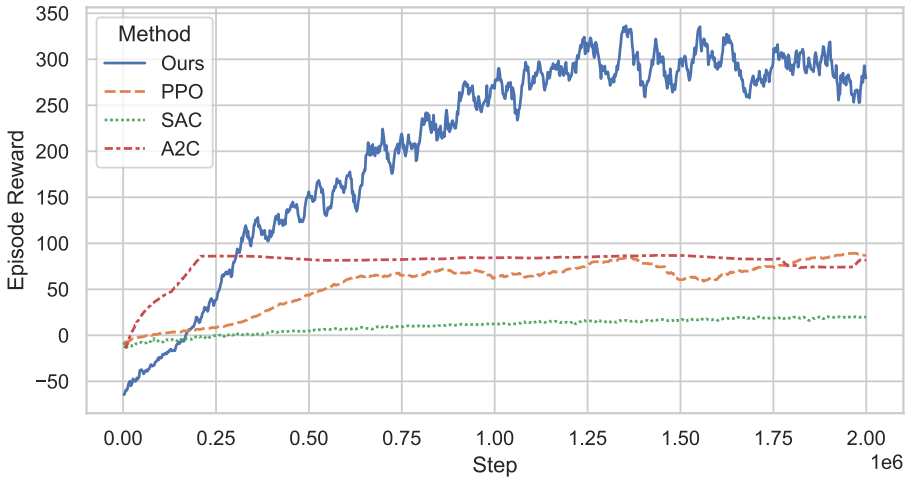


Fig. 8. Training curves of our method and baseline algorithms on the plane.

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, validation experiments show that their trained policies are not sufficient to drive the robot's forward motion on flat ground. 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.

6 CONCLUSIONS

This paper proposes a reinforcement learning method to generate motion of the rat robot for complex terrain. We design a reinforcement architecture to explore the nonlinear kinematics and dynamics of the small, flexible rat robot. Through post-processing of time-series data during the robot's interaction with the environment and time cluster updating that integrates self-action status and perceptual information from the environment, the initially large state space is compressed to enhance training speed. 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 tested and outperforms other benchmark algorithms.

References

1. Lucas, P., Oota, S., Conradt, J., Knoll, A.: Development of the neurorobotic mouse. In: 2019 IEEE International Conference on Cyborg and Bionic Systems (CBS). pp. 299–304. IEEE (2019)
2. Hutter, M., Gehring, C., Jud, D., Lauber, A., Bellicoso, C.D., Tsounis, V., Hwangbo, J., Bodie, K., Fankhauser, P., Bloesch, M., Diethelm, R., Bachmann, S., Melzer, A., Hoepflinger, M.: Anymal - a highly mobile and dynamic quadrupedal robot. In: 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 38–44 (2016)
3. Neuman, S.M., Plancher, B., Duisterhof, B.P., Krishnan, S., Banbury, C., Mazumder, M., Prakash, S., Jabbour, J., Faust, A., de Croon, G.C., Reddi, V.J.: 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). pp. 296–299 (2022)
4. Rus, D., Tolley, M.T.: Design, fabrication and control of soft robots. Nature 521(7553), 467–475 (May 2015), https://doi.org/10.1038/nature14543
5. Duisterhof, B.P., Krishnan, S., Cruz, J.J., Banbury, C.R., Fu, W., Faust, A., de Croon, G.C.H.E., Janapa Reddi, V.: Tiny robot learning (tinyrl) for source seeking on a nano quadcopter. In: 2021 IEEE International Conference on Robotics and Automation (ICRA). pp. 7242–7248 (2021)
6. Fankhauser, P., Bjelonic, M., Dario Bellicoso, C., Miki, T., Hutter, M.: Robust rough-terrain locomotion with a quadrupedal robot. In: 2018 IEEE International Conference on Robotics and Automation (ICRA). pp. 5761–5768 (2018)
7. Wang, X., Wang, S., Liang, X., Zhao, D., Huang, J., Xu, X., Dai, B., Miao, Q.: Deep reinforcement learning: A survey. IEEE Transactions on Neural Networks and Learning Systems pp. 1–15 (2022)
8. Kober, J., Bagnell, J.A., Peters, J.: Reinforcement learning in robotics: A survey. The International Journal of Robotics Research 32(11), 1238–1274 (2013)
9. Nguyen, H., La, H.: Review of deep reinforcement learning for robot manipulation. In: 2019 Third IEEE International Conference on Robotic Computing (IRC). pp. 590–595. IEEE (2019)
10. Xie, Z., Clary, P., Dao, J., Morais, P., Hurst, J., van de Panne, M.: Iterative reinforcement learning based design of dynamic locomotion skills for cassie. arXiv preprint arXiv:1903.09537 (2019)
11. Hwangbo, J., Lee, J., Dosovitskiy, A., Bellicoso, D., Tsounis, V., Koltun, V., Hutter, M.: Learning agile and dynamic motor skills for legged robots. Science Robotics 4(26), eaau5872 (2019)
12. Shi, H., Zhou, B., Zeng, H., Wang, F., Dong, Y., Li, J., Wang, K., Tian, H., Meng, M.Q.H.: Reinforcement Learning With Evolutionary Trajectory Generator: A General Approach for Quadrupedal Locomotion. IEEE Robotics and Automation Letters 7(2), 3085–3092 (Apr 2022), conference Name: IEEE Robotics and Automation Letters
13. Kamthe, S., Deisenroth, M.: Data-efficient reinforcement learning with probabilistic model predictive control. In: International conference on artificial intelligence and statistics. pp. 1701–1710. PMLR (2018)
14. Hyun, D.J., Seok, S., Lee, J., Kim, S.: High speed trot-running: Implementation of a hierarchical controller using proprioceptive impedance control on the mit cheetah. The International Journal of Robotics Research 33(11), 1417–1445 (2014)

15. Raibert, M., Blankespoor, K., Nelson, G., Playter, R.: Bigdog, the rough-terrain quadruped robot. IFAC Proceedings Volumes 41(2), 10822–10825 (2008)
16. Rohregger, A.: Mouse gait: Visual analysis of front leg. Website (2021), https://www.notion.so/Rodent-Gait-89ae86764cc243cfa1d58a04dace5a15
17. Todorov, E., Erez, T., Tassa, Y.: Mujoco: A physics engine for model-based control. In: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 5026–5033 (2012)
18. Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., Dormann, N.: Stable-baselines3: Reliable reinforcement learning implementations. J. Mach. Learn. Res. 22(1) (jan 2021)