A Trajectory-based Reinforcement Learning Approach for Autonomous Locomotion of a Rat Robot

Zijian Zhao[1], Zitao Zhang[1], Kai Huang [1,*]

***Abstract*— Developing effective strategies in complex environments has long been a challenge for small robots. Due to limited space, small robots often have scarce sensing and computation resources, making it difficult for them to comprehend the intricate nature of nonlinear dynamics and learn motion strategy from scratch. In this paper, we propose a novel lightweight trajectory-based reinforcement learning approach for enabling robots to learn adaptive motion in complex terrains. Our method leverages the Augmented Random Search (ARS) algorithm to optimize control parameters for Bezier curves, which in turn generate motion trajectories for the robot. The Bezier curves provide prior knowledge to decrease the learning difficulty, while ARS provides an efficient optimization strategy with small computation cost. Additionally, we design a simple environment randomization method that trains the robot in diverse environments, enabling it to learn a general strategy for walking and navigating obstacles instead of being limited to a specific scenario. We evaluate our method in various scenarios with different obstacles, and the experimental results demonstrate its superior performance with minimal computational cost. To support future research, we will release the code for our model upon publication.**

I. Introduction

The rat robot is a small quadrupedal robot designed based on the real rat, as depicted in Fig. 1 [1], [2]. It possesses several advantages, including easy assembly, portability, flexibility, low cost, and low energy consumption. In comparison to wheeled robots and bipedal robots, quadrupedal robots demonstrate greater competitiveness in navigating complex terrains while maintaining high stability. Additionally, due to the inherent flexibility of small robots, the small rat robot finds wide application potential in various fields, such as rescue operations and surveys [3], [4], [5].

Conventional control methods for quadruped locomotion often require extensive manual modeling, computation, and solving complex optimization problems, which are time-consuming and require significant manual tuning. In contrast, Reinforcement Learning (RL) enables robots to learn proper motion strategies by interacting adaptively with environment. Recently, there has been significant research focusing on RL approaches for autonomous locomotion in rat robots [6], [7], [8], [9], [10].

However, there are also challenges in small robots like the rat robot [11]. Firstly, some RL control methods depend on specific sensor resources, and some deep RL control methods require substantial computational resources like GPUs. However, due to space limitations, small robots often have extremely limited sensing and computational resources [5], [12]. Lighter-weight RL methods such as Basic Random Search (BRS) and Augmented Random Search (ARS) [13] can help address this problem. However, most of these methods are difficult to optimize for complex networks. Many methods that use ARS only employ one-layer networks [14], [15], which may limit the policy network's capacity and affect the learned strategy's performance.

To overcome this problem, we propose a novel Bezier trajectory generator to enhance the control ability of the lightweight RL framework. What's more, many works illustrate that training robots to learn efficient quadrupedal locomotion skills from scratch is challenging to a certain extent [15], [16]. It is a promising approach to provide prior knowledge like using open-loop trajectory generators [10], [17], [18] such as central pattern generators (CPG) [19] and Bezier trajectory generator [20]. Additionally, most of these methods can also help avoid overshooting, which often occurs in traditional end-to-end methods or methods based on end-effector control.

Another problem is the limited transferability of learned strategies in small robots [11]. Due to the small observation space and relative exploration ability compared to larger robots, the policy network may easily converge to a local optimal solution. This can make the learned strategy difficult to apply in a novel scenario. To address this issue, we design a simple yet effective environment randomization technique. By training the robot in randomized scenarios, it can learn a more general motion strategy and adapt to various obstacles.

In this paper, we present a novel method for autonomous locomotion of a rat robot. Our approach aims to address the high requirements of sensors and computation while enabling the robot to learn general strategies in different scenarios. To achieve this, we introduce an adaptive Bezier trajectory generator in our method, which generates appropriate leg trajectories for autonomous robot motion. Furthermore, we incorporate the ARS algorithm into our framework to facilitate lightweight optimization. Additionally, we propose a simple yet effective environment randomization method to enhance the robot's ability to learn more general strategies. We evaluate the performance of our method in four different scenarios, and the results demonstrate its advanced performance and stability. In summary, the main contributions of our work are as follows:

- We propose a lightweight trajectory-based reinforcement learning framework for autonomous locomotion of a rat robot. It combines the benefits of the low-cost of ARS optimization method and the powerful control

*Corresponding Author: Kai Huang

[1] Authors from the the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510275, China

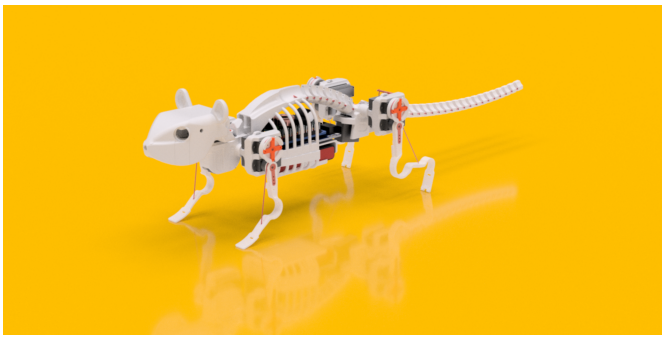


Fig. 1. The rat robot is designed with soft actuated components and supports more flexible robot motions.

ability of Bezier-based locomotion method.

- We design a novel Bezier trajectory generator according to the controlling principle of Bezier curves. It can help decrease the training difficulty and increase the motion ability of the robot.
- We develop a simple environment randomization method that facilitates the learning of a general approach for walking and overcoming obstacles by robots.

II. Related Work

Recently, RL has become the most promising method for robot motion control. It can be divided into two types: model-free methods and model-based methods. Model-free methods directly learn the policy from the observed interactions between the agent and the environment, without explicitly learning the environment's dynamics, which makes them easy to realize. However, due to the complex nonlinear dynamics and sparse rewards, training a robot from scratch is challenging [16]. On the other hand, model-based methods involve building a model of the environment or utilizing a predetermined dynamic system, which can be used to decrease the training difficulty and enhance the motion ability. This is similar to the idea of combining black box and white box, where the white box provides prior knowledge to enhance the black box, which has been proven efficient in many fields [21], [22]. Shi et al. [16] designed a trajectory generator and optimized the trajectory to guide the robot motion. Crespi et al. [23] used CPG to mimic the locomotion skills of animals for a robot fish control.

Currently, most RL methods are designed for full-sized robots like Cheetah [24], Bigdog [25] and ANYmal [26], and only limited research focuses on small-sized robots like the robot rat [1]. Limited by the sensor and computation resources, it is challenging to directly transfer RL methods designed for large robots to small robots. It is promising to use prior knowledge to reduce the exploration space and difficulty, or use lightweight optimization methods to reduce the computational burden. Rahme et al. [15] proposed a similar method that combines ARS and Bezier curves. However, their Bezier trajectory generator is entirely based on the Bezier curve designed by Hyun et al. [20], where some hyperparameters are tailored for the Cheetah robot they used, which may not be suitable for other robots. As a result, this limits the generalizability of the method proposed by Rahme et al. [15].

III. Preliminary

A. Bezier Locomotion Control Method

Bezier curve is a smooth curve determined by a series of control points $P = [P_1, P_2, ..., P_n]$. Its mathematical expression is given by Eq. 1.

$$\begin{aligned} B(t) &= \sum_{i=0}^{n} P_i B_{i,n}(t),\ 0 \le t \le 1 \\ B_{i,n}(t) &= C_n^i (1-t)^{n-i} t^i \end{aligned} \tag{1}$$

where $B_{i,n}$ represents the i-th coefficient of the Bezier curve, and C_n^i denotes the binomial coefficient. By properly designing the control points, we can create smooth curves of any desired shape. Additionally, we can control the derivatives of any order at specific points. These features make Bezier curves widely used in various fields, including robot control, where the first and second derivatives correspond to velocity and acceleration, respectively.

One of the most well-known and widely used locomotion control methods based on Bezier curves was proposed by Hyun et al. [20], as shown in Figure 2. In this method, the leg position is determined by a predefined trajectory and a phase signal that indicates the current position within the trajectory. The phase signal is defined by Eq. 2.

$$\begin{aligned} S(t) = (&\Delta S \\ &+ \begin{cases} \frac{t+\hat{T}_{sw}}{\hat{T}_{sw}} + 1,\ -\hat{T}_{sw} < t < 0 \text{ (lift-off phase)} \\ \frac{t}{T_{st}},\ 0 < t < T_{st} \text{ (stand phase)} \\ \frac{t-\hat{T}_{st}}{\hat{T}_{sw}} + 1,\ \hat{T}_{st} < t < \hat{T}_{st} + \hat{T}_{sw} \text{ (touch-down phase)} \end{cases} \\ &) \bmod 2 \end{aligned} \tag{2}$$

where t represents time, T_{st} and T_{sw} represent the stand period and swing period, respectively. The stand period corresponds to the leg being on the ground, while the swing period corresponds to the leg being in the air, including lift-off phase and touch-down phase. ΔS represents the phase difference between the four legs, which can control the movement pattern, such as trotting or galloping. Then the leg positions can be obtained by Eq. 3.

$$[X, Y] = \begin{cases} [\tau(1 - 2S(t)), \delta cos\frac{\pi(1-2S(t))}{2}],\ 0 \le S(t) < 1 \\ B(S(t) - 1),\ 1 \le S(t) < 2 \end{cases} \tag{3}$$

As shown in Figure 2, Hyun et al. designed the control points based on the following principles:

- The ordinate of the first control point, P_0, and the last control point, P_{11}, is set to 0, ensuring that the start and end positions of the motion trajectory are exactly on the ground.
- The ordinate of the first two control points, P_0 and P_1, is the same, and the ordinate of the last two control points,

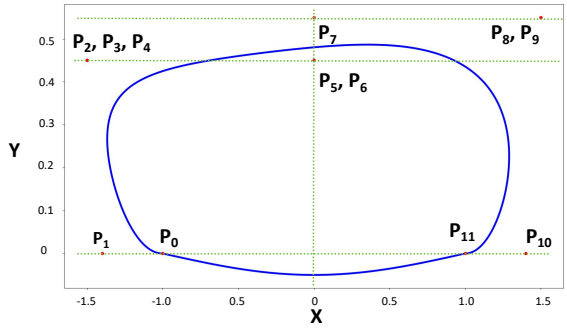


Fig. 2. Bezier curve of Hyun et al. [20].

P_{10} and P_{11}, is the same, ensuring that the vertical velocity at the start and end positions is 0.

- The three control points $P_2 \sim P_4$ are coincident, and the two control points $P_8 \sim P_9$ are coincident, used to change the direction of force and adjust the direction of leg movement trajectory.
- The abscissa of $P_5 \sim P_7$ is the same, and the ordinate of $P_7 \sim P_9$ is the same, providing longitudinal and transverse traction effects on the curve, respectively.

In our method, we follow their design principles but incorporate dynamic control points based on the environment.

B. Augmented Random Search

Augmented Random Search (ARS) is a popular lightweight reinforcement learning method that builds upon the Basic Random Search (BRS) algorithm [13]. Unlike gradient descent, BRS optimizes parameters through random sampling, which allows the parameters to move in directions that increase the value of the reward function. ARS is an enhanced version of BRS that incorporates additional techniques, such as standardizing the update step and normalizing the observation space, to mitigate covariate shift [27] and improve the stability of the training process. In our approach, we utilize ARS-v2top, where the parameter update follows Eq. 4:

$$\theta := \theta + \frac{\alpha}{N\sigma}\sum_{i}(r(\pi_{\theta+\delta_i}) - r(\pi_{\theta-\delta_i}))\delta_i \tag{4}$$

where θ represents the parameters being optimized, α is the learning rate, N denotes the top N sampling point pairs in the epoch, $r()$ represents the long-term reward, π refers to the policy network, δ_i represents the randomly sampled noise, and σ represents the standard deviation of all $(r(\pi_{\theta+\delta_i}) - r(\pi_{\theta-\delta_i}))$. For more detailed information, interested readers can refer to the ARS paper [13].

IV. Methodology

A. Overview

As illustrated in Figure 3, our method encompasses three key highlights:

(1) Environment Randomization: In each epoch, we generate a random terrain to train the robot. This approach enables the robot to acquire a generalized motion strategy, rather than focusing solely on specific scenarios.

(2) Bezier Trajectory Generator: Our method utilizes the parameters of a Bezier curve as the action space. The Bezier Trajectory Generator generates the trajectory for each leg accordingly. This simplifies the learning process, as the robot does not need to start from scratch as in end-to-end methods. As a result, we can employ a lightweight policy network with a single layer to leverage our method.

(3) Optimization Based on ARS: Given the limited computational resources of small robots, we utilize the optimization capabilities of ARS instead of gradient descent for policy network optimization. ARS has demonstrated its effectiveness in reinforcement learning while keeping computation costs low. By employing random search, we can train the policy network without requiring a GPU.

B. Environment Randomization

There has been some research focused on training robots in random terrains to improve sim-to-real transfer performance [15] or enhance their ability to navigate complex environments [28]. Previous methods have utilized simulated environments with numerous hills of varying heights as random terrains. However, implementing such terrains in certain simulation tools can be challenging. Additionally, constructing such random terrains during real-world training poses difficulties.

Therefore, we propose a simple yet effective method of environment randomization. We generate random terrains by using multiple non-overlapping rectangular prisms with random sizes. In Section V-C, we demonstrate the efficiency of our approach, showing that the learned strategies can be successfully applied in different scenarios. Specifically, the rectangular prisms have fixed lengths and widths, while the height follows a uniform distribution $U(0, h_{max})$. The construction of random terrains based on rectangles is also straightforward in the real world, making it easier to train robots in real-world environments.

C. Bezier Trajectory Generator

In our method, we employ a Bezier trajectory generator to generate trajectories for the robot. The policy network generates the parameters of the Bezier trajectory. Given the trajectory and phase signal, we can control the robot's motion, as described in Section III-A. We design a 12-dimensional action space to control the Bezier trajectory, following the principle proposed by Hyun et al. [20]. When the phase signal $S(t) \leq 1$, the position is determined by the control points shown in Table I, where the 12 control points are determined by the first 10 action spaces. When $S(t) < 1$, the position is determined by Eq. 5, which is related to the last 2 action spaces.

$$[X, Y] = [a[10](1 - 2S(t)), a[11]cos\frac{\pi(1 - 2S(t))}{2}] \tag{5}$$

The policy network helps generate a Bezier trajectory with a proper shape based on the observation. Compared to the

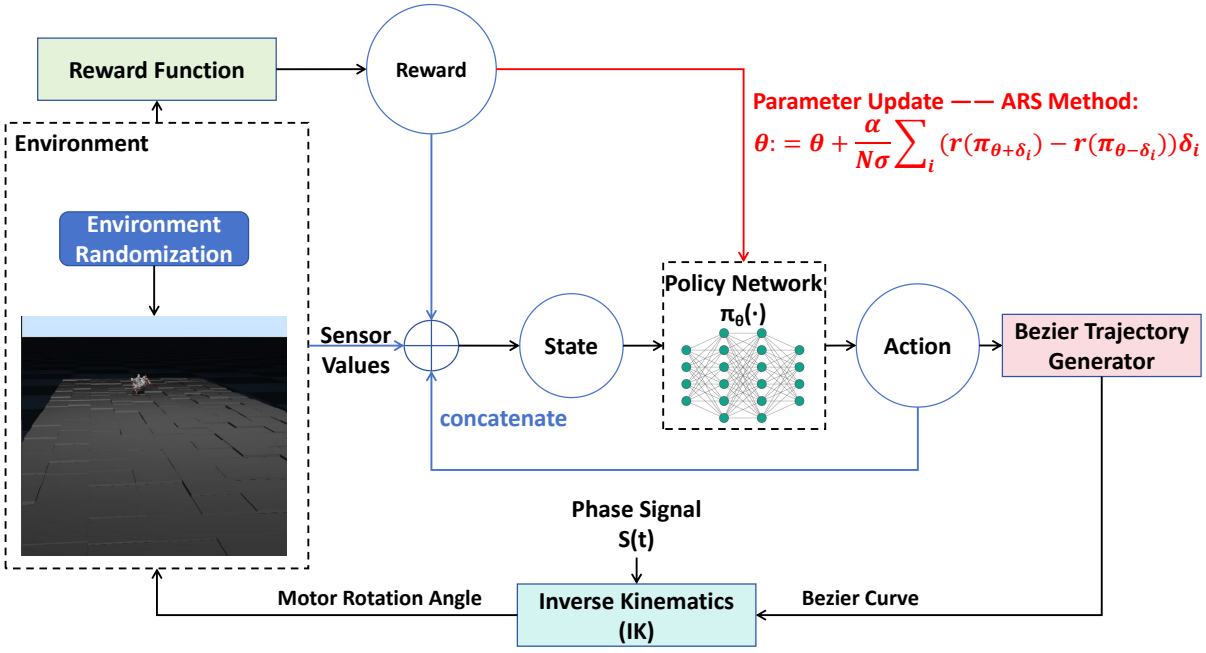


Fig. 3. Architecture of our trajectory-based reinforcement learning method.

TABLE I
CONTROL POINTS IN SWING PHASE

Control Point	Coordinate	Control Point	Coordinate
P_0	$(a[0], 0)$	P_7	$(a[3], a[9])$
P_1	$(a[1], 0)$	P_8, P_9	$(a[4], a[9])$
P_2, P_3, P_4	$(a[2], a[7])$	P_{10}	$(a[5], 0)$
P_5, P_6	$(a[3], a[8])$	P_{11}	$(a[6], 0)$

TABLE II
VALUE RANGE OF ACTION SPACE

Action	Range	Action	Range	Action	Range
$a[0]$	$-k \sim 0$	$a[1]$	$-1.4k \sim 0$	$a[2]$	$-1.5k \sim 0$
$a[3]$	$-0.5k \sim 0.5k$	$a[4]$	$0 \sim 1.5k$	$a[5]$	$0 \sim 1.4k$
$a[6]$	$0 \sim k$	$a[7]$	$0 \sim 0.3k$	$a[8]$	$0 \sim 0.3k$
$a[9]$	$0 \sim 0.4k$	$a[10]$	$0 \sim k$	$a[11]$	$-0.2k \sim 0$

traditional control method with a fixed Bezier trajectory, our method offers more flexibility. However, a good trajectory should have a similar shape to Figure 2. Therefore, to reduce training difficulty, we design the range of each dimension of the action space as shown in Table II, where k is a positive threshold.

D. RL Loop Based on ARS

In this section, we will provide a detailed description of our RL framework. As mentioned earlier, our action space consists of a 12-dimensional vector, denoted as a_t. We design the observation state s_t using sensor information, as well as the action and reward from the previous step, as shown in Eq. 6.

$$s_t = (a_{t-1} \in R^{12}, r_{t-1} \in R, \vec{V}_{vel} \in R^3, \vec{V}_{gyro} \in R^3, \vec{V}_Q \in R^4, \mathbf{c}_F \in R^4) \in R^{27} \tag{6}$$

where a_{t-1} and r_{t-1} represent the action and reward at the previous time step, $\vec{V}_{vel}$ and $\vec{V}_{gyro}$ represent the linear speed and angular velocity, $\vec{V}_Q$ represents the quaternion that represents the orientation, and $\mathbf{c}_F$ represents the values of the pressure sensors on the four legs. Next, we define the reward function as shown in Eq. 7..

$$r = \begin{cases} W_{vel}\vec{V}_{vel} \cdot \vec{u}_{dir}, & \vec{V}_{vel} \cdot \vec{u}_{dir} < T_{vel} \\ -r_p, & \vec{V}_{vel} \cdot \vec{u}_{dir} \geq T_{vel} \end{cases} \tag{7}$$

where $\vec{u}_{dir}$ is a unit vector indicating the direction, W_{vel} is a weighting factor, T_{vel} is the velocity threshold, and $r_p > 0$ is a penalty. The reward function is based on the speed in the forward direction. Additionally, if the speed exceeds a certain threshold, we apply a penalty to the robot to prevent it from moving too fast and potentially falling. Finally, we define the policy network as shown in Eq. 8.

$$\pi_\theta(s_t) = \text{Tanh}(\theta^T \text{Norm}(s_t)) \tag{8}$$

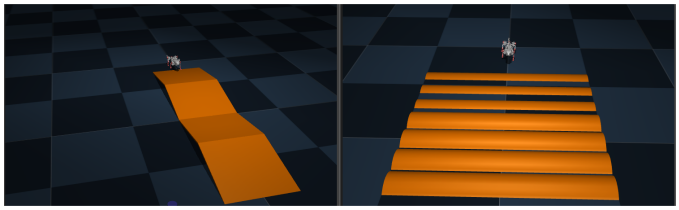
where $\theta \in R^{27\times12}$ represents the network parameters. To keep the framework lightweight, we use a single-layer network with only 324 parameters. The output of π_θ is constrained to the range of -1 to 1. The actual action is then obtained by clipping the output according to Table II. Following the establishment of the fundamental components of RL, we proceed to train the model using the ARS algorithm. A comprehensive overview of the algorithm can be found in Algorithm 1.

Algorithm 1 RL Algorithm with the Rat Robot

Require:
s_0: state with initial environment
M: the number of steps in an episode
N: the number of episodes in an epoch
σ: standard deviation of Gaussian noise for exploration
α: learning rate

Initialize policy network π_θ
while Not Converged **do**
  for $i = 1$ to N **do**
    Create an empty buffer β
    Sample δ_i from the normal distribution $N(0, \sigma)$
    for ξ in $[\delta_i, -\delta_i]$ **do**
      Reset the environment, set state $s_t := s_0$, and set the long-term reward $r_\xi := 0$
      for $t = 1$ to M **do**
        Get the action signal $a_t := \text{Clip}(\pi_{\theta+\xi}(s_t))$
        Get the phase signal $S(t)$
        Generate the Bezier trajectory $B_t \leftarrow a_t$
        Get the end-effector space tracking point $(x, y) \leftarrow (B_t, S(t))$
        Transfer to the motor signal $(q_1, q_2) := \text{IK}(x, y)$
        Conduct the servo control signal (q_1, q_2)
        Get the observation s_{t+1} and the reward r_t
        Update the long-term reward $r_\xi := r_\xi + r_t$
      end for
      Append the result (ξ, r_ξ) to the buffer β
    end for
  end for
  Compute the update step $\nabla_\theta := \text{ARS}(\beta)$
  Update the parameters $\theta := \theta + \alpha\nabla_\theta$
end while

(a) Scene1: Flat (b) Scene2: Planks



(c) Scene3: Hills (d) Scene4: Logs

Fig. 4. Four test scenarios for the rat robot.

TABLE III
MODEL CONFIGURATION

Parameter	Value
Learning rate α	0.03
Standard deviation of exploration noise σ	0.05
Number of episodes in an epoch N	16
Number of steps in an episode M	5000
Weight of reward function W_{vel}	4.0
Punishment in reward function r_p	0.1
Velocity threshold in reward function T_{vel}	0.35
Phase difference in phase signal ΔS	[0,1,1,0]

V. EXPERIMENT

A. Experiment Setup

As depicted in Figure 4, we have constructed four scenarios in the MuJoCo [29] simulation platform to test our method. These scenarios include a flat surface and three obstacle scenarios. The model configuration of our method is presented in Table III.

During the experiment, we have created a digital twin using the physical platform employed in our previous research, as illustrated in Figure 1. The rat robot is a small quadrupedal robot with dimensions of approximately 40 cm × 25 cm and 8 degrees of freedom in its legs for control. The simulation experiments were conducted on a server running on Ubuntu 20.04 system, equipped with dual Intel Xeon Gold 6234 Processors. It is worth noting that our method does not require any GPU for training.

B. Performance Analysis in Fixed Environment

In this section, we train models in the four given scenarios depicted in Figure 4. We compare our method with PPO [30], ARS [13], and the method proposed by Rahme et al. [15]. Additionally, for PPO and ARS, we consider two variations: the end-to-end method (referred to as PPO-end and ARS-end) and the method that uses the position of the end point of legs as the action space, which is then mapped to motor signals through inverse kinematics (referred to as PPO-pos and ARS-pos).

The training curves are depicted in Fig. 5, showcasing the maximum advance distance achieved in each epoch. The results clearly demonstrate the superiority of our method over other approaches in all scenarios, exhibiting both a high converged reward and convergence speed. Additionally, the ARS method exhibits a low training cost and rapid convergence, while the incorporation of Bezier trajectories further enhances motion capabilities, resulting in significantly larger advance distances. In comparison, the PPO method requires approximately 2GB of GPU memory during training and necessitates a greater number of training epochs to converge. Furthermore, we take the advance distance of the pure Bezier trajectory in the flat scenario as a benchmark. It can be observed that our method outperforms this benchmark in all scenarios, which suggests that our method efficiently optimizes the shape of the Bezier trajectory.

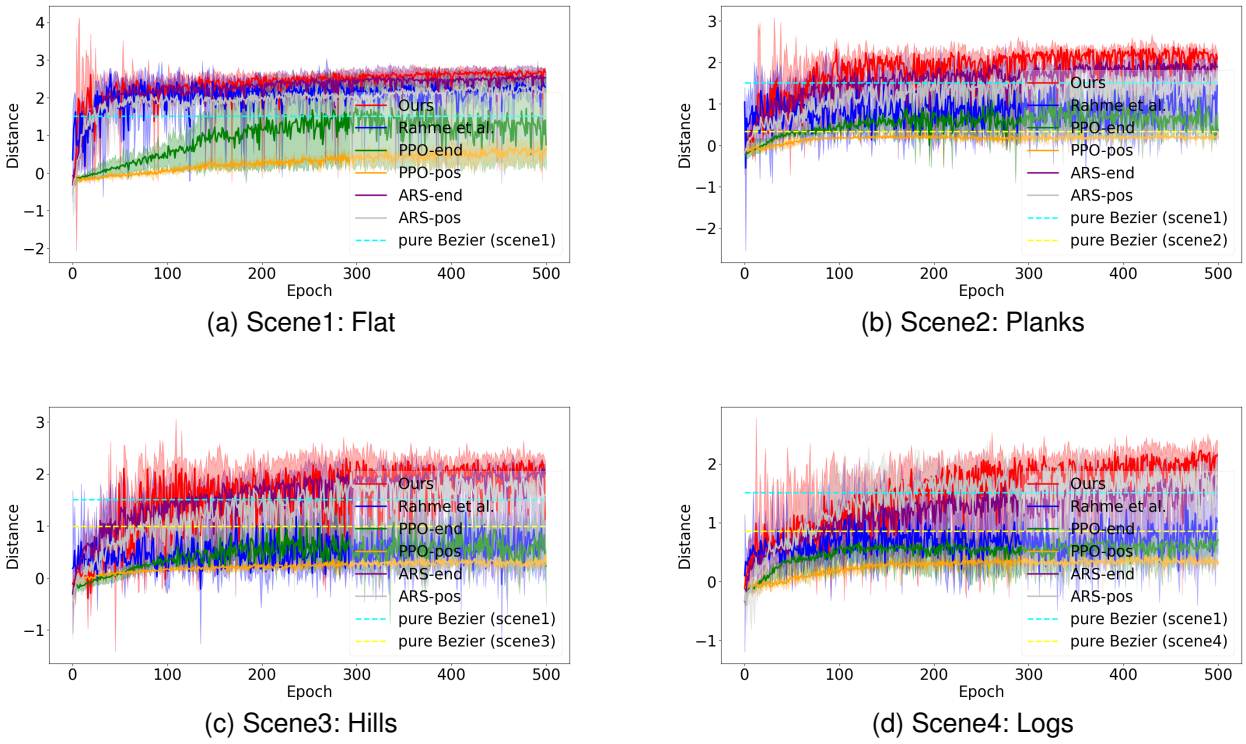


Fig. 5. Training curves on 4 simulation tasks. Each experiment is repeated four times. The solid lines depict the average advancement distance, while the shaded areas indicate the standard deviation. The dashed lines represent the advancement distance achieved using the pure Bezier control method with identical steps.

TABLE IV

PERFORMANCE OF OUR METHOD IN EACH SCENARIO

Training Scenario \ Evaluating Scenario	Scene1: Flat	Scene2: Planks	Scene3: Hills	Scene4: Logs
Scene1: Flat	2.637	0.435	deviation	fall down
Scene2: Planks	1.614	2.113	deviation	fall down
Scene3: Hills	0.882	deviation	2.026	fall down
Scene4: Logs	2.309	deviation	fall down	1.932

TABLE V

PERFORMANCE COMPARISON OF THE ABILITY TO OVERCOME OBSTACLES: THE RESULT REPRESENTS THE AVERAGE VALUE OBTAINED FROM FOUR REPEATED EXPERIMENTS.

Scenario	Maximum Distance of Advance		Maximum Height Clearance over Obstacles	
	Ours	**Rahme et al. [15]**	**Ours**	**Rahme et al. [15]**
Random Environment	**1.344**	1.116	–	–
Scene1: Flat	1.538	1.31	–	–
Scene2: Planks	**0.7291**	0.5545	**0.196**	0.157
Scene3: Hills	**0.8852**	fall down	**0.407**	fall down
Scene4: Logs	0.7320	**0.8438**	**0.120**	0.119

C. Performance Analysis in Random Environment

Table IV presents the performance of our approach in each scenario. It is evident that the strategy learned in one scenario is challenging to transfer to other scenarios. Especially when there is a significant difference in the obstacle type, the robot is more prone to falling down or deviating from the intended forward direction. Therefore, our environment randomization plays a crucial role in enabling the robot to acquire a generalized strategy.

We compare our method with the method proposed by Rahme et al. [15]. They proposed a similar method to ours, combining the ARS algorithm and Bezier curves. However, their action space consists of 10 dimensions, including 2 dimensions for generating the Bezier trajectory and 8 dimensions for fine-tuning the action in each leg. As mentioned earlier, their Bezier trajectory is entirely based on [20] which was designed for the Cheetah robot [24], and may not be directly suitable for other robots. We conducted a comparison

Fig. 6. Montage in scene 2: planks. The first row shows the movement based on a pure Bezier trajectory [20]. The second row shows the movement based on our method. The third row shows the movement based on the method of Rahme et al. [15].

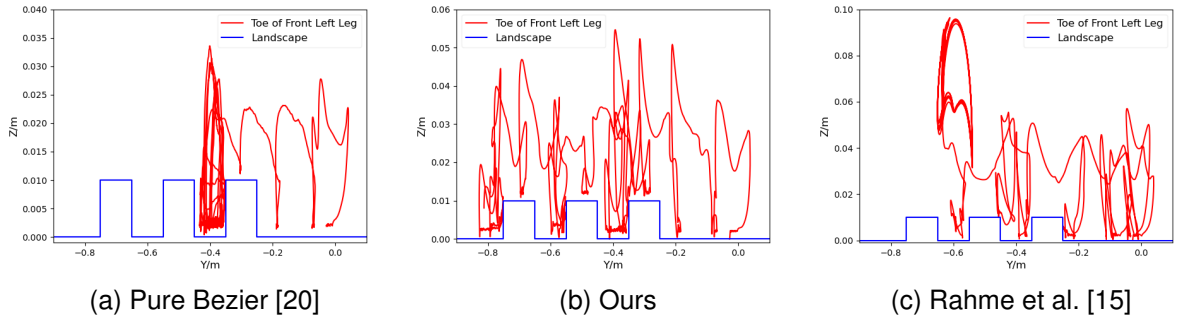


Fig. 7. End point trajectory of the rat robot's front left leg in scene2: planks.

between our method and theirs, as presented in Table V. Both methods are trained in a random environment and evaluated for their performance in different scenarios. The comparison focuses on the maximum distance of advance and the maximum height clearance over obstacles within 5000 steps. The results indicate that our method achieves success and outperformes Rahme et al.'s method in most scenarios. Conversely, Rahme et al.'s method failes to overcome the hill in Scene 3, leading to the robot falling down from the obstacle. It is important to note that both our method and Rahme et al.'s method use a similar Bezier trajectory with 12 control points. The results suggest that our Bezier trajectory generator is more powerful.

In Fig. 6 and Fig. 7, we illustrate the montage of the robot motion and the end point trajectory in the planks scenario, respectively. We compare our method with the method of Rahme et al. and the pure Bezier trajectory. It can be seen that our method successfully overcomes all three planks. The pure Bezier control method can only make the front legs stride over the first plank, but the rear legs are blocked. The Rahme et al. method can only help the robot stride the first plank, but it causes the robot to fall down on the second plank.

According to previous research [31], the Bezier-based control method may result in a relatively high lateral displacement, causing the actual path to deviate from the planned path. In order to evaluate the stability of different methods, we conducted an experiment as shown in Table VI. The experiment involved calculating the maximum lateral offset and the standard deviation of triaxial acceleration on a flat surface. We tested the strategies learned in both a flat surface environment and a random environment using our method and the method proposed by Rahme et al. , respectively. The results demonstrate that our method exhibits better stability compared to Rahme et al.'s method.

VI. Conclusion

In this paper, we present a novel lightweight trajectory-based reinforcement learning method for autonomously generating motion for a rat robot in complex terrains. Our framework utilizes the ARS algorithm to optimize the policy network, resulting in a low training cost that is more suitable for small robots. We also design a novel Bezier trajectory generator to enhance the performance of the strategy and decrease the training difficulty. Additionally, we propose a simple yet efficient environment randomization method to enable the robot to learn a general strategy for walking and crossing obstacles. Through a series of case studies, the results demonstrate that our method achieves superior performance and fast convergence speed in complex terrains.

TABLE VI
STABILITY TEST ON FLAT SURFACE

Indicator	Trained in Flat Surface		Trained in Random Environment	
	Ours	Rahme et al. [15]	Ours	Rahme et al. [15]
Maximum Lateral Offset	**0.181**	0.252	0.549	0.638
Standard Deviation of X-axis Acceleration	6.381	8.959	**5.481**	8.088
Standard Deviation of Y-axis Acceleration	6.181	8.243	**5.802**	7.073
Standard Deviation of Z-axis Acceleration	7.782	11.715	**6.778**	11.065

Furthermore, our method exhibits greater stability compared to previous approaches.

REFERENCES

[1] 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)*, pp. 299–304, IEEE, 2019.

[2] 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.

[3] S. D. De Rivaz, B. Goldberg, N. Doshi, K. Jayaram, J. Zhou, and R. J. Wood, "Inverted and vertical climbing of a quadrupedal microrobot using electroadhesion," *Science Robotics*, vol. 3, no. 25, p. eaau3038, 2018.

[4] B. P. Duisterhof, S. Li, J. Burgués, V. J. Reddi, and G. C. de Croon, "Sniffy bug: A fully autonomous swarm of gas-seeking nano quadcopters in cluttered environments," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 9099–9106, IEEE, 2021.

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

[6] 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)*, pp. 2622–2627, IEEE, 2022.

[7] 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)*, pp. 331–338, IEEE, 2023.

[8] Y. Huang, Z. Bing, Z. Zhang, G. Zhuang, K. Huang, and A. Knoll, "Optimizing dynamic balance in a rat robot via the lateral flexion of a soft actuated spine," *arXiv preprint arXiv:2403.00944*, 2024.

[9] X. Shan, Y. Huang, Z. Bing, Z. Zhang, X. Yao, K. Huang, and A. Knoll, "Locomotion generation for a rat robot based on environmental changes via reinforcement learning," *arXiv preprint arXiv:2403.11788*, 2024.

[10] 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)*, pp. 1–6, IEEE, 2023.

[11] S. M. Neuman, B. Plancher, B. P. Duisterhof, S. Krishnan, C. Banbury, M. Mazumder, S. Prakash, J. Jabbour, A. Faust, G. C. de Croon, *et al.*, "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, IEEE, 2022.

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

[13] H. Mania, A. Guy, and B. Recht, "Simple random search provides a competitive approach to reinforcement learning," *arXiv preprint arXiv:1803.07055*, 2018.

[14] J. Dibachi and J. Azoulay, "Teaching a robot to walk using reinforcement learning," *arXiv preprint arXiv:2112.07031*, 2021.

[15] M. Rahme, I. Abraham, M. L. Elwin, and T. D. Murphey, "Linear policies are sufficient to enable low-cost quadrupedal robots to traverse rough terrain," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8469–8476, IEEE, 2021.

[16] 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, 2022.

[17] G. Endo, J. Morimoto, T. Matsubara, J. Nakanishi, and G. Cheng, "Learning cpg-based biped locomotion with a policy gradient method: Application to a humanoid robot," *The International Journal of Robotics Research*, vol. 27, no. 2, pp. 213–228, 2008.

[18] A. Kamimura, H. Kurokawa, E. Yoshida, S. Murata, K. Tomita, and S. Kokaji, "Automatic locomotion design and experiments for a modular robotic system," *IEEE/ASME Transactions on mechatronics*, vol. 10, no. 3, pp. 314–325, 2005.

[19] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: a review," *Neural networks*, vol. 21, no. 4, pp. 642–653, 2008.

[20] 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.

[21] Z.-Q. Luo, X. Zheng, D. López-Pérez, Q. Yan, X. Chen, N. Wang, Q. Shi, T.-H. Chang, and A. Garcia-Rodriguez, "Srcon: A data-driven network performance simulator for real-world wireless networks," *IEEE Communications Magazine*, vol. 61, no. 6, pp. 96–102, 2023.

[22] D. López-Pérez, A. De Domenico, N. Piovesan, and M. Debbah, "Data-driven energy efficiency modelling in large-scale networks: An expert knowledge and ml-based approach," *IEEE Transactions on Machine Learning in Communications and Networking*, 2024.

[23] A. Crespi, D. Lachat, A. Pasquier, and A. J. Ijspeert, "Controlling swimming and crawling in a fish robot using a central pattern generator," *Autonomous Robots*, vol. 25, pp. 3–13, 2008.

[24] S. Seok, A. Wang, M. Y. Chuah, D. J. Hyun, J. Lee, D. M. Otten, J. H. Lang, and S. Kim, "Design principles for energy-efficient legged locomotion and implementation on the mit cheetah robot," *Ieee/asme transactions on mechatronics*, vol. 20, no. 3, pp. 1117–1129, 2014.

[25] M. Raibert, K. Blankespoor, G. Nelson, and R. Playter, "Bigdog, the rough-terrain quadruped robot," *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 10822–10825, 2008.

[26] M. Hutter, C. Gehring, D. Jud, A. Lauber, C. D. Bellicoso, V. Tsounis, J. Hwangbo, K. Bodie, P. Fankhauser, M. Bloesch, *et al.*, "Anymal-a highly mobile and dynamic quadrupedal robot," in *2016 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pp. 38–44, IEEE, 2016.

[27] N. G. Nair, P. Satpathy, J. Christopher, *et al.*, "Covariate shift: A review and analysis on classifiers," in *2019 Global Conference for Advancement in Technology (GCAT)*, pp. 1–6, IEEE, 2019.

[28] A. Ettlin and H. Bleuler, "Randomised rough-terrain robot motion planning," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5798–5803, IEEE, 2006.

[29] 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*, pp. 5026–5033, IEEE, 2012.

[30] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[31] Y. Wang, Z. Liu, A. Kandhari, and K. A. Daltorio, "Obstacle avoidance path planning for worm-like robot using bezier curve," *Biomimetics*, vol. 6, no. 4, p. 57, 2021.