PAMUKKALE ÜNİVERSİTESİ
Mühendislik Bilimleri Dergisi

Araştırma makalesi | Research paper

PAMUKKALE UNIVERSITY
Journal of Engineering Sciences

Cilt | Vol: XX • Sayı | No: X • ss | pp: XXXX-XXXX • Yıl | Year: 20XX

Application and evaluation of reinforcement learning for two-dimensional trajectory tracking in snake-like robots

Furkan Mezgil, Mustafa Can Bingol

ARTICLE INFO

Dates:
Received: 02.04.2026
Accepted: 05.08.2026

Doi:
10.65206/pajes.1922237

Corresponding author:
Furkan Mezgil
(furkanmezgil2016@gmail.com)

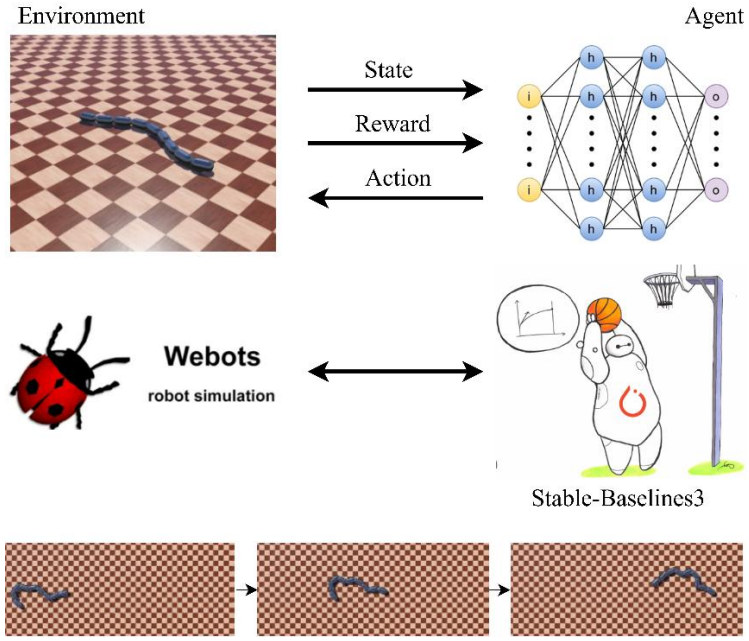
Author addresses:
Department of Electrical-Electronics Engineering, Faculty of Engineering-Architecture, Burdur Mehmet Akif Ersoy University, Burdur, 15100, Türkiye
(furkanmezgil2016@gmail.com; mcbingol@mehmetakif.edu.tr)

ABSTRACT

Context—Snake-like robots are biomimetic systems that can move effectively in narrow, complex, and restricted environments thanks to their modular and flexible body structures composed of numerous serially connected joints. These characteristics offer significant advantages, particularly in areas such as pipeline inspection, search and rescue operations, industrial maintenance applications, and exploration missions. The multiple degrees of freedom distributed along the body enable the robot to achieve high maneuverability but also make the control problem quite complex. Due to the dynamic interactions between segments, friction-based motion characteristics, and nonlinear system behavior, achieving reliable and accurate trajectory tracking emerges as a significant engineering problem.

Objective—In this study, a reinforcement learning (RL) based control method has been developed to solve the trajectory tracking problem for snake-like robots in a two-dimensional plane.

Method—In the proposed approach, the robot's dynamic model was created in the Webots simulation environment, an open-source simulation program, and all training and testing processes were carried out in this environment. During the learning process, policy-based RL algorithms from the Stable-Baselines library were used. In this context, Proximal Policy Optimization (PPO) and three different RL algorithms were used during the training process. To enable the robot to adapt to different orientation scenarios, seven different angles defined in the range of +45° to −45° and trajectories of varying lengths were used. Thus, the goal was for the agent to learn a generalizable control policy not only for a specific trajectory type but also for tracks with different slopes and orientations.



Results—The results obtained show that the PPO algorithm produced a higher average reward compared to other methods and exhibited a more stable learning process. After training was completed, the developed method was tested both on trajectories used during the training phase and on previously unseen trajectories. For the 0° trajectory, maximum errors were recorded as 0.093 m and 0.040 m for the x and y axes, respectively. Furthermore, the system exhibited robust generalization capabilities on a +22.5° trajectory, not encountered during the training phase, yielding maximum errors of 0.099 m and 0.052 m.

Conclusion—These findings demonstrate that the proposed RL-based control approach can effectively solve the two-dimensional trajectory tracking problem in snake robots. In future studies, the proposed method can be extended to the three-dimensional trajectory tracking problem, or it can be evaluated under more complex conditions, such as scenarios involving obstacles.

Key Words—Reinforcement learning, Snake-like robot control, Snake-like robot modeling, Trajectory tracking.

I. INTRODUCTION

The biomimetic design approach has found an increasingly broad application area in the field of robotics in recent years, and the transfer of efficient movement strategies developed by natural organisms in response to environmental stimuli to engineering solutions has attracted significant interest. Over long evolutionary processes, living organisms have developed high adaptability, enabling them to move in complex and unpredictable environments. Adapting these behaviors to robots is critically important, especially for robotic applications operating in confined spaces and on irregular surfaces. In this context, snake-like robots stand out as systems offering high mobility thanks to their multi-jointed designs inspired by the flexible body structure of snakes. The advantages of snake-like robots are particularly evident in areas such as advancing through narrow pipe systems [1], search and rescue under rubble [2], and exploration [3].

Despite these advantages, the multi-jointed and flexible structures of snake-like robots significantly complicate control problems. Control methods for snake-like robots, such as model-based, sinusoidal motion-based, and central pattern generator (CPG) based approaches, are mostly based on parameters optimized for a specific environment and task. Such systems cannot demonstrate flexibility in response to changes in the environment and may lose stability under dynamic conditions. Therefore, learning-based approaches in robot control, particularly reinforcement learning (RL) methods, are gaining prominence [4], [5]. They are advantageous because they can learn the optimal motion strategy directly through experience without requiring a precise dynamic model in advance, especially in complex and difficult to model systems such as snake robots.

For snake-like robots to perform reliably in real world environments, they require not only an effective control mechanism but also the ability to track a trajectory that ensures time dependent and stable progression toward the target. Trajectory tracking is the process of following a defined trajectory with minimal error within the desired timeframe [6]. In this process, the robot must precisely control both its spatial position and its time-dependent progression. Considering the multi jointed structure of snake-like robots, their friction-based motion characteristics, and the dynamic interactions between segments, tracking an orbit with high accuracy becomes a highly complex control problem. Therefore, trajectory tracking in snake-like robots is not only a requirement for error free path following but also a critical necessity in terms of maintaining timing sensitivity and adapting to environmental variables.

A. Related Works

Snake-like robots are being intensively researched for various working environments due to their biomimetic designs and high environmental adaptability. In underwater applications, it has been demonstrated that soft bodied, hydraulically driven snake-like robots can achieve stable swimming using serpentine based motion models and that the body shape can be detected in real time via sensors [7]. Within the scope of modular and reconfigurable approaches, snake-like systems capable of both movement and manipulation have been developed using tendon driven soft continuity units and spherical grippers [8]. In studies conducted in challenging and multi planar environments, movement strategies enabling snake-like robots to transition between planes by utilizing environmental contacts have been proposed, enhancing their exploration capabilities [9], [10]. Furthermore, the different walking patterns of snake-like robots on the ground have been analyzed using combined models based on the spine curvature and contact slip components [11]. Virtual leaders were defined for safe and autonomous movement, and line of sight (LOS) based guidance approaches were developed, with movement strategies defined considering collision avoidance scenarios [12]. In studies on snake-like robots operating on pipelines and cylindrical surfaces, systems capable of adapting to different diameters and surface conditions have been developed using geometric analyses, helical walking models, and radius estimation methods [13]-[15]. Furthermore, in snake-like robots developed for inspection and maintenance tasks in confined spaces, locomotion and manipulation capabilities have been addressed together using CPG-based controllers and integrated gripper mechanisms [16].

In parallel with this, studies addressing the path tracking problem in snake-like robots have shown that LOS-based methods are widely used. To reduce deviations during curved trajectory tracking, butterfly spiral thrust walking and integral LOS control were used together to reduce lateral drift in the steering angle and body position errors [17]. In a study on multi snake-like robot systems, synchronized LOS guidance rules and a high connectivity dynamic frequency compensator were used to maintain formation integrity under disturbances, ensuring that robot speeds were adjusted in a coordinated manner [18]. By suppressing time-varying external effects, optimized LOS guidance predicted position errors and interventions, adjusted the steering angle, and reduced lateral slip tendency [19]. A path following method combining fuzzy LOS guidance with backstepping-based slip mode control was proposed for snakelike robots with unknown parameters. Viscous friction coefficients were estimated using an adaptive adaptation strategy, and the parameters were compensated for in the torque input [20]. In another study that reduces lateral slip and uses coefficient estimation, an optimized LOS-based closed-loop structure was created using integral terms and virtual input variables, and unmeasurable conditions were approximated using virtual model variables [21]. For cases where drift limits tracking performance, a LOS guidance technique was developed to accurately determine robot orientation using a finite time stable lateral drift differentiator and auxiliary functions [22]. For snake-like robots operating under uncertain model parameters and lateral slip effects, directional slip and error oscillations were reduced using an improved integral LOS-based strategy [23]. Using an improved serpentine curve, variable amplitudes and unknown parameters were estimated online, and the path following performance of the multi jointed snake-like robot was improved [24]. To improve the steering of snake-like robots, the relationship between phase offsets at the pitch joints and robot orientation was investigated, and this relationship was combined with the LOS guidance law to obtain a path following algorithm applicable in challenging terrain conditions [25]. In another study, a dynamic model was developed for snake-like robots with spiral walking. By estimating variable viscous friction coefficients and external disturbances, it was demonstrated that path following errors were reduced and the robot could follow the ideal trajectory more accurately [26].

Commonly used LOS-based methods generate guidance commands based on the geometric relationship between the reference trajectory and the robot's position and are particularly effective under certain conditions. Therefore, they are insufficient for snake-like robots moving in complex environments. RL-based methods that learn through experience can control such systems. In this study, inspired by the LOS method, an RL-based trajectory tracking Proximal Policy Optimization (PPO) algorithm has been developed. Thus, a control framework capable of learning through experience has been presented. Furthermore, passive wheeled robots have been used in many existing studies [19], [25], [27]. In contrast, this study developed a wheelless snake-like robot to achieve more realistic movement. Unlike their wheeled robots, wheelless snake-like robots interact with the environment through friction, which introduces non-holonomic-like effects in a dynamic context. In a similar study, Twin Delayed Deep Deterministic Policy Gradient (TD3) and Deep Deterministic Policy Gradient

(DDPG) based methods were used for joint control [28]. The key point that distinguishes our method from similar studies is its ability to produce more adaptable and robust decisions in uncertain and variable environmental conditions without relying on deterministic models.

II. METHOD

The snake-like robot designed in this study has 8 degrees of freedom (DoF) and is shown in Fig. 1. These joints are connected sequentially, creating a continuous chain of motion along the robot body. The Denavit-Hartenberg (D-H) parameters for each joint are presented in Table 1. The L in the table represents the physical length of the joints and is considered a fixed value in robot design.

The transformations between the local coordinate systems defined for each joint were obtained using the transformation matrix given in (1):

$$ {}^{i-1}_{i}T = R_x(\alpha_{i-1})D_x(a_{i-1})R_z(\theta_i)D_z(d_i). \quad (1) $$

This allows the position of any point on the robot's body to be calculated precisely using the angular positions of the segments. To calculate the kinetic energy of the system, the velocities were obtained by taking the derivatives of the positions of the center of gravity of each joint. The obtained velocities were substituted into (2):

$$ T = \sum_{i=1}^{N} \frac{1}{2} m_i((\dot{x}_i)^2 + (\dot{y}_i)^2) + \sum_{i=1}^{N} \frac{1}{2} I_i(\dot{\theta}_i)^2. \quad (2) $$

Here, the joint's moment of inertia is denoted by I_i. Since there is no potential energy in the model, the kinetic energy expression is equal to the Lagrange function. When the obtained Lagrange function is substituted into (3), the dynamic model of the snake robot is obtained as in (4):

$$ \frac{d}{dt}\left(\frac{\partial \mathbb{L}}{\partial \dot{q}_k}\right) - \frac{\partial \mathbb{L}}{\partial q_k} - \sum_{i=1}^{\zeta} \lambda_i \left(\frac{\partial f_i}{\partial \dot{q}_i}\right) = 0, \quad (3) $$

where $\mathbb{L}$ denotes the Lagrange function, q and k denote the system parameters and the indices of the parameters, respectively. Additionally, λ, ζ, and f denote the Lagrange multiplier, the total number of constraints affecting the system, and the constraints, respectively. When (3) is solved, the dynamic model of the snake robot is obtained as (4):

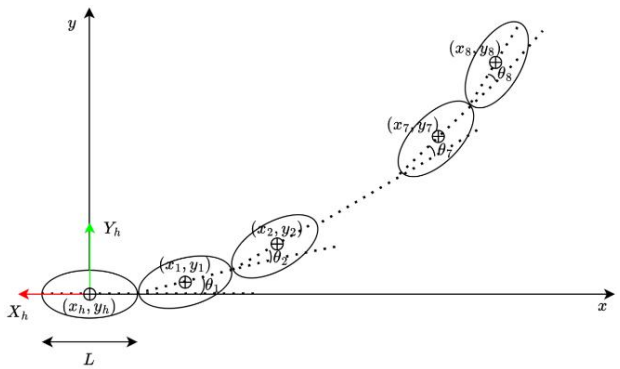


Fig. 1. Free-body diagram of the designed snake-like robot.

Table 1. D-H parameters.

i	a_{i-1}	α_{i-1}	d_i	θ_i
1	L	0	0	θ_1
2	L	0	0	θ_2
3	L	0	0	θ_3
4	L	0	0	θ_4
5	L	0	0	θ_5
6	L	0	0	θ_6
7	L	0	0	θ_7
8	L	0	0	θ_8

$$ M(\theta_i)\ddot{\theta} + C(\theta_i, \dot{\theta}_i)\dot{\theta}_i + B(\dot{\theta}_i) + K(\theta_i) = \tau_i, \quad (4) $$

where $\theta_i = [\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6, \theta_7, \theta_8]^T$. Furthermore, $M(\theta_i)$ is the inertia moment matrix; $C(\theta_i, \dot{\theta}_i)$ is the Coriolis effect matrix; $B(\dot{\theta}_i)$ is the damping matrix; $K(\theta_i)$ is the spring constant matrix; and τ_i is the matrix containing the applied moments. The open-source Webots simulation environment was used to simulate the created snake robot model. Each joint was created from ABS material using a 3D design program. Factoring in the battery and internal electronics, the frontal segment of the snake-like robot was estimated a mass of 500g, whereas each successive joint was determined to be 100g. Aside from the simulation time step, which was adjusted to 64 ms, the default specifications were preserved for the rest of the parameters.

The angle value of each joint is represented by an analytical function containing the parameters that determine the robot's overall waveform. The angle value to be applied to each joint was calculated using (5):

$$ \theta_i[k] = \alpha_i \sin(k\omega + \textstyle\sum_{n=0}^{i} \beta_i) + \gamma, \quad (5) $$

where α_i represents the amplitude coefficient defined for each joint, indicating the amplitude of the wave formed along the body and thus the magnitude of the joint movements. k represents the discrete time index, and ω represents the angular frequency of the wave. This parameter affects the robot's forward speed by determining the frequency of the wave formed along the body. β_i represents the phase shifts for each joint. The gradual increase of the phase components depending on the joint number ensures that the waveform gains a structure that progresses from the front to the rear of the robot body. The γ parameter represents a constant angle offset applied to the entire body. This term affects the robot's ability to change direction by shifting the entire waveform in a specific direction.

A. Reinforcement Learning

The study addresses an RL-based control framework to enable the snake-like robot to interact with its environment and perform its task. In RL, the robot uses information perceived from its environment to make specific movement decisions and improves its performance over time based on the results of these decisions. This process is shaped around a reward mechanism that guides robot behavior and an agent structure that processes this mechanism to produce decisions. According to the defined reward function, if the robot has passed the target position or if a positional deviation of more than 0.5 m from the objective is detected, -100 is assigned. In other cases, the other equation of the piecewise function applies. κ represents the position error and ψ represents the angular error. Values are measured in meters and radians. In this study, error tolerance of 10^{-3} meters and 10^{-3} radians were accepted. Due to this measurement metric and these acceptances, $\mathcal{C} = 10^3$ and $\mathcal{B} = 10^6$ were used. The reason for using a quadratic function is to emphasize that the error increases as the robot deviates from the reference trajectory. The reward function formulated in (6) incorporates a look-ahead mechanism deeply inspired by traditional LOS guidance laws:

$$ R[k] = \begin{cases} -100, & (\kappa[k] \geq 0.5) \text{ or} \\ & (ex[k] < 0 \text{ and } ey[k] < 0), \\ -\left(\frac{(\kappa[k]\mathcal{C})^2}{\mathcal{B}} + 0.2\frac{(\psi[k]\mathcal{C})^2}{\mathcal{B}}\right), & \text{otherwise} \end{cases} \quad (6) $$

The selection of reward function coefficients and error thresholds was carried out through an empirical tuning process via extensive preliminary simulations. In the study, the agent was modeled using multi-layer perceptron (MLP) architecture. The general architecture of the agent structure used is given in Fig. 2.

The MLP-based structure is designed as a fully connected network consisting of an input layer, hidden layers, and an output

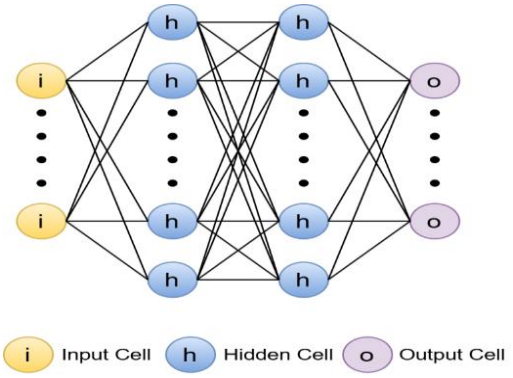


Fig. 2. MLP structure used.

layer. In traditional MLP architecture, the input layer takes the feature vector obtained from the environment or dataset and passes it to the first hidden layer without performing any computation. The n_f dimensional input vector $x = (x_1, x_2, \dots, x_f)$ undergoes a nonlinear transformation process in the hidden layer, and the layer outputs are obtained using the expression in (7):

$$h_1 = f(W_1 x + b_1) \tag{7}$$

Here, f is the activation function that performs the nonlinear transformation. With the number of neurons in each layer being k, W_1 represents the weight coefficient matrix of size $n_k \times n_f$, and b_1 represents the bias vector of size n_k [29]. In the MLP-based structure, the output produced by each layer forms the input for the subsequent layer. Therefore, a similar nonlinear transformation process is applied in each layer until reaching the output layer from the input layer.

In the MLP-based agent architecture used in this study, the input layer with 23 neurons represents the state vector obtained from the environment. The α, β, and γ values presented in Equation 5 contain information about the linear distance between the robot and the track (ed), the distance relative to the x-axis (ex), the distance relative to the y-axis (ey), the angular error between the robot and the track ($e\theta$), the robot's angle (θ), and the trajectory angle (ϑ). The state vector $s = [\alpha_1, \dots, \alpha_8, \beta_1, \dots, \beta_8, \gamma, ed, ex, ey, e\theta, \theta, \vartheta]$ is expressed in this way. This state information is processed through two sequentially placed hidden layers, each containing 64 neurons. In the output layer, 17 neurons are used because there are 17 variables in the action vector. The action vector $a = [\Delta\alpha_1, \dots, \Delta\alpha_8, \Delta\beta_1, \dots, \Delta\beta_8, \Delta\gamma]$ is expressed in this way. The values $\alpha_1, \dots, \alpha_8, \beta_1, \dots, \beta_8$, and γ are control parameters that are updated over time through the actions generated by the policy network learned by the agent. The policy network takes the state vector as input and produces outputs $\Delta\alpha_1, \dots, \Delta\alpha_8$, $\Delta\beta_1, \dots, \Delta\beta_8$, and $\Delta\gamma$, which represent the changes to be made. At each time step, these outputs are added to the current α, β, and γ values to update the parameters. Δ represents the changes, and the α, β, and γ changes are expressed in (5). The hyperbolic tangent function (tanh) in (8) is used as the activation function:

$$\tanh(x) = (e^x - e^{-x})/(e^x + e^{-x}). \tag{8}$$

As a result, when the reward function defined by (6) and the MLP-based agent structure are considered together, it becomes possible to learn robot behaviors in a stable manner. This framework provides a suitable infrastructure for applying different RL algorithms and forms a common basis for the methods discussed.

RL approaches can generally be categorized into two groups, value (Q) learning and policy (π) learning. Policy can be defined as a function $\pi : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ that maps the state space to probability distributions in the action space. The fundamental task of this function is to create a Markov Decision Process (MDP) that links the state space ($\mathcal{S}$) and the action space ($\mathcal{A}$). The construction of this process relies on integrating the components of the instantaneous state (s_t), the chosen action (a_t), and the resulting reward (r_t). These values are used in the discounted reward (R_t) equation as shown in (9):

$$R_t = \sum_{i=t}^{T} \varsigma^{i-t} r(s_i, a_i), \varsigma \in [0, 1]. \tag{9}$$

The parameter ς denotes the discount factor, where the core aim of policy-based RL involves seeking the maximum possible expected return from the onset, represented by the objective function $J = \mathbb{E}[R_1]$. The Bellman equation given in (10) is used in RL problems:

$$\mathcal{Q}^{\pi}(s_t, a_t) = \mathbb{E}\big[r_t + \varsigma \mathbb{E}[\mathcal{Q}^{\pi}(s_{t+1}, a_{t+1})]\big]. \tag{10}$$

The target policy $\mu : \mathcal{S} \leftarrow \mathcal{A}$ can be defined and is given in (11):

$$\mathcal{Q}^{\mu}(s_t, a_t) = \mathbb{E}\big[r_t + \varsigma \mathbb{E}[\mathcal{Q}^{\mu}(s_{t+1}, \mu(s_{t+1}))]\big]. \tag{11}$$

The loss function depends on Q-learning [30] and is shown in (12):

$$L(\theta^{Q}) = \mathbb{E}\left[\left(\mathcal{Q}(s_t, a_t|\theta^{Q}) - y_t\right)^2\right], \tag{12}$$

with

$$y_t = r_t + \varsigma \mathcal{Q}(s_{t+1}, \mu(s_{t+1})|\theta^{Q}). \tag{13}$$

In (12) and (13), the function parameters represent θ^{Q}. The PPO algorithm was used in the study to enable the snake robot to perform trajectory tracking using RL.

B. PPO Algorithm

The PPO algorithm is an improved version of policy optimization in the confidence region. This process is performed using only first-order optimization. The optimization process is performed using the proxy function ($\mathcal{L}$) in (14):

$$\mathcal{L}(\theta) = \hat{\mathbb{E}}_t\big[min(\mathcal{R}_t(\theta)\hat{A}_T, clip(\mathcal{R}_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_T)\big], \tag{14}$$

with

$$\mathcal{R}_t(\theta) = \pi_{\theta}(a_t\ s_t)/\pi_{\theta_{old}}(a_t\ s_t). \tag{15}$$

In equation (15), the variable $\mathcal{R}_t$ denotes the likelihood ratio, while the advantage estimations are symbolized by $\hat{A}_{1,\dots,T}$. Furthermore, the PPO methodology tailored for the actor-critic framework is illustrated in **Algorithm 1** [31].

When **Algorithm 1** is initiated, the simulation environment and batch are restarted. The agent interacts with the environment for 2048 steps to collect data; during this process, a noise parameter is added to the actions generated by the actor network. The main purpose of this process is to provide the agent with exploration capabilities. After each action, information about the current state, next state, action, and reward is stored in a batch. Upon reaching 2048 steps, advantage estimates are calculated using reward and value estimates. After the data collection process is complete, the optimization phase begins, and the 2048-step

Algorithm 1 PPO Algorithm

for episode: $1 \to M$ **do**
  Reset environment and obtain s_1
  Initialize Batch
  for iteration: $1 \to T$ **do**
    $a_t = \pi(s_t|\theta^{\pi}) + N_t$
    Get r_t s_{t+1} according to a_t
    $Batch \leftarrow [s_t, a_t, r_t, s_{t+1}]$
    Compute $\hat{A}_1, \dots, \hat{A}_T$
  end for
  Compute new weight by optimizing surrogate $\mathcal{L}$
  Update networks
end for

experience data is divided into minibatches of 64. The networks are optimized according to the loss function $\mathcal{L}$ specified in (14). Also, the discount factor was chosen as 0.99. All hyperparameters used during training were selected as described in the original PPO study [31]. Different from the original PPO study, the learning rate was reduced from 3×10^{-4} to 3×10^{-5} for fine-tuning.

III. RESULTS and DISCUSSION

In this study, an RL-based control approach has been developed to enable a snake-like robot to autonomously follow a reference trajectory. The robot's motion generation is provided by a sinusoidal-based motion model that defines the joint angles given in (5). The state vector is defined to include the robot's α, β, γ coefficients, ed, ex, ey distances, the angular error $e\theta$ between the robot and the track, and the θ, ϑ angles. The action vector consists of a total of 17 control variables corresponding to the robot's multi-joint structure.

The robot was trained over a total of 10^6 steps, and the performance of different RL algorithms was compared. The learning rate was set to 3×10^{-4} for the first 0.75×10^6 steps and then updated to 3×10^{-5} to allow for fine-tuning. The training graph in Fig. 3 shows the total reward changes for the PPO, DDPG, Soft Actor-Critic (SAC), and TD3 algorithms. The total training durations for PPO, DDPG, SAC, and TD3 were determined to be 4.021, 7.473, 5.508, and 4.666 hours, respectively. Also, descriptive analyses of training performance and statistical differences between algorithms are presented in Table 2. The table shows the algorithms based on the mean episodic training reward metric. Due to batch-size limitations, the PPO algorithm has 489 samples, whereas other algorithms have 1000. The Kolmogorov-Smirnov test was applied, and since the data were not normally distributed, the Kruskal-Wallis test was used. Upon examining the graph, it is observed that the PPO algorithm produces a higher reward compared to the other algorithms. The SAC and TD3 algorithms exhibit a more fluctuating learning process, while the DDPG algorithm has a lower reward profile compared to other methods. These results show that the PPO algorithm offers a more stable learning process in snake-like robot systems with continuous state and action spaces, such as trajectory tracking.

During training, the snake robot was made to follow the -45°, -30°, -15°, 0°, +15°, +30°, +45° trajectories. The robot's progress in simulation is shown in Fig. 4 with images taken every 10 steps.

The trajectory tracking performance during the 0° trajectory operation presented in Fig. 4 is shown in Fig. 5. Here, the blue line shows the trajectory, and the orange line shows the movement of the robot's head. The linear errors on the x and y axes and the angular errors along the z axis generated during this movement are shown in Fig. 6(a). Fig. 6(b) shows the actions performed to drive the robot along the 0° trajectory. Here, the actions applied to 8 different joints are visible. The actions for each joint are symbolized on the graph as 1,...,8. The angular profiles of the robot, determined for precise trajectory tracking, are displayed in Fig. 6(c). The coefficients of the motion equation resulting from the actions are given in Fig. 6(d).

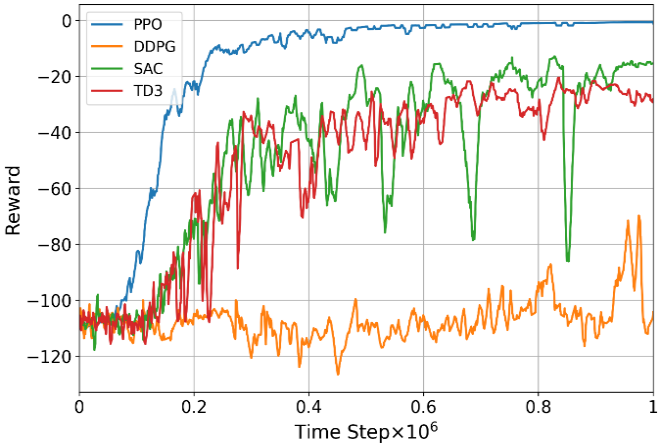


Fig. 3. Training performance of algorithms.

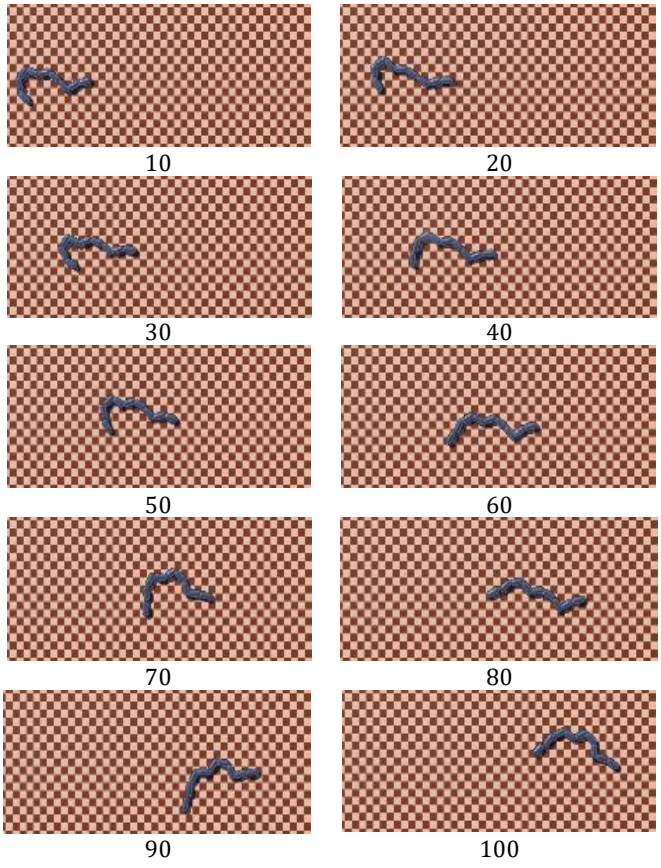


Fig. 4. The robot locomotion in 0° trajectory.

Table 2. Mean episodic training reward statistical analysis of algorithms.

Algorithm	$\bar{X}$±SD	Median(IQR)	Min	Max	Algorithm	p
PPO	-18.095 ± 32.157	-3.063 (9.719)	-110.082	-0.654	DDPG	<0.001*
					SAC	<0.001*
					TD3	<0.001*
SAC	-62.704 ± 36.620	-55.594 (80.426)	-117.809	-12.930	PPO	<0.001*
					DDPG	<0.001*
					TD3	<0.001*
TD3	-69.292 ± 35.344	-65.135 (73.655)	-115.653	-20.475	PPO	<0.001*
					DDPG	<0.001*
					SAC	<0.001*
DDPG	-106.999 ± 6.497	-107.030 (5.904)	-126.578	-69.718	PPO	<0.001*
					SAC	<0.001*
					TD3	<0.001*

The Kruskal-Wallis test was used. Significance values have been adjusted by the Bonferroni correction for multiple tests.

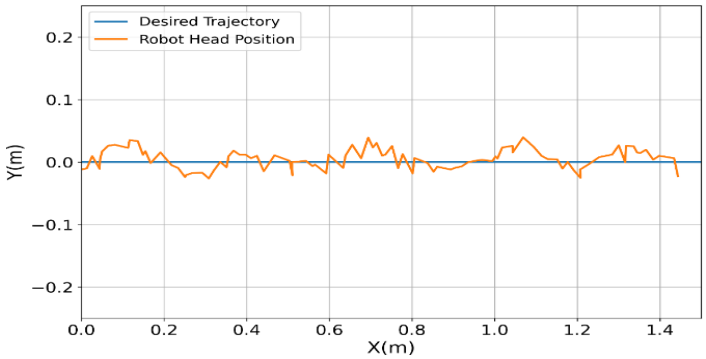


Fig. 5. Performance of 0° trajectory tracking.

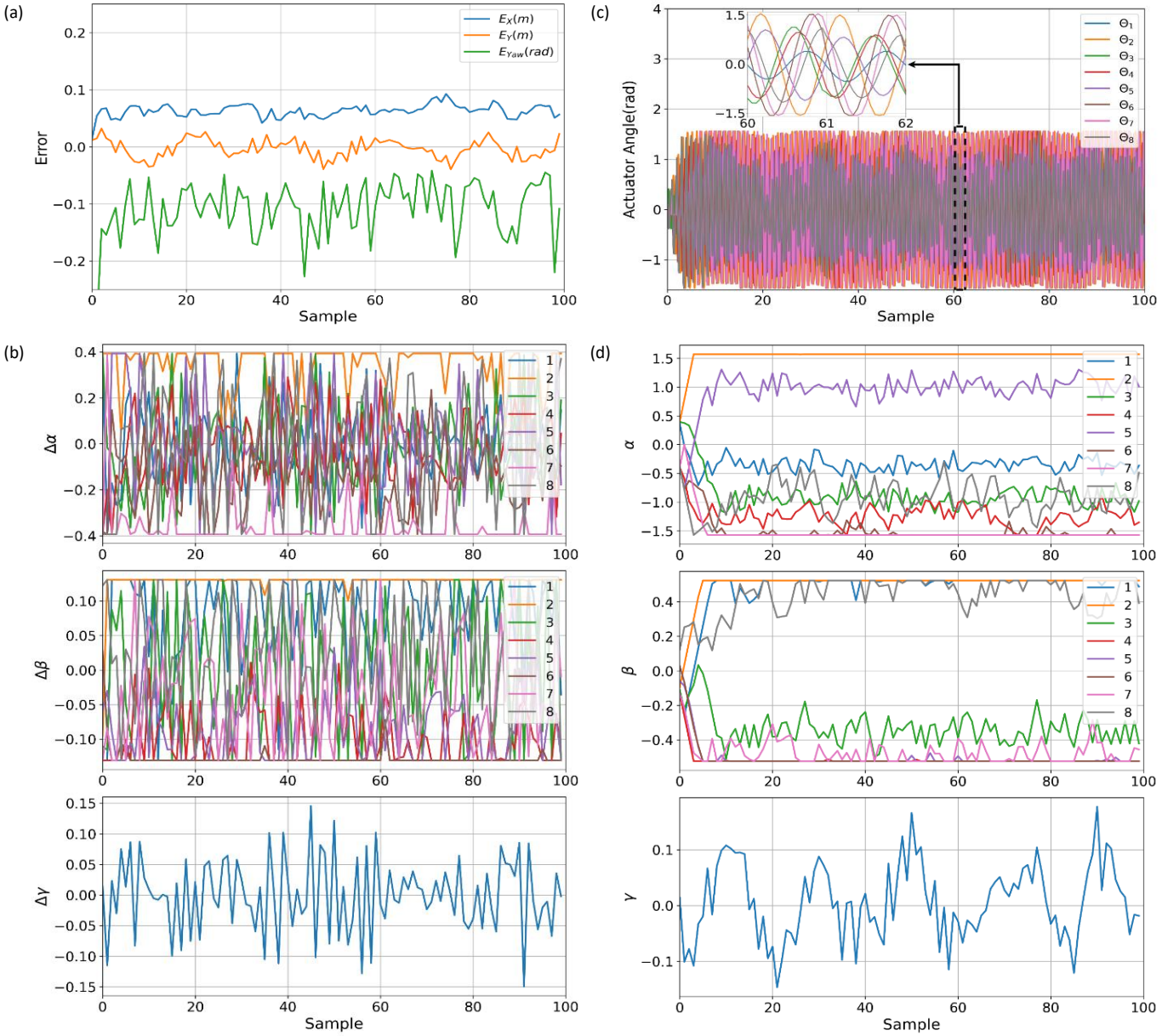


Fig. 6. Outcomes in 0° trajectory tracking: (a) Errors, (b) Actions, (c) Joint angles, (d) Coefficients.

The motion for a +22.5° trajectory never seen during the training process is visualized on Fig. 7. The linear errors on the x and y axes and the angular errors along the z axis generated during the robot's movement at a +22.5° trajectory are shown in Fig. 8(a). The control actions applied to the joints to enable the robot to move at a +22.5° trajectory is shown in Fig. 8(b). Fig. 8(c) displays the joint angle profiles determined for the robot to accurately follow the +22.5° trajectory. The coefficients of the motion equation resulting from the actions are given in Fig. 8(d).

The movement for another trajectory that the robot had never seen during its training process -22.5° is shown in Fig. 9. The robot performed the tracking with maximum errors of 0.076 m in the x-axis and 0.056 m in the y-axis.

A combined trajectory tracking task was also conducted, the results of which are depicted in Fig. 10. The snake-like robot the tracking of this trajectory with maximum errors of 0.079 m along the x-axis and 0.056 m along the y-axis.

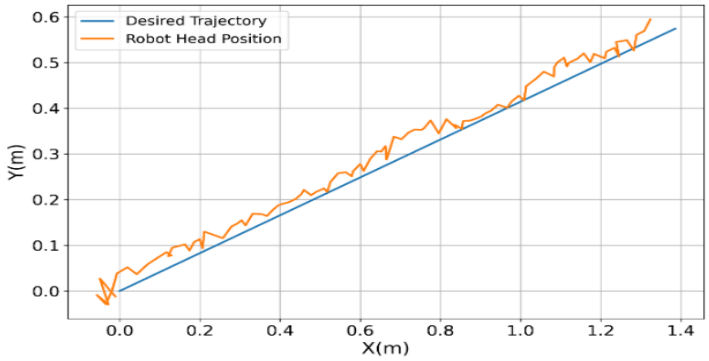


Fig. 7. Performance of +22.5° trajectory tracking.

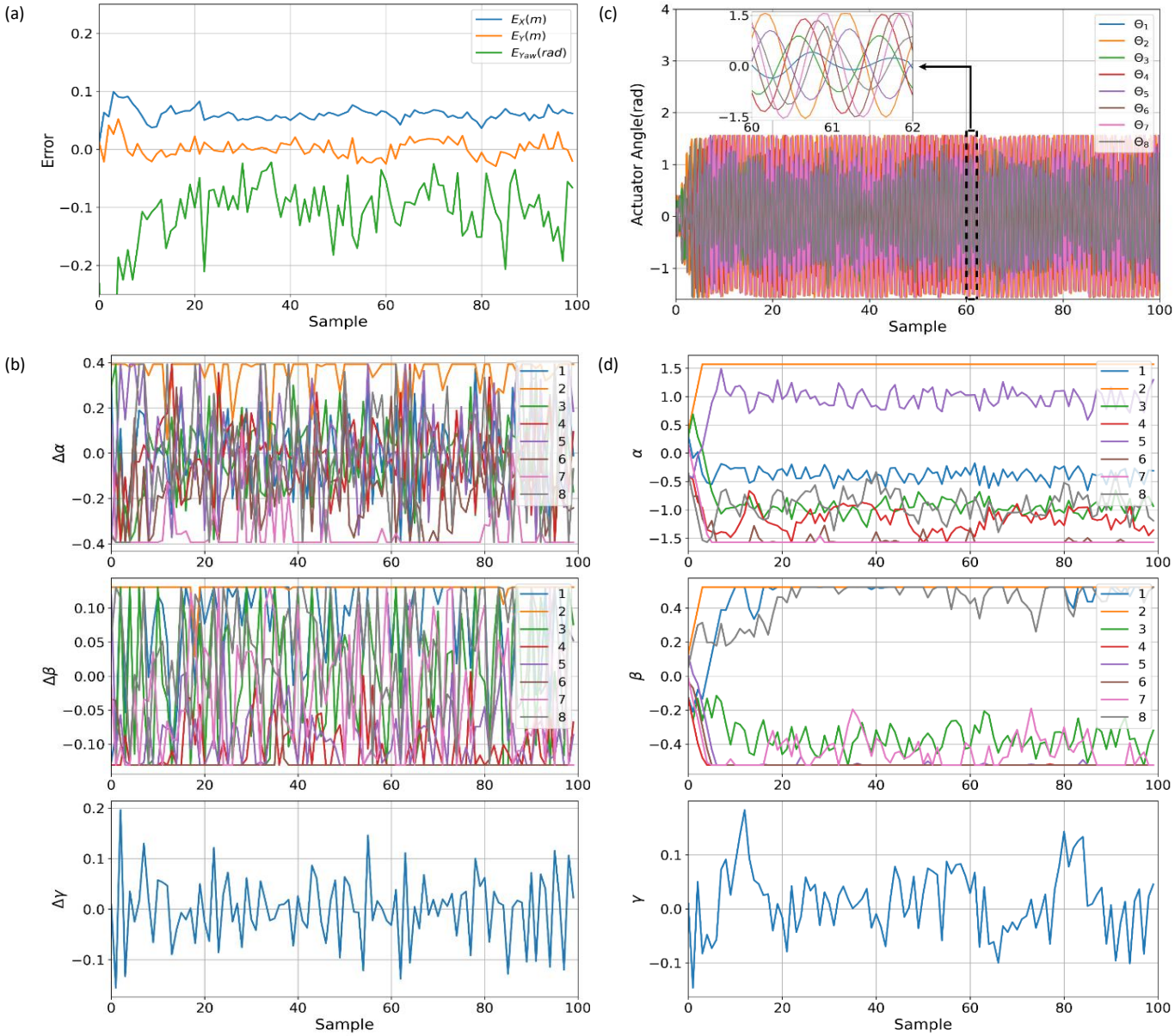


Fig. 8. Outcomes in +22.5° trajectory tracking: (a) Errors, (b) Actions, (c) Joint angles, (d) Coefficients.

This study has certain limitations. The study was conducted for an 8-DoF wheelless snake-like robot. Furthermore, the trajectory tracking performed is valid for 2-dimensional planes. The current study is confined to RL baselines. In the literature, trajectory tracking and motion control problems for snake-like robots are generally considered challenging problems due to the complex dynamics of systems with high degrees of freedom. The LOS method has been predominantly used in path tracking studies [17], [19], [23]. In contrast, this study uses an RL-based algorithm inspired by the LOS method. The difference between this study and previous studies in the literature is the use of a LOS-inspired RL algorithm. There is a similar study based on TD3 and DDPG [28]. In comparison, this study achieved better results with the PPO algorithm. When examining the results obtained in this study, it is seen that the robot tracking the trajectory is determined with a maximum error of 0.093 m and 0.040 m on 0° trajectory and 0.099 m and 0.052 m on +22.5° trajectory for the x and y axis, respectively. In conclusion, this study contributes to

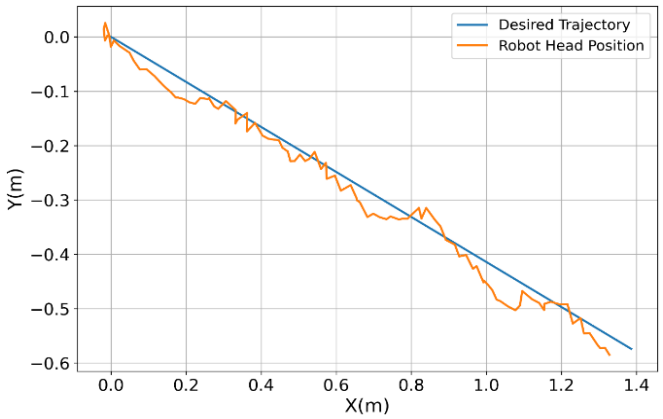


Fig. 9. Performance of -22.5° trajectory tracking.

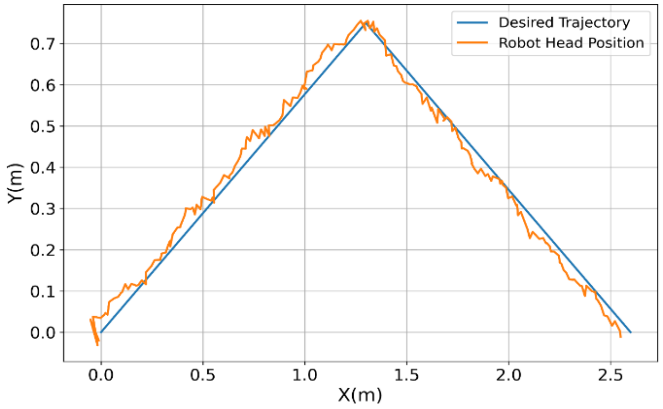


Fig. 10. Performance of combined trajectory tracking.

the literature by demonstrating that RL is a powerful and feasible alternative for solving the trajectory tracking problem in snake-like robots.

IV. CONCLUSION

Snake-like robots could move in various environments thanks to their structures inspired by biological snakes. These features offer significant advantages in many sectors, from pipe inspection to search and rescue. However, the high degree of freedom in the structure of these robots also brings control and motion planning problems. Therefore, developing an effective and generalizable trajectory tracking algorithm for snake-like robots is a critical requirement for practical applications. In this study, based on this need, the trajectory tracking problem of a snake-like robot in a two-dimensional plane is addressed.

Within the scope of this study, the snake robot was first modeled in the Webots simulation environment, considering realistic physical parameters and joint constraints. Policy-based algorithms from the Stable-Baselines library were used for trajectory tracking. Agent training was performed using PPO, DDPG, TD3, and SAC algorithms and variable length trajectories created for 7 different angles between +45° and -45°. When the results obtained were examined, it was observed that the PPO algorithm produced a higher average reward score compared to other algorithms. The developed method was evaluated using both seen trajectories from the training phase and unseen trajectories. The tests showed that the developed method enabled the snake robot to successfully track the presented trajectories.

Future work will focus on integrating CPG-based controllers to enhance locomotion/trajectory tracking and conducting a rigorous comparative study against these classical control paradigms to fully validate the real-world efficiency of the proposed method. Also, the proposed method can be extended to the three-dimensional trajectory tracking problem. In the 3D trajectory tracking process to be performed, it is planned to add new terms to the state vector, such as the position error along the z axis and the pitch angle error. Additionally, equation (5) will be updated based on an analysis of the climbing movements of snakes in nature. Furthermore, testing the algorithm on modular snake-like robot structures and examining its performance on robots with different numbers of joints and constraints could be an important research topic. Furthermore, the robustness of the controller can be evaluated under more complex conditions, such as uneven terrain, environments with variable friction coefficients or scenarios involving obstacles. In this regard, it is considered that the developed approach provides a flexible foundation that can be used in both simulation and real robot applications.

AUTHOR STATEMENT

Plagiarism Check—The article was scanned with iThenticate and found to be compliant with the journal's plagiarism policy.
Conflict of Interest—This article does not have any conflict of interest with any person or organization.
Ethics Committee Approval—Ethics committee approval is not required for this article.
Use of Artificial Intelligence Tools—In this study, DeepL and Grammarly were utilized to refine the manuscript's language, grammar, and overall stylistic clarity.
Funding—This study was supported by Burdur Mehmet Akif Ersoy University Scientific Research Projects Unit with the Project Number: 1206-YL-25.
Data availability—All data generated or analyzed during this study are included in this published article.
CRediT Author Contribution—Methodology, Software, Validation, Investigation, Data Curation, Writing - Original Draft, Visualization (**Furkan Mezgil**); Conceptualization, Writing - Review & Editing, Supervision, Project Administration (**Mustafa Can Bingol**).
Acknowledgment—This article is derived from the master's thesis entitled "Development of a Reinforcement Learning-Based Two-Dimensional Tracking Algorithm for Use in Snake-Like Robots", conducted by Furkan Mezgil at the Burdur Mehmet Akif Ersoy University under the supervision of Mustafa Can Bingol.

REFERENCES

[1] T. Takemori, M. Tanaka, F. Matsuno, "Adaptive helical rolling of a snake robot to a straight pipe with irregular cross-sectional shape", *IEEE Transactions on Robotics*, 39(1), 437–451, 2023. https://doi.org/10.1109/TRO.2022.3189224.

[2] G. Çetin, E. Aksakal, S. Taskın, "Göçük altında kalanların tespiti için biyonik yılan robot tasarımı", *Journal of Materials and Mechatronics: A*, 3(1), 63–78, 2022. https://doi.org/10.55546/jmm.1079962.

[3] X. Hou, Y. Shi, L. Li, Y. Tian, Y. Su, T. Ding, Z. Deng, "Revealing the mechanical characteristics via kinematic wave model for snake-like robot executing exploration of lunar craters", *IEEE Access*, 8, 38368–38379, 2020. https://doi.org/10.1109/ACCESS.2020.2971221.

[4] X. Liu, C. D. Onal, J. Fu, "Reinforcement learning of cpg-regulated locomotion controller for a soft snake robot", *IEEE Transactions on Robotics*, 39(5), 3382–3401, 2023. https://doi.org/10.1109/TRO.2023.3286046.

[5] F. Sheng, F. Wan, C. Tang, X. Guo, "A maneuverable winding gait for snake robots based on a delay-aware swing and grasp framework combining rules and learning methods", *IEEE Robotics and Automation Letters*, 10(1), 311–318, 2025. https://doi.org/10.1109/LRA.2024.3506274.

[6] Y. Li, C. Wei, Q. Wu, P. Chen, Y. Jiang, Y. Li, "Study of 3 dimension trajectory tracking of underactuated autonomous underwater vehicle", *Ocean Engineering*, 105, 270–274, 2015. https://doi.org/10.1016/j.oceaneng.2015.06.034.

[7] H. Shi, Y. Meng, W. Cui, M. Rao, S. Wang, Y. Xie, "Biomimetic underwater soft snake robot: Self-motion sensing and online gait control", *IEEE Transactions on Robotics*, 41, 1193–1210, 2025. https://doi.org/10.1109/TRO.2025.3530349.

[8] Y. Cai, H. Xu, Y. Wang, D. Chen, W. Matusik, W. Shou, Y. Chen, "Modular self-reconfigurable continuum robot for general purpose loco-manipulation", *IEEE Robotics and Automation Letters*, 10(2), 1976–1983, 2025. https://doi.org/10.1109/LRA.2025.3526560.

[9] Y. Yoshida, C. W. Chin, M. Tanaka, "Traversing between two planes using obstacle-aided locomotion of a snake robot", *IEEE Robotics and Automation Letters*, 9(11), 10288–10294, 2024. https://doi.org/10.1109/LRA.2024.3468154.

[10] S. Suyama, M. Nakajima, H. Arita, M. Tanaka, "Control of a snake robot with proximity sensors to adapt for two variable planes", *IEEE Access*, 12, 46864–46880, 2024. https://doi.org/10.1109/ACCESS.2024.3382205.

[11] W. Huang, Y. Fang, X. Guo, H. Liu, L. Liu, "A unified motion modeling approach for snake robot's gaits generated with backbone curve method", *IEEE Transactions on Robotics*, 40, 4132–4146, 2024. https://doi.org/10.1109/TRO.2024.3420803.

[12] Y. Xiu, H. Deng, D. Li, R. Law, E. Q. Wu, L. Zhu, "Collision avoidance regulation-compliant orientation guidance and maneuvering control approach for snake robots", *IEEE Transactions on Industrial Electronics*, 71(9), 10955–10965, 2024. https://doi.org/10.1109/TIE.2023.3342304.

[13] B. A. Elsayed, T. Takemori, M. Tanaka, F. Matsuno, "Mobile manipulation using a snake robot in a helical gait", *IEEE/ASME Transactions on Mechatronics*, 27(5), 2600–2611, 2022. https://doi.org/10.1109/TMECH.2021.3114168.

[14] W. Huang, X. Guo, H. Liu, Y. Fang, "A robust model-based radius estimation approach for helical climbing motion of snake robots", *IEEE/ASME Transactions on Mechatronics*, 28(6), 3284–3293, 2023. https://doi.org/10.1109/TMECH.2023.3256419.

[15] Q. Li, W. Zhao, "Design of a modular pipeline robot structure and passing ability analysis", *IEEE Access*, 11, 97978–97989, 2023. https://doi.org/10.1109/ACCESS.2023.3312171.

[16] Z. Ji, G. Song, F. Wang, Y. Li, A. Song, "Design and control of a snake robot with a gripper for inspection and maintenance in narrow spaces", *IEEE Robotics and Automation Letters*, 8(5), 3086–3093, 2023. https://doi.org/10.1109/LRA.2023.3265591.

[17] D. Li, B. Zhang, C. Wu, Y. Xu, J. Huang, E. Q. Wu, L. Zhu, "Tracking control of snake robots with butterfly spiral propulsion for multiscenario applications", *IEEE Transactions on Industrial Informatics*, 20(12), 13526–13536, 2024. https://doi.org/10.1109/TII.2024.3407866.

[18] D. Li, B. Zhang, R. Law, E. Q. Wu, X. Xu, "Error constrained-formation path-following method with disturbance elimination for multisnake robots", *IEEE Transactions on Industrial Electronics*, 71(5), 4987–4998, 2024. https://doi.org/10.1109/TIE.2023.3288202.

[19] D. Li, Y. Zhang, P. Li, R. Law, Z. Xiang, X. Xu, L. Zhu, E. Q. Wu, "Position errors and interference prediction-based trajectory tracking for snake robots", *IEEE/CAA Journal of Automatica Sinica*, 10(9), 1810–1821, 2023. https://doi.org/10.1109/JAS.2023.123612.

[20] Y. Xiu, D. Li, H. Deng, S. Jiang, E. Q. Wu, "Path-following based on fuzzy line-of-sight guidance for a bionic snake robot with unknowns", *IEEE/ASME Transactions on Mechatronics*, 28(6), 3167–3179, 2023. https://doi.org/10.1109/TMECH.2023.3254817.

[21] D. Li, L. Zeng, Y. Xiu, Z. Pan, D. Zhang, H. Deng, "Sideslip elimination and coefficient approximation-based trajectory tracking control for snake robots", *IEEE Transactions on Industrial Informatics*, 19(8), 8754–8764, 2023. https://doi.org/10.1109/TII.2022.3220846.

[22] Y. Xiu, D. Li, M. Zhang, H. Deng, R. Law, Y. Huang, E. Q. Wu, X. Xu, "Finite-time sideslip differentiator-based los guidance for robust path following of snake robots", *IEEE/-CAA Journal of Automatica Sinica*, 10(1), 239–253, 2023. https://doi.org/10.1109/JAS.2022.106052.

[23] D. Li, J. Zhou, Y. Huang, D. Zhang, P. Li, A. Song, "Integral line of sight guidance scheme-based tracking method for snake robots", *IEEE Transactions on Automation Science and Engineering*, 22, 4537–4547, 2023. https://doi.org/10.1109/TASE.2023.3327958.

[24] D. Li, Z. Pan, H. Deng, L. Hu, "Adaptive path following controller of a multijoint snake robot based on the improved serpenoid curve", *IEEE Transactions on Industrial Electronics*, 69(4), 3831–3842, 2022. https://doi.org/10.1109/TIE.2021.3075851.

[25] Z. Cao, D. Zhang, M. Zhou, "Direction control and adaptive path following of 3-d snake-like robot motion", *IEEE Transactions on Cybernetics*, 52(10), 10980–10987, 2022. https://doi.org/10.1109/TCYB.2021.3055519.

[26] D. Li, Y. Zhang, W. Tong, P. Li, R. Law, X. Xu, L. Zhu, E. Q. Wu, "Anti-disturbance path-following control for snake robots with spiral motion", *IEEE Transactions on Industrial Informatics*, 19(12), 11929–11940, 2023. https://doi.org/10.1109/TII.2023.3254534.

[27] Z. Bing, C. Lemke, F. O. Morin, Z. Jiang, L. Cheng, K. Huang, A. Knoll, "Perception-action coupling target tracking control for a snake robot via reinforcement learning", *Frontiers in Neurorobotics*, 14, 591128, 2020. https://doi.org/10.3389/fnbot.2020.591128.

[28] Y. A. Baysal, I. H. Altas, "Deep reinforcement learning-based control of snake robot joints", *Arabian Journal for Science and Engineering*, 51(8), 10937-10962, 2026. https://doi.org/10.1007/s13369-025-10600-4.

[29] Y. A. Baysal, "Learning-based adaptive motion control of a snake robot", *Ph.D. dissertation*, Karadeniz Technical University, Trabzon, Türkiye, 2022.

[30] C. J. Watkins, P. Dayan, "Q-learning", *Machine learning*, 8(3-4), 279–292, 1992. https://doi.org/10.1023/A:1022676722315.

[31] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, "Proximal policy optimization algorithms", *arXiv*, 2017. https://doi.org/10.48550/arXiv.1707.06347.