A Maneuverable Winding Gait for Snake Robots Based on a Delay-Aware Swing and Grasp Framework Combining Rules and Learning Methods

Fengwei Sheng, Fuxi Wan, Chaoquan Tang, and Xian Guo, *Member, IEEE*

***Abstract*—Due to the high redundant degree of freedom characteristics of snake robots, their joint lever arms tend to be very long and often result in torque saturation, especially in the case of inter-tree motion. Traditional static planning methods based on curve segments connecting or wave functions are often limited by torque saturation and cannot meet the requirements of inter-tree motion of snake robots. Therefore, in this letter, a delay-aware swing and grasp framework combining rules and learning methods (DSG) is proposed for extending the inter-tree motion capability of snake robots. Specifically, first, to overcome the torque saturation problem, a joint torque direction determination rule is proposed to fully utilize the kinetic energy of the snake robot and enable the robot to move in the desired manner. Then, the DSG reduces the exploration space of the policy and guarantees the performance under delay, and based on which a maneuverable winding gait is designed to enable it to wrap around the target horizontal branch with maneuverability. Simulation and sufficient experiment results demonstrate that the proposed reinforcement learning (RL) controller has low torque requirement, high robustness under high delay, fast motion speed, and high generalizability.**

***Index Terms*—Biologically-inspired robots, motion control, reinforcement learning.**

I. Introduction

Due to the various gaits and unique body structure of snake robots, they can adapt to complex and dynamic environments, making them promising for use in field exploration. To bring this application into reality, researchers have conducted extensive studies on the motion control of snake robots, such as concertina gait for narrow passages [1], side-winding gait for sandy soil [2], crawler gait for rough terrain [3], helical rolling gait for pipes and trees [4]. These various gaits enable snake robots to adapt to different environments, making them promising for use in field exploration.

However, for a very common environment, trees, past research has been limited to climbing trunks, including trunks with varying radius, trunks with branch points and so on [5], [6]. There is very little research on how to move from a trunk to a branch, and from one branch to another. For moving from a trunk to a branch, a very intuitive approach is to first transfer a segment of the snake robot to the branch and wrap around it. Then, the segment still wrapped around the trunk unwinds. Now the snake robot is composed of two parts: one is the fixed segment wrapped around the branch and the other is the moving segment hanging in the air. Finally, the moving segment is controlled to also wrap around the branch, completing the movement from the trunk to the branch. For moving from one branch to another, a very intuitive approach is to first divide the snake robot into two parts: a fixed segment and a moving segment. Then, transfer the moving segment to the new branch and wrap around it. Then, the fixed segment wrapped around the original branch unwinds, completing the movement from one branch to another. However, due to the highly redundant degrees of freedom, joint lever arms are often quite long. Directly employing traditional static planning and a position PD controller to control the moving segment to wrap around the branch as mentioned above will result in torque saturation. Therefore, we imitate the swinging motion of a swing to accumulate mechanical energy of the system and leverage the conversion between kinetic and potential energy to overcome the torque saturation problem.

Past researchers have done a lot of research on the swinging motion. A common research object is multi-link robots. In [7], [8], [9], [10], [11], [12], energy-based control methods were introduced to the swing-up control problems for multi-link robots like Acrobot and Pendubot and so on. Other methods like trajectory tracking [13], [14], [15], fuzzy control [16], [17], [18], [19], [20] and intelligent control [21], [22], [23], [24] have also proven to be effective. These works have also inspired researchers to study brachiation robots [25], [26], [27]. However, these methods cannot be directly applied to snake robots due to the highly redundant degrees of freedom. To solve this problem, Li proposed a segmented control method for the discontinuous rod spanning motion combining deep reinforcement learning and energy based control methods [28]. However, physical experiments were not provided.

In this article, a delay-aware swing and grasp framework combining rules and learning methods (DSG) is proposed to deal

Received 8 July 2024; accepted 4 November 2024. Date of publication 25 November 2024; date of current version 4 December 2024. This article was recommended for publication by Associate Editor X. LI and Editor C. Gosselin upon evaluation of the reviewers' comments. This work was supported by the National Natural Science Foundation of China under Grant 62073176, Grant 62073328, and Grant 62233011. *(Corresponding author: Xian Guo.)*

Fengwei Sheng, Fuxi Wan, and Xian Guo are with the Institute of Robotics and Automatic Information System, College of Artificial Intelligence, Nankai University, Tianjin 300350, China (e-mail: shengfengwei@mail.nankai.cn; wanfuxi@mail.nankai.cn; guoxian@nankai.edu.cn).

Chaoquan Tang is with the Jiangsu Key Laboratory of Mine Mechanical and Electrical Equipment, School of Mechatronic Engineering, China University of Mining and Technology, Xuzhou 221116, China (e-mail: tangchaoquan@cumt.edu.cn).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2024.3506274, provided by the authors.

Digital Object Identifier 10.1109/LRA.2024.3506274

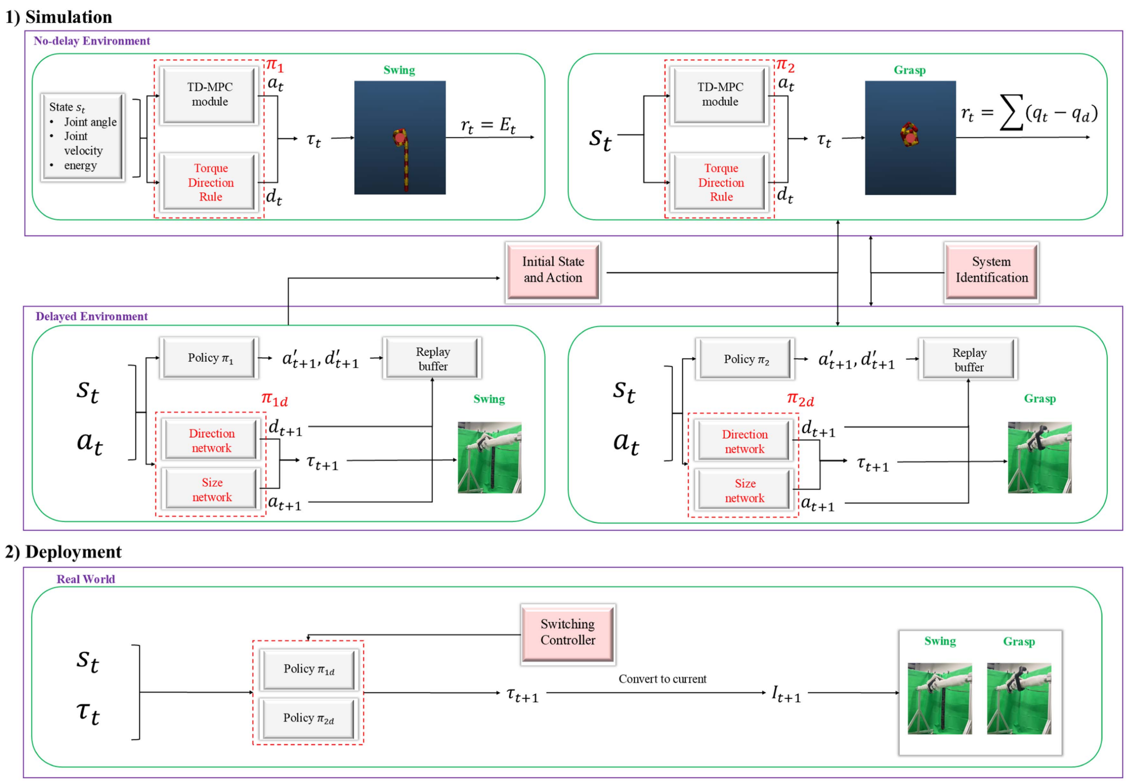


Fig. 1. The overview of the delay-aware swing and grasp framework. In simulation, the snake robot first learns a policy π_1, π_2 to perform swinging and grasping in a no-delay environment. Then, it imitates the policy π_1, π_2 to learn policy π_{1d}, π_{2d} in a delayed environment. Initial states and actions of the grasp phase are generated by the policy π_{1d}. During deployment, the torques generated by the policy π_{1d}, π_{2d} are converted into currents and sent to the snake robot. A switching controller is designed to switch between the swing policy and the grasp policy.

with the aforementioned problems. First, to overcome the torque saturation problem, a new joint torque direction determination rule is proposed, which ensures the snake robot moves like a swing and leverages the conversion between kinetic and potential energy. Secondly, to achieve sim-to-real transfer, double network delayed imitation with Dagger (DDIDA) is proposed for time delays in real-world experiments. Finally, a maneuverable winding gait based on DSG is designed to enable the snake robot to wrap around the horizontal branch with maneuverability. Compared with existing results, the main contributions of this article are summarized as follows.

1) A new joint torque direction determination rule is proposed, which ensure the snake robot leverages the conversion between kinetic and potential energy to overcome the torque saturation problem.
2) A novel framework DSG is proposed, based on which a maneuverable winding gait is designed to enable the snake robot to wrap around the horizontal branch with maneuverability.
3) The proposed DSG is successfully transferred into physical experiments with superior control performance. To the best of our knowledge, the proposed method achieves the first successful result for the maneuverable winding motion of a snake robot in the real world.

II. Methodology

The overview of DSG for snake robots is shown in Fig. 1. In this section, the overall training process of the policy for the maneuverable winding gait is introduced in detail, including the joint torque direction determination rule, the training framework of DSG and system identification.

A. Joint Torque Direction Determination Rule

Due to the highly redundant degrees of freedom characteristics of snake robots, the observation dimension is relatively very high and the exploration space of the policy is large. When the exploration space is too large, reinforcement learning (RL) often requires high-level reward shaping to guide the training process to enable the robot to move in the desired manner. However, high-level reward shaping often requires a lot of time for hyperparameter tuning. Therefore, we propose a joint torque direction determination rule to ensure that the snake robot's motion closely resembles the swinging motion of a swing. Our rule doesn't require complex reward shaping, reduce the exploration space of the reinforcement learning method and speed up the convergence speed of the policy, and finally get high-return policy.

Spong proposed an energy pumping method [7], widely utilized for the swing control of two-linked robots, as it does not need to model the system's dynamics. However, further expansion is needed to apply this method to snake robots. Li proposed a segmented control method [28], which divides the swing part of the snake robot into three parts: the swing joint, the self-locking segment, and the adjustment segment. The energy pumping method is extended to the snake robot by controlling the curvature of the adjustment segment. However, for the swing control of snake robots, the self-locking segment

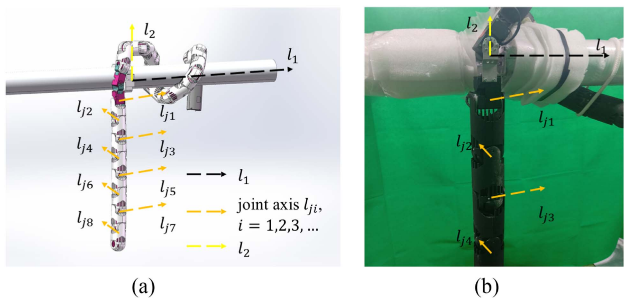
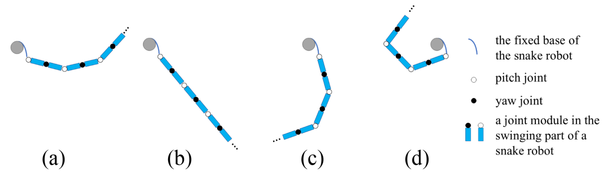


Fig. 2. The experimental setup for the maneuverable winding gait of the snake robot. (a) The 3D schematic of the experiment platform for the maneuverable winding gait of the snake robot, where l_1 is the reference axis parallel to the horizontal branch, l_2 is the reference axis perpendicular to the ground, and l_{ji} is the joint axis of the i-th joint. The joint id from top to bottom are numbered sequentially from 1 to 8. (b) The front view of the physical experiment platform, where the meaning of l_1, l_2, l_{ji} is the same as in Fig. 2(a).



(a)
(b)
(c)
(d)
the fixed base of the snake robot
pitch joint
yaw joint
a joint module in the swinging part of a snake robot

Fig. 3. The schematic of the top view of the maneuvering winding gait (a) The highest point on the right. (b) The snake robot is straightened. (c) The lowest point. (d) The highest point on the left.

is difficult to lock the joint angle to 0 solely based on its torque. And a significant portion of the exploration space with high returns is reduced by controlling the curvature of the adjustment segment. Therefore, we propose a new joint torque direction determination rule as follows.

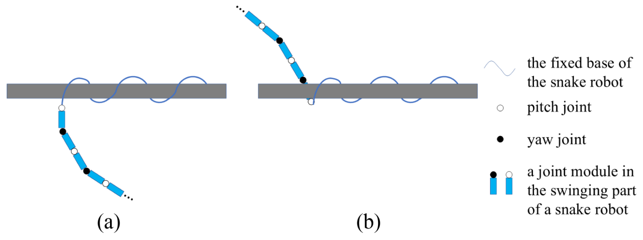
The experimental setup for the maneuverable winding gait of the snake robot is shown in Fig. 2. The angle between l_1 and l_{j1} is α. The angle between l_2 and l_{j1} is β, $\beta = 90^\circ$. The joint torque direction determination rule is shown as follows:

$$\begin{aligned} &sgn(T_i) = sgn(v_1), \text{ in the swing phase} \\ &sgn(T_i) = 1, \text{ in the grasp phase} \end{aligned} \tag{1}$$

Where T_i is the torque of the i-th joint, v_1 is the first joint velocity, sgn is the sign function. In the swing phase, for the pitch joint $j_1, j_3, \ldots$, the torque direction is the same as the first joint velocity to ensure the increase in mechanical energy of the system. For the yaw joint $j_2, j_4, \ldots$, the torque direction is the same as the first joint velocity to avoid the collision with the fixed base of the snake robot during the grasp phase. In the grasp phase, as it is necessary to ensure that the swing part of the snake robot wraps around the horizontal branch, all joint torque directions are positive.

For our rule in the swing phase, we analyzed the pitch and yaw joints separately.

Our rule for pitch joints is inspired by the motion of a swing. The schematic of the left view of the maneuvering winding gait is shown in Fig. 3. In Fig. 3(a), the velocity of the first joint is positive. From Fig. 3(a) to 3(b), the snake robot returns to a straightened state at the highest point, minimizing the gravitational potential energy's loss. From Fig. 3(b) to 3(c) and Fig. 3(c) to 3(d), the snake robot curls up to overcome gravity



(a)
(b)
the fixed base of the snake robot
pitch joint
yaw joint
a joint module in the swinging part of a snake robot

Fig. 4. The schematic of the left view of the maneuvering winding gait (a) The highest point on the right. (b) The highest point on the left.

and centrifugal force and increase mechanical energy. Then, the velocity of the first joint turns to negative. The snake robot repeats the process of energy pumping like a swing.

From Fig. 3(a) to 3(b), target joint angles are always larger than joint angles at the moment. From Fig. 3(b) to 3(c), the larger the joint angles of the snake robot, the greater the degree of curling up and the more mechanical energy is increased. Thus, target joint angles are always larger than joint angles at the moment. Therefore, from Fig. 3(a) to 3(d) the torque direction is positive. Similarly, when the velocity of the first joint is negative, the torque direction is negative.

Our rule for yaw joints is to prevent collisions. The schematic of the top view of the maneuvering winding gait is shown in Fig. 4. When the velocity of the first joint is positive, the snake robot moves from Fig. 4(a) to 4(b). Positive torque direction of yaw joints can make the snake robot swing to the left, avoiding collisions with the base during grasping. When the velocity of the first joint is negative, the torque direction is negative. This setup makes the yaw joints similar to the pitch joints in front, increasing mechanical energy during motion. This rule applies to situations where α is relatively small. When α is too large, it may result in the snake robot not having enough height to wrap. Here we specifically set $0^\circ < \alpha < 10^\circ$.

For our rule in the grasp phase, generally, target joint angles are larger than joint angles at the moment the system switches to the grasp phase. Thus, all joint torque directions are positive.

B. Delay-Aware Swing and Grasp Framework

The delay-aware swing and grasp framework is composed of three parts, training in a no-delay environment, training in a delayed environment, and real-world deployment. The joint torque direction determination rule enables the robot to move in the desired manner and accelerates the training of policy π_1, π_2 in a no-delay environment. As a teacher, policy π_1, π_2 guides the training of policy π_{1d}, π_{2d} in a delayed environment. Initial states and actions of the grasp phase are generated by the policy π_{1d}. Finally, policy π_{1d}, π_{2d} is deployed to the real world, where a switching controller is designed to switch between the swing policy and the grasp policy.

1) Training in a No-Delay Environment: The training process in a no-delay environment is shown in Fig. 1.

We use temporal difference learning for model predictive control [29] (TD-MPC) as our reinforcement learning method, which is a state-of-the-art model-based RL method published in 2022. TD-MPC is a RL approach that combines model predictive control. During the training process, it uses neural networks to fit the environment model, action value function, and reward function, updating the policy by maximizing the action value function. When selecting actions, it generates several sequences of actions $\mathbf{a}_{pi}$ through the policy and the environment model and generates several sequences of actions $\mathbf{a}_N$ based on the normal distribution and the environment model. The action sequences $\mathbf{a}_{pi}$ and $\mathbf{a}_N$ are then scored using the action value function and the reward function. The first action of the highest-scoring action sequence is selected as the executing action, while the remaining actions are treated as the mean of the normal distribution for selecting actions at the next time step.

For the training of the swing phase, the goal is to generate suitable action according to the no-delay state feedback of the snake robot, thereby enabling the robot to move like a swing and increase its mechanical energy. For the training of the grasp phase, the goal is to generate suitable action according to the no-delay state feedback of the snake robot, thereby enabling the robot to wrap around the horizontal branch. The training process can be formulated as a Markov decision process (MDP). At the current time step t, the robot state, denoted as $\boldsymbol{s_t}$, is sent to the policy π_1, π_2 to generate action $\boldsymbol{a_t}$ and direction $\boldsymbol{d_t}$, where $\boldsymbol{a_t}$ is the magnitude of joint torques and $\boldsymbol{d_t}$ is the direction of joint torques. Combining $\boldsymbol{a_t}$ and $\boldsymbol{d_t}$ we get joint torques $\boldsymbol{\tau_t}$, which is sent to the snake robot. Subsequently, the robot transfers to a new state $\boldsymbol{s_{t+1}}$ and returns the reward r_t. During the training process, TD-MPC continuously iterates through this cycle and optimizes the policy to maximize the expectation of the cumulative reward.

a) State Space: At the current time step t, $\boldsymbol{s_t} = [\boldsymbol{q}, \boldsymbol{dq}, E_t] \in \mathbf{R}^{2n+1}$ denotes the robot state, where $\boldsymbol{q} \in \mathbf{R}^n$ denotes the joint angles, $\boldsymbol{dq} \in \mathbf{R}^n$ denotes the joint velocities, $E_t \in \mathbf{R}$ denotes the mechanical energy of the swing part of the snake robot, $n \in \mathbf{R}$ denotes the number of movable joints.

b) Action Space: At the current time step t, the policy network takes the robot state $\boldsymbol{s_t}$ as input and then outputs the action $\boldsymbol{a_t}$, which is the magnitude of joint torques. We assume that there is no delay in the system. Therefore, the snake robot is able to return the current time step state $\boldsymbol{s_t}$, and send it to the torque direction rule module, which outputs the current time step torque direction $\boldsymbol{d_t}$. Combining $\boldsymbol{a_t}$ and $\boldsymbol{d_t}$ we get joint torques $\boldsymbol{\tau_t}$.

c) Reward Function: When the snake robot performs the joint torques $\boldsymbol{\tau_t}$, the environment returns a reward r_t. In the swing phase, to enable the snake robot to increase its mechanical energy and swing high enough to wrap around the horizontal branch, the reward function r_t is constructed in the following manner:

$$r_t = E_t \tag{2}$$

In (2), E_t encourages the robot to increase its mechanical energy. Mechanical energy includes gravitational potential energy, so E_t implicitly encourages the robot to reach a higher height.

In the grasp phase, to enable the snake robot to wrap around the horizontal branch, the reward function r_t is constructed in the following manner:

$$r_t = \sum_{i=0}^{n} (q_i - q_{di})^2 \tag{3}$$

In (3), q_i is the joint angle of the i-th joint, q_{di} is the desired joint angle of the i-th joint. r_t encourages the snake robot to reach the desired joint angles to wrap around the horizontal branch.

d) Initial State: For the swing phase, the initial state is set to a small random range near $\mathbf{0}$. For the grasp phase, we first sample k episodes with policy π_{1d} and find state-action pairs satisfying the function:

$$h_{com} > h_t + \delta_1, h_{end} > h_t + \delta_2 \tag{4}$$

Where h_{com} is the height of the center of figure of the swinging part of the snake robot, h_{end} is the height of the center of figure of the last joint module, h_t is the target height, generally equal to the height of the target horizontal branch. δ_1, δ_2 are positive constants, which are set tolerances. We collect state-action pairs satisfying this function.

After find the initial state-action pairs, we use the Laplace distribution to fit the data and sample from it to obtain the initial states and actions.

2) Training in a Delayed Environment: In the real world, due to the existence of computation delay and communication delay, the policy π_1, π_2 cannot calculate the action $\boldsymbol{a_t}$ and direction $\boldsymbol{d_t}$ through the robot state $\boldsymbol{s_t}$ at the time step t. There is one time step delay in the input of the policy. Therefore, we can only use robot state and action before the time step t to calculate the joint torques $\boldsymbol{\tau_t}$.

Liotet proposed a method named Delayed Imitation with Dagger (DIDA) to train an agent in a delayed environment [30]. Inspired by this method, DDIDA is proposed for our framework. In DDIDA, instead of training one policy network for joint torques $\boldsymbol{\tau_t}$, we train two policy networks for the action $\boldsymbol{a_t}$ and direction $\boldsymbol{d_t}$. The size policy network imitates the magnitude of joint torques in the no-delay environment. The direction policy network imitates the direction of joint torques in the no-delay environment. Combining the output of these policies, we get joint torques $\boldsymbol{\tau_t}$. Training two policy networks is more suitable for our control law, helps imitate the policy π_1, π_2 better and ultimately performs better in the delayed environment. The training process of DDIDA is shown in Fig. 1.

For the initial states and actions, the setting is the same as Section II. At the current time step t, the robot state $\boldsymbol{s_t}$ and joint torques $\boldsymbol{\tau_t}$ are sent to the policy π_{1d}, π_{2d} to generate action $\boldsymbol{a_{t+1}}$ and direction $\boldsymbol{d_{t+1}}$ at the next time step $t+1$. The joint torques $\boldsymbol{\tau_t}$ is calculated at the time step $t-1$, and is sent to the snake robot. Then the robot transfers to a new state $\boldsymbol{s_{t+1}}$. The policy π_1 or π_2 takes the robot state $\boldsymbol{s_{t+1}}$ as input and then outputs the desired action $\boldsymbol{a_{d,t+1}}$ and torque direction $\boldsymbol{d_{d,t+1}}$. Our goal is to make the action $\boldsymbol{a_{t+1}}$ and direction $\boldsymbol{d_{t+1}}$ close to the desired action $\boldsymbol{a_{d,t+1}}$ and torque direction $\boldsymbol{d_{d,t+1}}$, so as to make the snake robot imitate the motion in the no-delay environment. The loss function L is constructed in the following manner:

$$\begin{aligned} l_1 &= \sum_{i=0}^{N} \sum_{j=0}^{n} (a_j - a_{dj})^2 \\ l_2 &= \sum_{i=0}^{N} \sum_{j=0}^{n} (d_j - d_{dj})^2 \\ L &= l_1 + l_2 \end{aligned} \tag{5}$$

Where N is batch size, n is the number of movable joints, a_j is the j-th dimension of the output of π_1, d_j is the j-th dimension of the output of π_2, a_{dj} is the j-th dimension of desired action $\boldsymbol{a_d}$, d_{dj} is the j-th dimension of desired direction $\boldsymbol{d_d}$.

During the training process, the DDIDA algorithm continuously iterates through this cycle and optimizes the policy π_{1d} and π_{2d} to minimize the loss L.

TABLE I
CALIBRATED PARAMETERS

number	odd joint	even joint
1	0.26	0.26
2	(6.3-04, 1.4e-03, 0)	(7.7e-03, 3.9e-04, 0)
3	(1.1e-06, 1.1e-06, 1.1e-06)	(1.1e-06, 1.1e-06, 1.1e-06)
4	(-1.56, 1.56, -3.14)	(-1.56, 1.56, -3.14)
5	0.1	0.1
6	0.2	0.2
7	0.2	0.12

3) Deployment in the Real World: The policy π_{1d}, π_{2d} and switching controller are deployed in the real world. The switching controller is constructed by formula (4). If the state of the snake robot satisfies formula (4), switch to the grasp phase. For this switching controller, we focus on the height of the center of figure of the swinging part of the snake robot and the height of the center of figure of the last joint module. Therefore, it can be easily generalized to snake robots with more segments.

At the initial time step t_0, the snake robot returns the robot state $\boldsymbol{s_0}$, the initial torque $\boldsymbol{\tau_0}$ is set to $\mathbf{0}$. The policy π_{1d} or π_{2d} takes $\boldsymbol{s_0}$ and $\boldsymbol{\tau_0}$ as input and then outputs the action $\boldsymbol{a_1}$ and torque direction $\boldsymbol{d_1}$. Combining $\boldsymbol{a_1}$ and d_1 we get joint torque $\boldsymbol{\tau_1}$. The joint torque $\boldsymbol{\tau_0}$ is sent to the snake robot and robot transfers to a new state $\boldsymbol{s_1}$. Our algorithm continuously iterates through this cycle and enable the snake robot to wrap around the horizontal branch with maneuverability.

C. System Identification

During the training process, the policies try to learn the best decision from the simulation system. However, the mismatch between simulation and reality prevents the successful application of the learned policy. Therefore, system identification is required to estimate the parameter of the snake robot. This helps bridge the reality gap and prevents policy transfer failures due to the significant difference between the simulation and the practical system.

Considering the base part of the snake robot is fixed, the swinging part of the snake robot is like a robot arm. Therefore, the swing part can perform system identification similar to a robotic arm. The system identification problem of a robot arm is a classical problem, for which researchers have proposed a wide range of feasible solutions. In 2017, Wensing proposed a method for parameter identification with constraints [31]. This method takes constraints into account to keep the physical feasibility of the parameters. Therefore, we apply this method to our robot.

Since our snake robot is composed of continuously repeating orthogonal joints, we set the parameters of all odd joints to be the same, and the parameters of all even joints to be the same as well. In this way, when generalizing our method to a snake robot with more joints, there is no need for re-identification. It is only necessary to add the required odd and even joints to the simulation model.

III. EXPERIMENT

In this section, the system identification is performed. All the calibrated parameters are provided in Table I, where parameters from top to down represent: link mass; center of mass coordinates in the link coordinate system, which is defined by the DH method; principal moment of inertia; the XYZ fixed angles from the link coordinate system to the principal axis system; armature inertia; damping coefficient; Coulomb friction coefficient which defines a constant friction force opposite to the joint velocity. In Table I, all parameters are in SI units. The identification results are shown in Fig. 5. Due to the low control frequency, the sampled joint velocities and accelerations are not accurate, resulting in significant identification errors. However, our method can still achieve the maneuvering winding gait, demonstrating its robustness. Besides, to make the base part of the snake robot fixed like the flange of a robot arm, we design a fixture as shown in Fig. 6(a).

Then, the DSG algorithm is first trained in the simulation to show its superior task performance. Subsequently, several physical experiments are conducted on a snake robot to verify the low torque requirement, high robustness under high delay, and fast motion speed of the proposed algorithm. The physical parameter of the snake robot is shown in Table II. The simulation scenario is shown in Fig. 6(b)–6(c). The physical experiment scenario is shown in Fig. 6(d)–6(e). We first compared the performance of our method with other methods in scenario

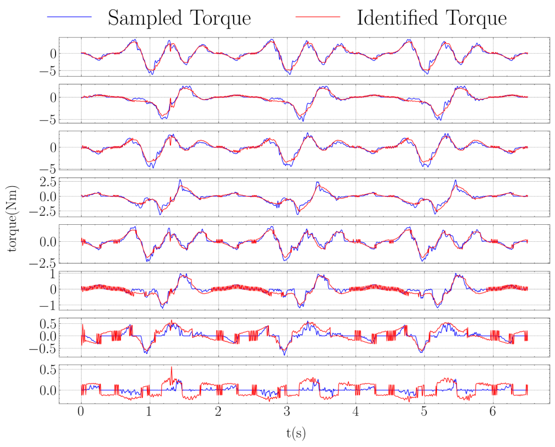


Fig. 5. The identification results for joint 1 to 8, which are shown from top to bottom. Red lines represent the identified torque, and blue lines represent the sampled torque.

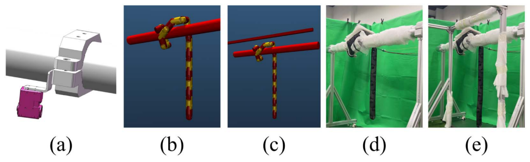


Fig. 6. The fixture and experiment scenario. (a) The fixture. The first joint module of the snake robot is fixed to the fixture using screws. (b)-(c) Simulation scenario. (d)-(e) Physical experiment scenario.

TABLE II
PHYSICAL PARAMETERS OF THE SNAKE ROBOT

Joint Number	8
Link Mass(kg)	0.258
Link Radius(m)	0.33
Link Length(m)	0.98
Max Torque(Nm)	[-2, 2]
Joint Angle(deg)	[-90, 90]
Control Frequency(Hz)	10
α(deg)	0, 5, 10

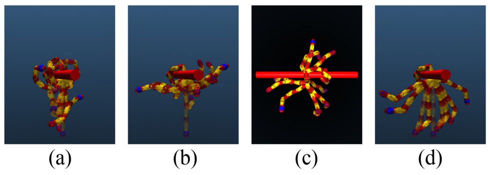


Fig. 7. The optimized motion trajectories of three methods in the swing phase. The transparent sections indicate the motion trajectories of the snake robot. The blue spheres represent the centroid of the last snake robot's joint module. The opaque sections indicate the initial state of the snake robot.(a) Optimized trajectory by TD-MPC + our rule. (b) The left view of optimized trajectory by original TD-MPC. (c) The top view of optimized trajectory by original TD-MPC. (d) Optimized trajectory by TD-MPC + Li's rule.

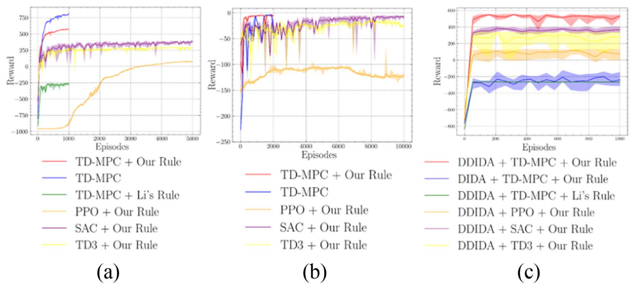


Fig. 8. The training results in simulation. (a) The training results of the swing phase in no-delay environment. (b) The training results of the grasp phase in no-delay environment. (c) The training results of the swing phase in delayed environment.

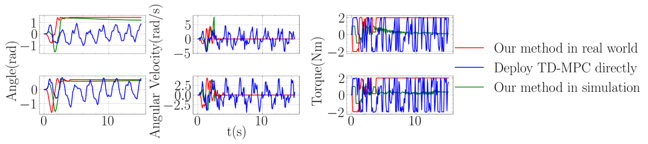


Fig. 9. The joint angle, velocity and torque of trajectories generated by the maneuverable winding gait. From top to bottom are joints 1 to 2. Joints 3 to 8 display similar behavior. The green line represents the trajectory in a delayed simulation environment. The red line represents the trajectory of Fig. 10(a). The blue line represents the trajectory of directly deploying TD-MPC without considering delay.

Fig. 6(b) and validated its generalization in scenario Fig. 6(d). Finally, we validated that our method can be extended to snake robots with more joints in scenario Fig. 6(d) and 6(e).

A. Simulation Results and Analysis

For a snake robot, the training for the maneuvering winding gait is performed in the MuJoCo simulator. In simulation, the identification parameters are applied to the joint modules of the snake robot and we set $\alpha = 10^\circ$. The remaining parameters are the same as those in Table II. The episode length limit of the swing phase is 100 steps. The episode length limit of the grasp phase is 50 steps. We evaluate the significance of each component in the proposed DSG and highlight its advantages over other methods. All algorithms are trained concurrently on a standard computer equipped with a 64-core CPU and an NVIDIA 4060 GPU. Here all simulation results here are based on Fig. 6(b).

To validate the effectiveness of the proposed joint torque direction determination rule, we compare our method with several methods. We first compare our method with original TD-MPC and Li's rule. The optimized motion trajectories of three methods in the swing phase are shown in Fig. 7. The training results of the swing phase in a no-delay environment are shown in Fig. 8(a). For Li's method, we set the joint number of the self-locking segment to 0. Compared to Li's rule, our method retains more exploration space with high rewards, therefore yields higher rewards, higher mechanical energy of the system, and higher swinging heights. Besides, our method does not require self-locking joint, which is often difficult to achieve the self-locking effect through the joint's torque alone and requires additional mechanical structures. The original TD-MPC method achieves higher rewards, but it makes the snake robot move like a fan, which is shown in Fig. 7(b)–7(c). This policy results in higher rewards and higher mechanical energy, but the swinging height is insufficient, making further grasping impossible. Compared to original TD-MPC method, our method restricts the movement pattern of the snake robot and ensures it swings like a swing even without any training. Moreover, our method significantly increases the convergence speed of the training process and ensures the snake robot increases its mechanical energy and achieves further winding.

At the same time, we compared TD-MPC + our rule with PPO [32] + our rule, SAC [33] + our rule and TD3 [34] + our rule. As shown in Fig. 8(a), TD-MPC + our rule converges faster and achieves a higher discounted cumulative reward.

The training results of the grasp phase in a no-delay environment are shown in Fig. 8(b). Compared to original TD-MPC method, while achieving similar rewards, our method reduces the exploration space and results in faster convergence and more stable training. Compared to SAC + our rule, TD3 + our rule and PPO + our rule, our method converges faster and achieves better final performance. In the grasp phase, we did not compare with Li's rule, because it considers two-dimensional motion.

To validate the effectiveness of the proposed DDIDA method, we compare our method(DDIDA + TD-MPC + our rule) with DIDA + TD-MPC + our rule. The training results of the swing phase in a delayed environment are shown in Fig. 8(c). Our method uses one network to output the magnitude of joint torques and another network to output the directions of joint torques, decomposing the problem into a regression problem and a classification problem. This approach is more suitable for our proposed joint torque direction determination rule and ultimately achieves rewards similar to those in a no-delay environment, significantly higher than the DIDA method. At the same time, we compared DDIDA + TD-MPC + Li's rule, DDIDA + PPO/SAC/TD3 + our rule, our method achieves better final performance.

The joint angle, velocity and torque of the trajectory generated by our method in a delayed environment is shown in Fig. 9. In the first 2.5 seconds, the snake robot accumulates mechanical energy through swinging. When reaching the switching controller's judgment condition, it switches to the grasping policy. Finally, under a delay of 0.1 s and a maximum torque of 2 Nm, the snake robot spends 5 s wrapping around the horizontal branch with maneuverability. It demonstrates that the proposed RL controller has low torque requirement, high robustness under high delay, and fast motion speed.

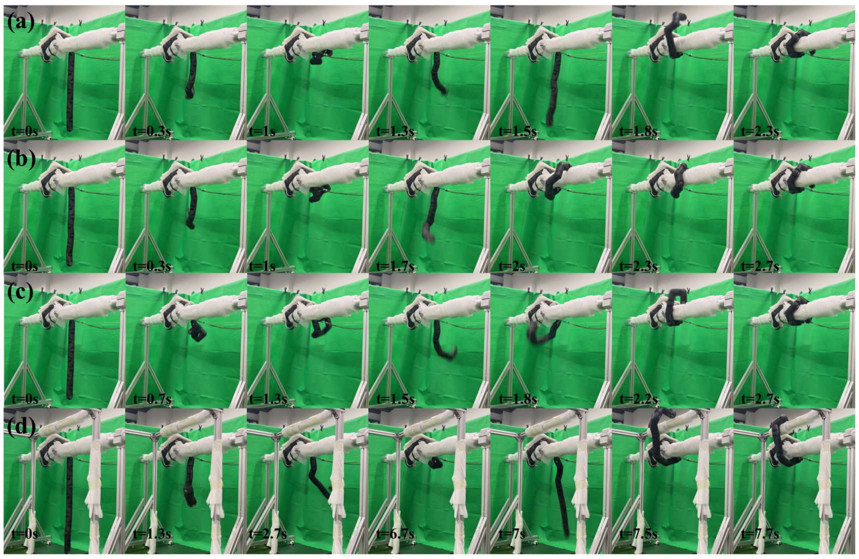


Fig. 10. The complete motion process of the maneuverable winding gait in the real world. (a) alpha=10°, joint angles are all 0. (b) alpha=10°, non-zero initial joint angles. (c) alpha=0°, joint angles are all 0. (d) alpha=10°, joint angles are all 0. The snake robot grasps a higher horizontal branch beside it.

B. Physical Experiments Results and Analysis

To validate the effectiveness of our method, we conducted physical experiments. The complete motion process of the maneuverable winding gait in the real world is shown in Fig. 10. The joint angle, velocity and torque of the trajectory in Fig. 10(a) in are shown in Fig. 9.

Fig. 10(a)–10(b) shows the trajectories of the maneuverable winding gait in different initial states. Since the dynamics model of the system does not change when β does not change and the grasp policy can adapt to small variations in α, We deploy our policy directly to $\alpha = 0°$. Fig. 10(c) shows the trajectories of the maneuverable winding gait in $\alpha = 0°$. It demonstrates our method's generalizablity for different initial states and different α.

The red line in Fig. 9 shows the joint angle, velocity and torque of the trajectory in Fig. 10(a). Compared to the green line in Fig. 9, The torque of the snake robot stops changing after wrapping, as we stop sending new torque commands when the error with the target joint angle is below a certain threshold. The blue line in Fig. 9 shows if we deploy TD-MPC without considering delay, the torque direction and the angular velocity of the first joint fail to meet our torque direction determination rule, leading to excessive shaking, insufficient swinging height, and wrapping failure.

We extend the snake robot to ten joints by adding an odd and an even joint in the simulation without extra identification. The agent trained in simulation was deployed successfully in the real world, where the robot wrapped around a horizontal bar with a 0.2 m horizontal distance and 0.15 m vertical distance. The trajectory is shown in Fig. 10(d).

All in all, In Fig. 10(a)–10(c), under a high delay of 0.1 s and a low maximum torque of 2 Nm, the snake robot uses just 2.3–2.7 s to wrap around the target horizontal branch with maneuverability. It demonstrates that the proposed DSG has low torque requirement, high robustness under high delay, fast motion speed. For comparison, the total mass of the swing part of the snake robot is 2.06 kg. When the first joint's lever arm exceeds 0.1 m, the torque saturation problem occurs. Traditional static planning methods cannot solve this problem. Our method leverages the conversion between kinetic and potential energy and overcomes the torque saturation problem. In Fig. 10(b)–10(c), the snake robot wraps around the horizontal branch with different initial angles and different α. In Fig. 10(d), the snake robot uses just 7.7 s to wrap around a higher horizontal branch beside it with maneuverability. It demonstrates that the proposed DSG has high generalizability.

IV. Conclusion

In this letter, a novel delay-aware swing and grasp framework DSG is proposed, based on which a maneuverable winding gait is designed to enable the snake robot to wrap around the horizontal branch with maneuverability. Specifically, first, to overcome the torque saturation problem, a joint torque direction determination rule is proposed to fully utilize the kinetic energy of the snake robot and enable the robot to move in the desired manner. Then, the framework DSG reduces the exploration space of the policy and guarantees the performance under delay, and based on which a maneuverable winding gait is designed to enable it to wrap around the target horizontal branch with maneuverability. Simulation and sufficient experiment results demonstrate that the proposed RL controller has low torque requirement, high robustness under high delay, fast motion speed, and high generalizability.

In the future, We will focus on achieving maneuverable wrapping gaits without the fixture and at different β to enhance the generalization ability of the algorithm. Besides, we want to model the swing phase and grasp phase as a multi-agent cooperation problem rather than training them separately. Furthermore, we will use DSG to achieve movement between branches. These will extend the inter-tree motion capability of snake robots and thereby expanding the application scenarios of snake robots.

REFERENCES

[1] B. C. Jayne and J. D. Davis, "Kinematics and performance capacity for the concertina locomotion of a snake (coluber constrictor)," *J. Exp. Biol.*, vol. 156, no. 1, pp. 539–556, 1991.

[2] M. Tesch et al., "Parameterized and scripted gaits for modular snake robots," *Adv. Robot.*, vol. 23, no. 9, pp. 1131–1158, 2009.

[3] T. Takemori, M. Tanaka, and F. Matsuno, "Gait design for a snake robot by connecting curve segments and experimental demonstration," *IEEE Trans. Robot.*, vol. 34, no. 5, pp. 1384–1391, Oct. 2018.

[4] D. Rollinson and H. Choset, "Pipe network locomotion with a snake robot," *J. Field Robot.*, vol. 33, no. 3, pp. 322–336, 2016.

[5] W. Qi, T. Kamegawa, and A. Gofuku, "Helical wave propagation motion for a snake robot on a vertical pipe containing a branch," *Artif. Life Robot.*, vol. 23, pp. 515–522, 2018.

[6] W. Huang, X. Guo, H. Liu, and Y. Fang, "A robust model-based radius estimation approach for helical climbing motion of snake robots," *IEEE/ASME Trans. Mechatron.*, vol. 28, no. 6, pp. 3284–3293, Dec. 2023.

[7] M. W. Spong, "The swing up control problem for the acrobot," *IEEE control Syst. Mag.*, vol. 15, no. 1, pp. 49–55, Feb. 1995.

[8] M. W. Spong, "Energy based control of a class of underactuated mechanical systems," *IFAC Proc. Volumes*, vol. 29, no. 1, pp. 2828–2832, 1996.

[9] I. Fantoni, R. Lozano, and M. W. Spong, "Energy based control of the pendubot," *IEEE Trans. Autom. Control*, vol. 45, no. 4, pp. 725–729, Apr. 2000.

[10] O. Kolesnichenko and A. S. Shiriaev, "Partial stabilization of underactuated Euler–Lagrange systems via a class of feedback transformations," *Syst. Control Lett.*, vol. 45, no. 2, pp. 121–132, 2002.

[11] X. Xin and M. Kaneda, "Analysis of the energy-based swing-up control of the acrobot," *Int. J. Robust Nonlinear Control, IFAC-Affiliated J.*, vol. 17, no. 16, pp. 1503–1524, 2007.

[12] X. Xin, J. She, and Y. Liu, "A unified solution to swing-up control for n-link planar robot with single passive joint based on virtual composite links and passivity," *Nonlinear Dyn.*, vol. 67, pp. 909–923, 2012.

[13] A. Zhang, J. She, X. Lai, and M. Wu, "Motion planning and tracking control for an acrobot based on a rewinding approach," *Automatica*, vol. 49, no. 1, pp. 278–284, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109812005146

[14] A. Majumdar, A. A. Ahmadi, and R. Tedrake, "Control design along trajectories with sums of squares programming," in *Proc. 2013 IEEE Int. Conf. Robot. Automat.*, 2013, pp. 4054–4061.

[15] A. Zhang, X. Lai, M. Wu, and J. She, "Stabilization of underactuated two-link gymnast robot by using trajectory tracking strategy," *Appl. Math. Computation*, vol. 253, pp. 193–204, 2015.

[16] M. Lee and M. Smith, "Automatic design and tuning of a fuzzy system for controlling the acrobot using genetic algorithms, DSFS, and meta-rule techniques," in *Proc. NAFIPS/IFIS/NASA '94. Proc. 1st Int. Joint Conf. North Amer. Fuzzy Inf. Process. Soc. Biannual Conf. Ind. Fuzzy Control Intell.*, 1994, pp. 416–420.

[17] M. Smith, T. Zhang, and W. Gruver, "Dynamic fuzzy control and system stability for the acrobot," in *Proc. 1998 IEEE Int. Conf. Fuzzy Syst. Proc. IEEE World Congr. Comput. Intell.*, 1998, vol. 1, pp. 286–291.

[18] X. Lai, J. She, Y. Ohyama, and Z. Cai, "Fuzzy control strategy for acrobots combining model-free and model-based control," 1999. [Online]. Available: https://api.semanticscholar.org/CorpusID

[19] X. Lai, Z. Cai, M. Wu, and J. She, "A fuzzy control strategy for acrobot [j]," *Control Theory Appl.*, vol. 17, no. 3, pp. 326–330, 2000.

[20] C. Jun-Qing, L. Xu-Zhi, and W. Min, "Position control method for a planar acrobot based on fuzzy control," in *Proc. 34th Chin. Control Conf.*, 2015, pp. 923–927.

[21] G. DeJong and M. Spong, "Swinging up the acrobot: An example of intelligent control," in *Proc. 1994 Amer. Control Conf. - ACC '94*, 1994, vol. 2, pp. 2158–2162.

[22] S. C. Brown and K. M. Passino, "Intelligent control for an acrobot," *J. Intell. Robotic Syst.*, vol. 18, pp. 209–248, 1997.

[23] G. Boone, "Efficient reinforcement learning: Model-based acrobot control," in *Proc. Int. Conf. Robot. Automat.*, 1997, vol. 1, pp. 229–234.

[24] S. C. Duong, E. Uezato, H. Kinjo, and T. Yamamoto, "Intelligent control strategies for the acrobot using neurocontroller optimized by genetic algorithm," *SICE J. Control, Meas., System Integration*, vol. 2, no. 5, pp. 317–324, 2009.

[25] J. Nakanishi, T. Fukuda, and D. E. Koditschek, "A brachiating robot controller," *IEEE Trans. Robot. Automat.*, vol. 16, no. 2, pp. 109–123, Apr. 2000.

[26] H. Kajima, M. Doi, Y. Hasegawa, and T. Fukuda, "Energy based swing control of a brachiating robot," in *Proc. 2005 IEEE Int. Conf. Robot. Automat.*, 2005, pp. 3670–3675.

[27] W. Wu, M. Huang, and X. Gu, "Underactuated control of a bionic-ape robot based on the energy pumping method and big damping condition turn-back angle feedback," *Robot. Auton. Syst.*, vol. 100, pp. 119–131, 2018.

[28] Z. Li, C. Tang, G. Zhou, X. Guo, J. Lu, and X. Shu, "Discontinuous rod spanning motion control for snake robot based on reinforcement learning," in *Proc. 2023 42nd Chin. Control Conf.*, 2023, pp. 4437–4442.

[29] N. A. Hansen and X. Wang, "Temporal difference learning for model predictive control," in *Proc. 39th Int. Conf. Mach. Learn.*, Jul. 2022, vol. 162, pp. 8387–8406. [Online]. Available: https://proceedings.mlr.press/v162/hansen22a.html

[30] P. Liotet, D. Maran, L. Bisi, and M. Restelli, "Delayed reinforcement learning by imitation," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 13528–13556.

[31] P. M. Wensing, S. Kim, and J.-J. E. Slotine, "Linear matrix inequalities for physically consistent inertial parameter identification: A statistical perspective on the mass distribution," *IEEE Robot. Automat. Lett.*, vol. 3, no. 1, pp. 60–67, Jan. 2018.

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

[33] T. Haarnoja et al., "Soft actor-critic algorithms and applications," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1861–1870.

[34] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1587–1596.