Morphology-Aware Graph Reinforcement Learning for Tensegrity Robot Locomotion

Chi Zhang, Mingrui Li, Wenzhe Tong, and Xiaonan Huang

***Abstract*— Tensegrity robots combine rigid rods and elastic cables, offering high resilience and deployability but posing major challenges for locomotion control due to their underactuated and highly coupled dynamics. This paper introduces a morphology-aware reinforcement learning framework that integrates a graph neural network (GNN) into the Soft Actor-Critic (SAC) algorithm. By representing the robot's physical topology as a graph, the proposed GNN-based policy captures coupling among components, enabling faster and more stable learning than conventional multilayer perceptron (MLP) policies. The method is validated on a physical 3-bar tensegrity robot across three locomotion primitives, including straight-line tracking and bidirectional turning. It shows superior sample efficiency, robustness to noise and stiffness variations, and improved trajectory accuracy. Notably, the learned policies transfer directly from simulation to hardware without fine-tuning, achieving stable real-world locomotion. These results demonstrate the advantages of incorporating structural priors into reinforcement learning for tensegrity robot control.**

I. Introduction

Tensegrity robots, composed of rigid rods and elastic cables in prestressed equilibrium, have attracted increasing interest for their unique structural advantages [1]–[3]. Their intrinsic deformability provides exceptional impact resistance, enabling survival after high drops, and allows the robot to shrink into compact forms for efficient transport and deployment [4]. These features make tensegrity robots promising for operation in challenging environments, ranging from disaster response to planetary exploration. However, the very properties that give tensegrity robots their advantages also complicate locomotion control.

Locomotion in tensegrity systems is challenging because of their floating-base configuration, underactuated cable actuation, and distributed tension network, which together yield highly non-linear and globally coupled dynamics. Local actuation propagates throughout the structure, producing complex global deformations. Such characteristics make accurate modeling difficult and limit the effectiveness of traditional control strategies.

Recent advances in deep reinforcement learning offer a potential solution, as model-free algorithms can learn policies directly from interaction without requiring explicit dynamics models [5]. In particular, the Soft Actor-Critic (SAC) algorithm [6] has demonstrated strong performance in continuous, high-dimensional robotic control. However, most existing approaches adopt generic multilayer perceptrons (MLPs) as policy networks, treating the robot state as an unstructured vector of observations. This design neglects the robot's morphology and intrinsic coupling, often leading to slow convergence and suboptimal final performance in tensegrity locomotion. To address this limitation, we seek to incorporate the robot's structural priors directly into the policy representation.

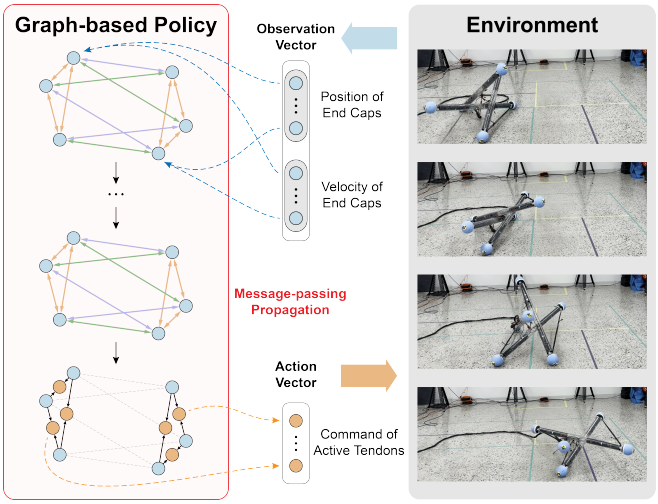

Fig. 1. Morphology-aware graph reinforcement learning for tensegrity locomotion. The robot's states (end-cap positions and velocities) are encoded as node features in a graph-based policy, which propagates information along the robot's structural connections. The network outputs tendon length commands to actuate the tensegrity robot to roll forward in physical experiments.

In this work, we present a morphology-aware reinforcement learning framework tailored for tensegrity robots. Our approach integrates a graph-based policy architecture into SAC, where nodes represent rod end-caps and edges represent mechanical connections via cables and rods. By explicitly encoding the robot's physical structure, the policy can capture the coupling between components, leading to more efficient and effective learning.

We evaluated the proposed framework on fundamental locomotion primitives, including straight-line tracking and in-place turning, which serve as building blocks for trajectory following. Our results demonstrate that GNN-based policies achieve higher task performance, and require fewer samples than standard MLP-based baselines. Moreover, the learned policies enable zero-shot transfer from simulation to the 3-bar tensegrity robot, resulting in robust real-world locomotion.

The contributions of this work are summarized as follows:

- We propose a morphology-aware policy architecture by designing a GNN-based actor that encodes the physical topology of tensegrity robots and captures the intrinsic coupling between components.

- By integrating the proposed GNN actor into the SAC framework, we demonstrate substantial gains in both training efficiency and final task performance compared with conventional MLP baselines.
- We validate our approach on hardware by learning locomotion primitives, straight-line tracking and clockwise/counterclockwise turning, that transfer directly from simulation to physical robot without fine-tuning, demonstrating effective sim-to-real generalization.

II. Related Work

Tensegrity robots have been studied for their unique advantages in impact resistance and deployability, with prototypes explored for applications ranging from terrestrial mobility to space exploration. Locomotion control, however, remains challenging due to their floating-base, underactuated, and globally coupled dynamics.

Early approaches to tensegrity locomotion relied on open-loop controllers, where manually designed actuation sequences generated rolling gaits [7]–[10]. While these studies demonstrated feasibility, their lack of feedback rendered them fragile in uncertain environments. Model predictive control (MPC) subsequently improved robustness by incorporating system dynamics [11], [12], and hybrid MPC-learning schemes further enhanced efficiency [13]. However, the difficulty of accurately modeling globally coupled tensegrity dynamics and the high computational cost limited the scalability of such approaches.

These limitations motivated a shift toward data-driven methods. Reinforcement learning (RL) has achieved remarkable success across a wide range of robotic domains, with algorithms such as DDPG [14], PPO [15], SAC [6] and TD3 [16] enabling robust policies for high-dimensional continuous control. RL has powered advances in manipulation [17], legged locomotion [18], and aerial navigation [19], and sim-to-real studies have shown that policies trained in simulation can often be deployed to hardware with minimal fine-tuning [20]. These achievements highlight RL as a promising paradigm for tensegrity locomotion, where conventional model-based control struggles with complexity and modeling inaccuracies.

RL has since demonstrated the ability to produce transferable locomotion policies for tensegrity robots, even under partial observability or on the rough terrain [21]–[24]. These results confirm that RL can produce robust and versatile behaviors for tensegrity robots. However, existing approaches almost universally adopt generic MLPs to represent policies, flattening the robot's complex morphology into an unstructured input vector. This design neglects the physical coupling intrinsic to tensegrity systems and often results in slow training convergence and suboptimal final performance.

Recent studies suggest that embedding morphology or structural priors into learning architectures can substantially improve efficiency and generalization [25], [26]. In the context of tensegrity robots, morphology has been incorporated into graph-based models, for instance through graph neural networks (GNNs) used for differentiable dynamics learning [27], [28]. However, these efforts focus primarily on system identification and forward simulation, rather than policy learning for locomotion. To bridge this gap, our work introduces a morphology-aware reinforcement learning framework that integrates GNNs into SAC, enabling policies that explicitly encode the robot's physical topology.

III. Preliminaries

A. 3-bar Tensegrity Robot Model

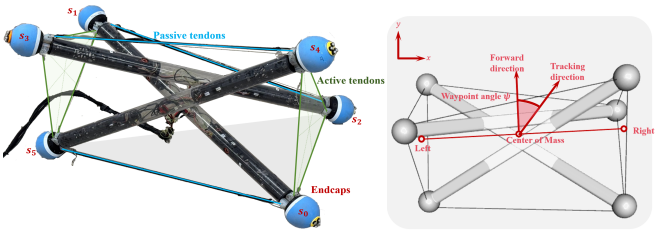

Fig. 2. Physical 3-bar tensegrity robot platform and reference coordinate definitions.

The 3-bar tensegrity robot consists of three rigid rods connected by a network of elastic tendons, forming a twisted triangular prism with two triangular faces defined as the left and right sides (Fig. 2). The line connecting the centroids of these faces defines the robot's lateral axis, while the forward locomotion direction lies perpendicular to this axis, providing a consistent reference for orientation and motion evaluation.

Each rod terminates in two end-caps, which act as structural connection points. The tendons are categorized as:

- Cross tendons (passive): connect end-caps in the same side, maintaining the global stability of the structure.
- Side tendons (active): connect end-caps left and right sides, and can be actuated to change length, enabling deformation and rolling motion.

The coordinated actuation of side tendons drives locomotion, while cross tendons preserve the tensegrity's geometry. Owing to this modular design, the robot's topology can naturally be represented as a graph: end-caps correspond to nodes, and tendons or rods define edges. This abstraction forms the foundation of the graph-based policy architecture.

B. Soft Actor-Critic Background

We formulate the tensegrity locomotion control as a Markov Decision Process (MDP) defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, where states $s_t \in \mathcal{S}$ represent the endcap positions and velocities, while actions $a_t \in \mathcal{A}$ correspond to active tendon length commands.

The Soft Actor-Critic (SAC) algorithm [6] optimizes an entropy-regularized objective to balance reward maximization and exploration:

$$J(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t\left(r(s_t,a_t) + \alpha\mathcal{H}(\pi(\cdot|s_t))\right)\right] \quad (1)$$

where α controls the trade-off between entropy and task reward.

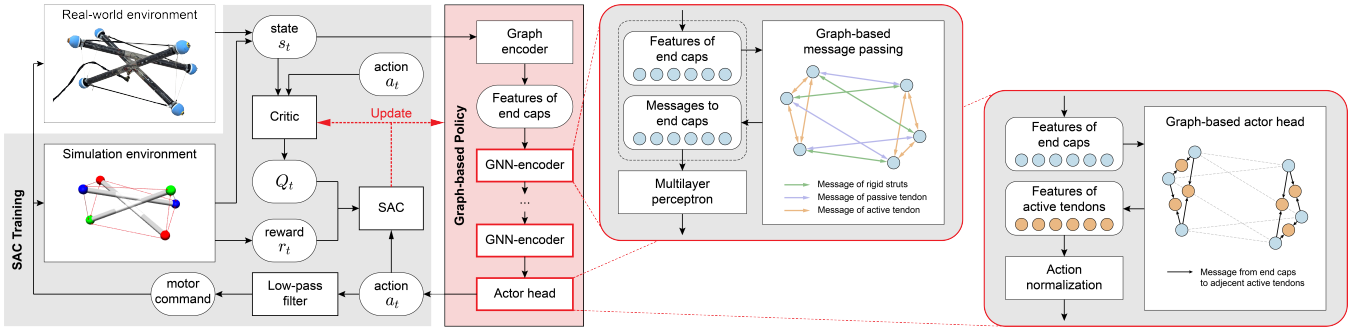

Fig. 3. Overview of the proposed morphology-aware GNN-SAC framework for tensegrity robot locomotion. The Soft Actor-Critic (SAC) algorithm integrates a graph neural network (GNN)-based policy that encodes the robot's topology via message passing among end-cap nodes. The actor generates tendon length commands based on structured observations, enabling morphology-aware learning in both simulation and real-world environments.

The actor is trained to minimize loss $\mathcal{L}_\pi$:

$$\mathcal{L}_\pi = \mathbb{E}_{s_t \sim \mathcal{D}} \left[\alpha \log \pi \left(a_t | s_t \right) - Q(s_t, a_t) \right] \tag{2}$$

while two Q-value critics are updated via soft Bellman backups to minimize temporal-difference loss. The temperature parameter α is automatically tuned to maintain a target policy entropy, ensuring balanced exploration during learning.

This standard SAC framework provides a sample-efficient and robust foundation for continuous tensegrity control. In the next section, we extend SAC by introducing a graph neural network (GNN)-based actor that explicitly encodes the robot's physical topology and coupling relationships, forming the core of our morphology-aware reinforcement learning approach.

IV. Methodology

A. Graph Construction

Tensegrity robots exhibit highly coupled dynamics due to their tension-based equilibrium: a local actuation propagates through the tension network and induces global structural deformation. To effectively capture this distributed behavior, we represent the 3-bar tensegrity robot as a directed graph that mirrors its physical topology.

The constructed graph, denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, contains six vertices $\mathcal{V}$ and 24 directed edges $\mathcal{E}$. Each vertex represents a corresponding rod end-cap V_i, and each physical connection (rod or tendon) is modeled as a pair of directed edges $(E_{i,j}, E_{j,i})$, enabling bidirectional message passing between vertices. The resulting graph, therefore, consists of three types of edges:

- Rigid rods (6 directed edges),
- Passive tendons (6 directed edges),
- Active tendons (12 directed edges).

Each edge encodes the physical relationship between two end caps, while vertices carry local state information. This representation allows the policy network to leverage message passing over the morphology-aware graph structure, rather than treating the robot state as an unstructured flat vector.

B. GNN-based Soft Actor-Critic

We integrate the above tensegrity graph representation into the actor network of Soft Actor-Critic (SAC). This design enables the policy to explicitly capture spatial coupling among robot components through relational message passing.

1) Observation encoding: At each timestep, the raw robot state is encoded into the vertex and edge features of the tensegrity graph. Vertex V_i features include the 3D position p_i and velocity v_i of the corresponding rod end-cap, augmented with task-related parameters (e.g., global motion commands for target position) that are broadcast to all vertices. Edge features encode the relative distance between the connected vertices, together with a categorical indicator of edge type (rigid rod, passive tendon, or active tendon). These structured features serve as the input to the GNN encoder, enabling the policy to reason over the robot state in a morphology-aware manner.

2) GNN encoder: The encoder consists of multiple message-passing layers. At each layer, a message is generated from vertex V_i to its neighbor V_j as:

$$M(V_i, V_j) = \text{MLP}_m(V_i, V_j, E_{i,j}), \tag{3}$$

where $E_{i,j}$ denotes the feature vector of the edge connecting V_i and V_j. Incoming messages are aggregated at vertex V_i:

$$M(V_i) = \sum_{j \in \mathcal{N}(i)} M(V_i, V_j), \tag{4}$$

with $\mathcal{N}(i)$ denoting the neighbors of V_i. The vertex feature is then updated as:

$$V_i' = \text{MLP}_f(V_i, M(V_i)). \tag{5}$$

After several layers of message passing, the vertices encode rich contextual information about both local states and global coupling effects.

3) Actor head: The final policy output corresponds to the actions applied to active tendons. For each tendon between vertices V_i and V_j, the control command is predicted as:

$$A_{i,j} = \text{MLP}_a(V_i, V_j). \tag{6}$$

where $A_{i,j}$ denotes the length command for the active tendon connecting vertices V_i and V_j.

This formulation naturally aligns the action space with the robot morphology, ensuring that each actuated tendon is directly associated with its corresponding pair of vertices.

4) Policy training with SAC: The GNN-based actor is embedded into the standard Soft Actor-Critic (SAC) framework. At each step, the encoded graph state is processed by the GNN encoders and the actor head to produce tendon length commands, which are filtered and applied as motor targets in the environment. The SAC algorithm then updates the actor and critic networks based on observed rewards and transitions, preserving the benefits of entropy-regularized reinforcement learning while leveraging morphology-aware policy representations.

C. Training Formulation

To enable waypoint-based navigation, we design three motion primitives for the tensegrity robot: (i) straight-line tracking toward a designated target point, (ii) counterclockwise in-place turning, and (iii) clockwise in-place turning. Each primitive is implemented as a reinforcement learning task under the GNN-SAC framework, where the reward combines task progress with control regularization. All tasks are defined in the 2-D ground plane with the robot center of mass (CoM) position denoted as $\boldsymbol{p} \in \mathbb{R}^2$ and yaw angle ψ. For the tracking task, the observation additionally includes a tracking vector $\boldsymbol{v}_{\text{tr}} = \boldsymbol{p}_{target} - \boldsymbol{p}$, while turning tasks use only the generic robot state.

1) Tracking reward: For the tracking task, the robot is required to move from its initial center of mass position $\boldsymbol{p}_0 \in \mathbb{R}^2$ toward a target waypoint $\boldsymbol{p}_{target} \in \mathbb{R}^2$. The desired direction is

$$\boldsymbol{d}_{\text{tr}} = \frac{\boldsymbol{p}_{target} - \boldsymbol{p}_0}{\|\boldsymbol{p}_{target} - \boldsymbol{p}_0\|_2}. \tag{7}$$

Let $\boldsymbol{p}_t$ denote the CoM position at time t. The CoM displacement $\boldsymbol{p}_t - \boldsymbol{p}_0$ is decomposed into the aligned component $d_a(\boldsymbol{p}_t) = \boldsymbol{d}_{\text{tr}}^\top(\boldsymbol{p}_t - \boldsymbol{p}_0)$ and the lateral deviation $d_b(\boldsymbol{p}_t) = \|\boldsymbol{d}_{\text{tr}} \times (\boldsymbol{p}_t - \boldsymbol{p}_0)\|_2$. A potential function encodes both progress and alignment:

$$P_{\text{tr}}(\boldsymbol{p}_t) = \lambda_1 d_a(\boldsymbol{p}_t) \exp\left(-\tfrac{d_b(\boldsymbol{p})^2}{2\sigma_{\text{tr}}^2}\right) - \lambda_2 \|\boldsymbol{p}_t - \boldsymbol{p}_{target}\|_2, \tag{8}$$

The tracking reward is the potential difference over a control step:

$$r_{\text{tr}} = P_{\text{tr}}(\boldsymbol{p}_{t+1}) - P_{\text{tr}}(\boldsymbol{p}_t). \tag{9}$$

2) Turning reward: For in-place turning, the robot is encouraged to change yaw with minimal translation. Let ψ_t denote the yaw angle of the robot at step t and $\boldsymbol{p}_t$ the corresponding CoM position. A potential function penalizes translational deviation from the initial position $\boldsymbol{p}_0$:

$$P_{\text{turn}}(\boldsymbol{p}) = -\lambda_{\text{turn}} \|\boldsymbol{p} - \boldsymbol{p}_0\|_2^2 \tag{10}$$

The desired turning direction is represented as $d_{\text{turn}} \in \{+1, -1\}$, where $+1$ indicates counterclockwise and -1 indicates clockwise rotation. The turning reward is then defined as

$$r_{\text{turn}} = d_{\text{turn}} \cdot (\psi_{t+1} - \psi_t) + \left[P_{\text{turn}}(\boldsymbol{p}_{t+1}) - P_{\text{turn}}(\boldsymbol{p}_t)\right] \tag{11}$$

3) Control cost: For all tasks, tendon actuation is regularized by a quadratic penalty:

$$c(\boldsymbol{a}_t) = \lambda_c \sum_i (a_{t,i} - l_{0,i})^2 \tag{12}$$

where $a_{t,i}$ is the commanded length of tendon i at step t, and $l_{0,i}$ its nominal balanced length.

4) Overall reward: The per-step reward is expressed as

$$r_t = \begin{cases} r_{\text{tr}} - c(\boldsymbol{a}_t), & \text{tracking} \\ r_{\text{turn}} - c(\boldsymbol{a}_t), & \text{turning} \end{cases} \tag{13}$$

These task-specific rewards provide the optimization signal for training the GNN-based SAC policy.

V. Experimental Results

A. Experimental Setup

All simulation experiments were conducted using the MuJoCo physics engine to evaluate the proposed GNN-SAC algorithm. The control loop operated at 50 Hz, and passive tendon elasticity was modeled as linear springs with stiffness $k = 450$ N/m. A low-pass filter (LPF) has been applied to the action commands before it executed in simulation.

The learning algorithm was implemented in PyTorch and executed on an Ubuntu 20.04 laptop equipped with an Intel Core i5-12500H CPU and Nvidia GeForce RTX 3070Ti GPU. The average inference time of the GNN-SAC network was 1.19 ms in a single-threaded process, corresponding to a theoretical maximum control frequency of 840 Hz. And on a Jetson Orin Nano, the average inference time was measured to be 9.61 ms, which is 105Hz. This demonstrates the onboard computing capacities of the proposed algorithm.

For hardware validation, a physical 3-bar tensegrity robot was constructed using polycarbonate tubes connected with both active and passive tendons. Active tendons were fabricated from Dyneema cords, while passive tendons used nonlinear elastic polymer climbing ropes with stiffness ranging from 242 to 643 N/m. Each active tendon was actuated by a Quasi-Direct Drive actuator with GIM4310 brushless DC motor with 10:1 planetary gearbox and a cable spool.

An external optical motion-capture (MoCap) system provided global end-cap positions with 2mm std at 100 Hz to an external Jetson Orin Nano through ROS. Jetson Orin Nano also communicated with the onboard motor controller via daisy-chained CAN bus at 50Hz, enabling the execution of the learned policies during the real-world experiments.

B. Benchmark of learning performance

To assess learning efficiency and overall task performance, we benchmarked four algorithms:

- **G-SAC:** SAC with our proposed graph-based policy
- **M-SAC:** SAC with a standard multilayer perceptron policy
- **PPO:** Proximal Policy Optimization
- **TD3:** Twin Delayed Deep Deterministic Policy Gradient

Each algorithm was trained to learn three fundamental motion primitives: straight-line tracking, counterclockwise in-place turning, and clockwise in-place turning.

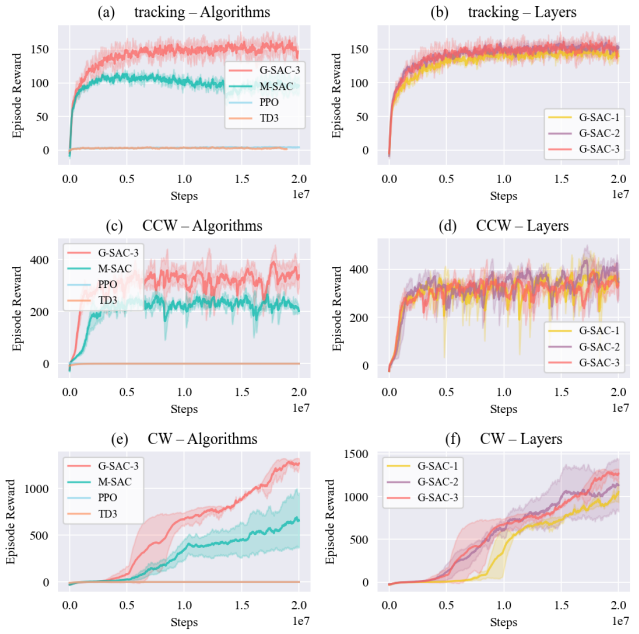

Fig. 4. Benchmark of learning performance across algorithms and network depths. The proposed GNN-SAC consistently outperforms MLP-based SAC (M-SAC), PPO, and TD3 in terms of training reward and sample efficiency for all three locomotion primitives. Subplots (a,c,e) compare algorithms, while (b,d,f) analyze the effect of GNN encoder depth, showing improved performance with multi-layer message passing.

As shown in Fig. 4, G-SAC consistently outperforms the baselines in both reward per training step and reward per wall-clock time across all primitives. The advantage over M-SAC indicates that explicitly encoding the robot's morphology significantly accelerates policy learning, while the performance gap to PPO and TD3 reflects the efficiency of entropy-regularized off-policy actor–critic methods in high-dimensional continuous control.

We further conducted an ablation study on the depth of the GNN encoder, comparing single-layer (**G-SAC-1**), two-layer (**G-SAC-2**), and three-layer (**G-SAC-3**) variants. As shown in Fig. 4, deeper encoders generally yielded better performance: G-SAC-3 achieved slightly higher final rewards than G-SAC-2, while both substantially outperformed G-SAC-1. This trend reflects the benefit of multi-hop message passing, allowing the policy to capture long-range coupling across the tensegrity structure.

Overall, these results establish that incorporating structural priors via GNNs improves both sample efficiency and policy quality, forming a strong baseline for the subsequent evaluations.

C. Motion Primitive Evaluation and Trajectory Composition

After training, the learned policies were evaluated in simulation for all three motion primitives under randomized initial poses to assess stability and consistency.

For straight-line tracking, the robot was commanded to reach multiple waypoints positioned at different orientation angles relative to its initial heading. Each test consisted of repeated trials per waypoint, and performance was quantified by the deviation between the robot's final and target positions. As shown in Fig. 5(a), the G-SAC policy achieved substantially lower deviation in distance than M-SAC across all waypoint directions, demonstrating improved directional accuracy and robustness.

For in-place turning, policies were evaluated by the average yaw rate during counterclockwise (CCW) and clockwise (CW) rotations. G-SAC exhibited faster and more stable rotations compared to M-SAC as shown in Fig. 5(b).

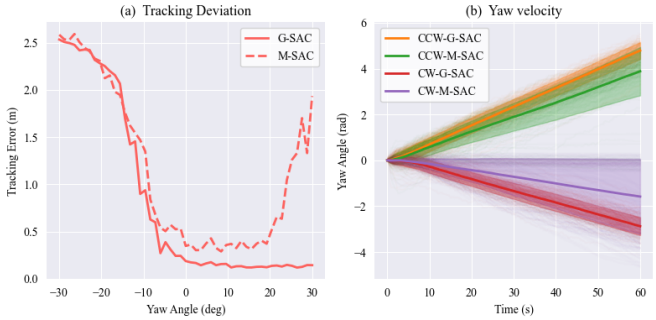

Fig. 5. Simulation evaluation of learned motion primitives between Graph-based SAC (G-SAC) and MLP-based SAC (M-SAC): (a) Straight-line tracking error for different waypoint yaw angles; (b) Yaw rate and stability in bidirectional turning tasks.

To demonstrate motion composability, the three primitives were combined for waypoint-based trajectory following. A high-level planner sequentially selected primitives according to the robot's relative pose to successive targets. As illustrated in Fig. 6, the robot successfully reached a sequence of waypoints, showing that the learned primitives can serve as reliable building blocks for higher-level navigation.

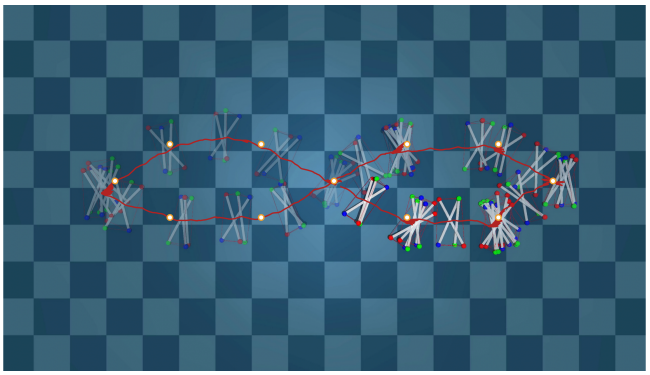

Fig. 6. Composed trajectory tracking using learned motion primitives. The robot follows an infinity-shaped (∞) waypoint sequence by sequentially combining straight-line and turning primitives. The resulting CoM trajectory (red) closely aligns with target waypoints (orange), confirming effective motion composition and trajectory accuracy.

Together, these results confirm that graph-based policies not only accelerate learning but also enable precise, stable, and composable locomotion behaviors.

D. Robustness Evaluation

To evaluate robustness under real-world–like uncertainties, we introduced controlled perturbations in simulation along three dimensions:

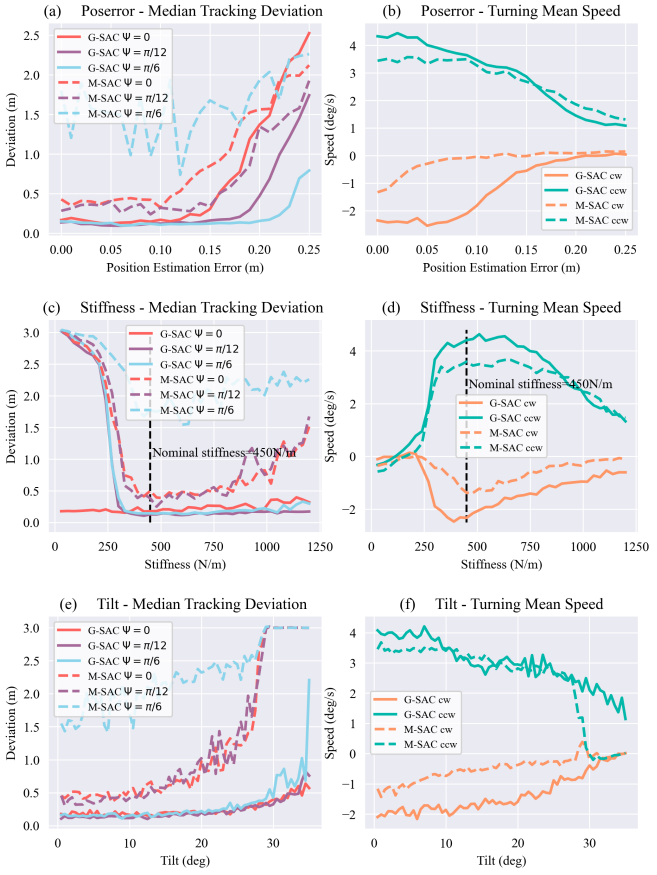

Fig. 7. Robustness evaluation under model and environment perturbations. (a) (b) Cross-tendon stiffness variation, (c) (d) Observation noise, (e) (f) Ground slope.

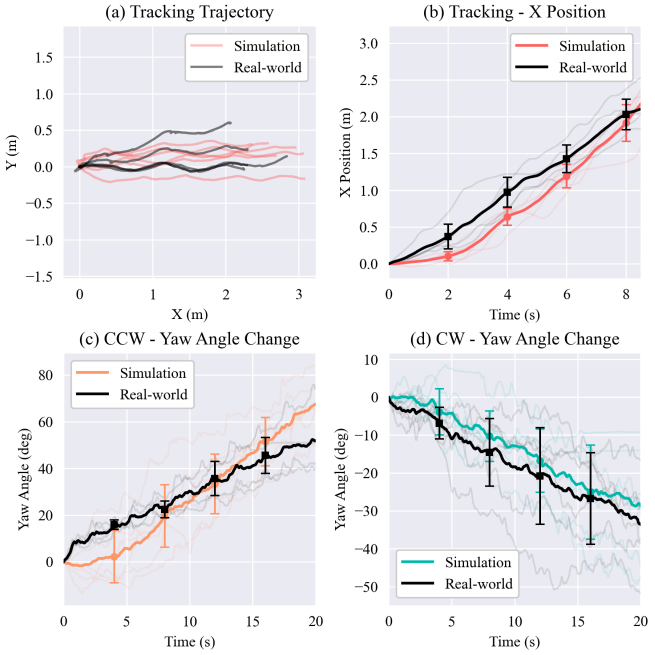

Fig. 8. Comparison between simulation and real-world performance. The proposed GNN-SAC achieves close agreement between simulated and physical results in (a) CoM trajectories, (b) forward tracking, and (c,d) bidirectional turning, validating robust zero-shot sim-to-real transfer.

TABLE I. Comparison between simulated and real-world performance over motion primitives

Motion primitive	Simulation	Real	Relative Error
Tracking (mm/s)	287	256	-12.1%
CCW Turning $(^\circ/s)$	3.38	2.76	-18.3%
CW Turning $(^\circ/s)$	1.63	1.72	+6%

- **Cross-tendon stiffness:** varied from 30–1200 N/m (nominal 450 N/m) to capture material and assembly variations.
- **State estimation noise:** Gaussian noise $\mathcal{N}(0, \sigma_n)$ with $\sigma_n \in [0, 0.25]$ m was added to end-cap positions to simulate degraded sensing.
- **Ground slope:** inclinations from 0° to 35° were tested using policies trained on flat terrain.

As summarized in Fig. 7, under observation noise up to $\sigma_n = 0.15$ m, tracking accuracy was maintained with only moderate degradation in turning speed. Turning rates preserved over half of their maximum for stiffness values between 300–800 N/m, and tracking performance remained stable within this range. When operating on inclined planes, the robot remained stable up to about 25° tilt, maintaining both climbing and turning capabilities. Across all perturbations and primitives, G-SAC consistently outperformed M-SAC, exhibiting smoother trajectories and greater tolerance to sensing variations, stiffness, and inclined surfaces.

These results indicate that embedding morphological structure in the policy network not only improves nominal performance but also enhances robustness to uncertainties critical for real-world deployment.

E. Sim-to-Real Validation

To validate sim-to-real transfer, the policies trained with G-SAC were deployed on a physical 3-bar tensegrity robot without any additional fine-tuning. All three motion primitives, straight-line tracking, clockwise turning, and counterclockwise turning, were successfully reproduced in real-world trials.

During straight-line tracking, the reference point was dynamically updated to remain 1 m ahead of the robot's current position, ensuring continuous forward motion. As shown in Fig. 8(a,b), the CoM trajectory remained close to the desired path as simulated, with an average lateral error of 0.058 m and a maximum of 0.281 m over a 2–3 m run. The robot reached an average velocity of 0.256 m/s before leaving the testing area.

For counterclockwise turning, the robot followed a distinct flipping-based gait, producing an average angular velocity of 2.76°/s with a maximum orientation error below 0.59° (Fig. 8(c)). In contrast, clockwise turning generated rotation through coordinated tendon actuation and cyclic fore–aft swinging, effectively avoiding toppling. The average angular velocity reached 1.72°/s, with a maximum orientation error below 0.45° (Fig. 8(d)).

To enable a fair comparison with simulation, the stiffness parameter was set to approximately 300 N/m, corresponding

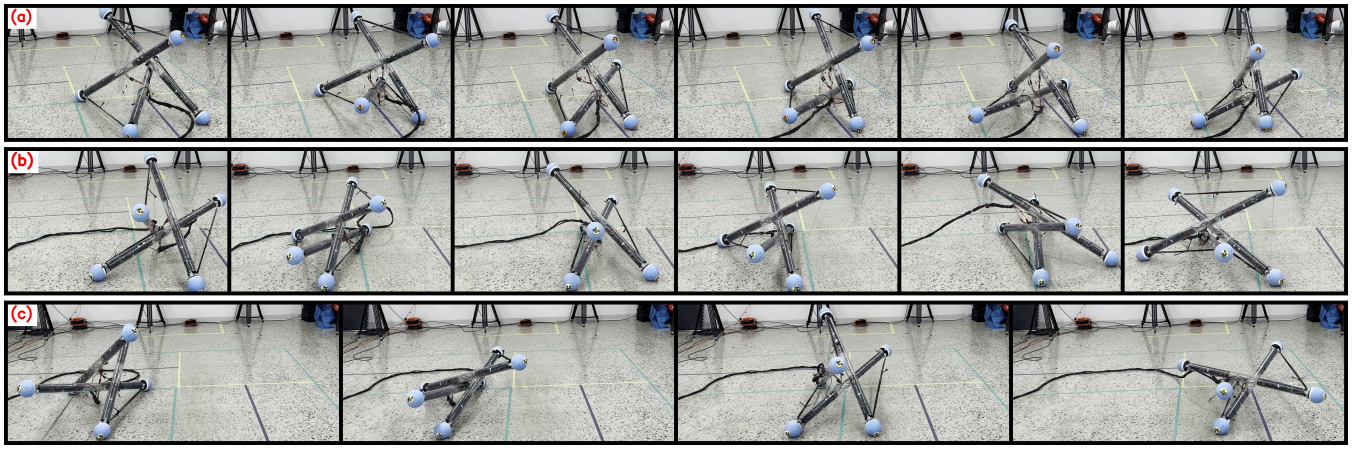

Fig. 9. Real-world rollout sequences of learned locomotion primitives. The tensegrity robot executes (a) clockwise turning, (b) counterclockwise turning, and (c) straight-line tracking using zero-shot transferred GNN-SAC policies. The sequences show coordinated rolling and stable motion across all tasks.

to the average stiffness of the climbing rope used in the real-world setup. The median simulation results are presented in Fig. 8 and Table I, which closely match the real-world measurements, with relative errors within 6–18%. Considering that these primitives were not explicitly designed for velocity tracking, such deviations remain reasonably small, indicating a satisfactory gap between the simulation and physical behavior.

Overall, these results confirm that the morphology-aware graph-based policies trained solely in simulation successfully transfer to the physical tensegrity robot in a zero-shot manner, demonstrating stable and coordinated locomotion across all three primitives.

VI. Conclusion

In this work, we proposed a morphology-aware graph reinforcement learning framework for tensegrity robot locomotion by integrating a graph neural network actor into the Soft Actor-Critic algorithm. By explicitly encoding the robot's structural topology, the GNN-based policy achieved higher sample efficiency, superior final rewards, and stronger robustness than MLP-based baselines. Simulation studies confirmed superior performance on tracking and turning tasks, and robustness under noise, variable stiffness and non-horizontal terrain. Importantly, the learned policies transferred directly to hardware without additional fine-tuning, enabling the 3-bar tensegrity robot to consistently accomplish straight-line tracking and bidirectional turning in real-world experiments. These results demonstrate the effectiveness of morphology-aware policy design for sim-to-real transfer in tensegrity locomotion. In future work, we plan to train a unified policy that can track linear and angular velocities for tensegrity robots.

References

[1] R. E. Skelton and M. C. De Oliveira, *Tensegrity systems*. Springer, 2009, vol. 1. [Online]. Available: https://link.springer.com/book/10.1007/978-0-387-74242-7

[2] W. Tong, T.-Y. Lin, J. Mi, Y. Jiang, M. Ghaffari, and X. Huang, "Tensegrity robot proprioceptive state estimation with geometric constraints," *IEEE Robotics and Automation Letters*, vol. 10, no. 4, pp. 4069–4076, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/10910166

[3] J. Mi, W. Tong, Y. Ma, and X. Huang, "Design of a variable stiffness quasi-direct drive cable-actuated tensegrity robot," *IEEE Robotics and Automation Letters*, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/11072300

[4] D. S. Shah, J. W. Booth, R. L. Baines, K. Wang, M. Vespignani, K. Bekris, and R. Kramer-Bottiglio, "Tensegrity robotics," *Soft robotics*, vol. 9, no. 4, pp. 639–656, 2022. [Online]. Available: https://www.liebertpub.com/doi/epub/10.1089/soro.2020.0170

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015. [Online]. Available: https://doi.org/10.1038/nature14236

[6] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," 2018. [Online]. Available: https://openreview.net/forum?id=HJjvxl-Cb

[7] C. Paul, F. Valero-Cuevas, and H. Lipson, "Design and control of tensegrity robots for locomotion," *IEEE Transactions on Robotics*, vol. 22, no. 5, pp. 944–957, 2006.

[8] C. Paul, J. Roberts, H. Lipson, and F. Valero Cuevas, "Gait production in a tensegrity based robot," in *ICAR '05. Proceedings., 12th International Conference on Advanced Robotics, 2005.*, 2005, pp. 216–222.

[9] K. Kim, A. K. Agogino, A. Toghyan, D. Moon, L. Taneja, and A. M. Agogino, "Robust learning of tensegrity robot control for locomotion through form-finding," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 5824–5831.

[10] K. Kim, A. K. Agogino, and A. M. Agogino, "Rolling locomotion of cable-driven soft spherical tensegrity robots," *Soft Robotics*, vol. 7, no. 3, pp. 346–361, 2020, pMID: 32031916. [Online]. Available: https://doi.org/10.1089/soro.2019.0056

[11] Y. Guo and H. Peng, "Full-actuation rolling locomotion with tensegrity robot via deep reinforcement learning," in *2021 5th International Conference on Robotics and Automation Sciences (ICRAS)*, 2021, pp. 51–55.

[12] K. Wang, W. R. Johnson, S. Lu, X. Huang, J. Booth, R. Kramer-Bottiglio, M. Aanjaneya, and K. Bekris, "Real2sim2real transfer for control of cable-driven robots via a differentiable physics engine," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 2534–2541.

[13] B. Cera and A. M. Agogino, "Multi-cable rolling locomotion with spherical tensegrities using model predictive control and deep learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–9.

[14] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa,

D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[15] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017. [Online]. Available: https://arxiv.org/abs/1707.06347

[16] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor-critic methods," in *International conference on machine learning*. PMLR, 2018, pp. 1587–1596.

[17] O. M. Andrychowicz, B. Baker, M. Chociej, R. Jozefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray *et al.*, "Learning dexterous in-hand manipulation," *The International Journal of Robotics Research*, vol. 39, no. 1, pp. 3–20, 2020.

[18] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, Jan. 2019. [Online]. Available: http://dx.doi.org/10.1126/scirobotics.aau5872

[19] D. Gandhi, L. Pinto, and A. Gupta, "Learning to fly by crashing," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 3948–3955.

[20] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 23–30.

[21] M. Zhang, X. Geng, J. Bruce, K. Caluwaerts, M. Vespignani, V. SunSpiral, P. Abbeel, and S. Levine, "Deep reinforcement learning for tensegrity robot locomotion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 634–641.

[22] J. Luo, R. Edmunds, F. Rice, and A. M. Agogino, "Tensegrity robot locomotion under limited sensory inputs via deep reinforcement learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6260–6267.

[23] D. Surovik, K. Wang, and K. E. Bekris, "Adaptive tensegrity locomotion on rough terrain via reinforcement learning," 2018. [Online]. Available: https://arxiv.org/abs/1809.10710

[24] D. Surovik, K. Wang, M. Vespignani, J. Bruce, and K. E. Bekris, "Adaptive tensegrity locomotion: Controlling a compliant icosahedron with symmetry-reduced reinforcement learning," *The International Journal of Robotics Research*, vol. 40, no. 1, pp. 375–396, 2021. [Online]. Available: https://doi.org/10.1177/0278364919859443

[25] T. Wang, R. Liao, J. Ba, and S. Fidler, "Nervenet: Learning structured policy with graph neural networks," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=S1sqHMZCb

[26] Z. Xiong, J. Beck, and S. Whiteson, "Universal morphology control via contextual modulation," in *International Conference on Machine Learning*. PMLR, 2023, pp. 38 286–38 300.

[27] N. Chen, K. Wang, W. R. Johnson III, R. Kramer-Bottiglio, K. Bekris, and M. Aanjaneya, "Learning differentiable tensegrity dynamics using graph neural networks," *arXiv preprint arXiv:2410.12216*, 2024.

[28] K. Wang, M. Aanjaneya, and K. Bekris, "Sim2sim evaluation of a novel data-efficient differentiable physics engine for tensegrity robots," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 1694–1701.