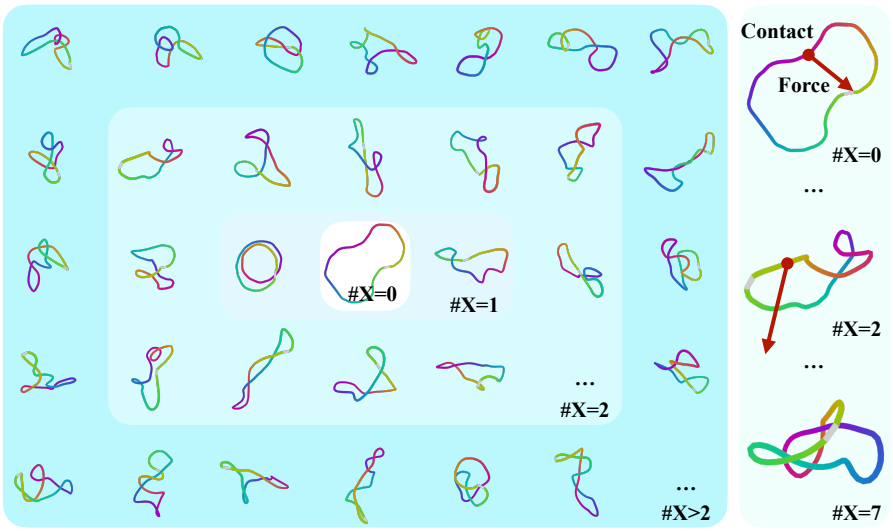
Knot So Simple:
A Minimalistic Environment for Spatial Reasoning

Zizhao Chen and Yoav Artzi
Department of Computer Science and Cornell Tech, Cornell University
{czz, yoav}@cs.cornell.edu

Abstract

We propose KNOTGYM, an interactive environment for complex, spatial reasoning and manipulation. KNOTGYM includes goal-oriented rope manipulation tasks with varying levels of complexity, all requiring acting from pure image observations. Tasks are defined along a clear and quantifiable axis of complexity based on the number of knot crossings, creating a natural generalization test. KNOTGYM has a simple observation space, allowing for scalable development, yet it highlights core challenges in integrating acute perception, spatial reasoning, and grounded manipulation. We evaluate methods of different classes, including model-based RL, model-predictive control, and chain-of-thought reasoning, and illustrate the challenges KNOTGYM presents. KNOTGYM is available at https://github.com/lil-lab/knotgym.



(a) Goal Space (b) Action Space

Figure 1: KNOTGYM is a visual reasoning knot manipulation environment. It includes three tasks: transforming complex knots to a simple loop (example in the center); tying a loop into complex knots (right pane); and converting one knot into another, given a goal knot image. KNOTGYM has a continuous visual observation space (left pane) and an action space of applying forces to contact points (right pane), abstracting the specifics of robot end effectors. The space of goals, specified using Gauss code, is a factorial of the number of crossings (#X), which creates a ladder of generalization. Each goal defines an easily testable equivalence class over a continuous set of states.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Track on Datasets and Benchmarks.

https://doi.org/10.52202/085713-2125

1 Introduction

KNOTGYM is a knot manipulation environment to train and evaluate visual reasoning agents. Figure 1 illustrates KNOTGYM. The agent observes the world state as an image and needs to transform a knot into a topological goal by applying forces to rope segments. The goal is communicated to the agent by observing a secondary knot as an exemplar. KNOTGYM defines three tasks (Table 1), each characterized by its start state and goal. KNOTGYM is designed with several aspects in mind:

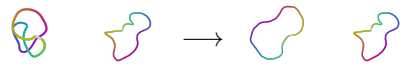
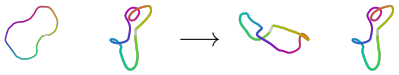
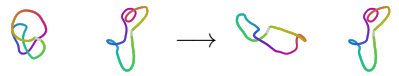
Research Challenges KNOTGYM evaluates spatial reasoning, action prediction, planning, and abstraction skills in a single environment. Solving a task requires analyzing the current rope configuration, planning on deforming it to achieve the goal, and mapping this plan to a sequence of continuous action predictions. Critically, KNOTGYM does not enforce one specific state as the goal – the goal is a large equivalence class of states. Goals are defined via Gauss code, a formal mathematical description of knot configuration based on its crossings. When the goal is communicated using an exemplar image, solving a task requires abstracting over the exemplar to identify the right set of actions to complete the task. We show that KNOTGYM is a significant challenge for contemporary methods.

Measurable Complexity Knots are well studied in mathematics, with formal descriptions like Gauss code and a measure of complexity in the number of crossings.[1] Figure 1 shows various knot configurations with different numbers of crossings, illustrating the increasing complexity of more crossings. This measure of complexity enables to control the challenges of learning and evaluation.

Generalization Ladder The number of crossings creates a clear generalization ladder. This allows a clear train-test complexity split, where we train up to a certain level of complexity (i.e., number of crossings) and test if a model generalizes beyond this number. During training, this allows us to experiment with a natural learning curriculum, where the number of crossings in training examples is gradually increased. It also enables the study of self-improving bootstrapping processes in a visual domain, as recently proposed for arithmetic and maze-solving [Lee et al., 2025b].

Research Accessibility KNOTGYM is an accessible task particularly suited for studying extended visual reasoning in a laboratory environment, just as Pendulum for classic control and Countdown for reasoning in natural language [Gandhi et al., 2024]. We follow the implementation and design principles of OpenAI Gym [Brockman et al., 2016] to make the KNOTGYM as accessible for researchers as possible. This includes following the standard Gymnasium API [Towers et al., 2024] and supporting vectorized environments on multiple CPUs. The KNOTGYM environment and our baselines are available under the MIT license at `https://github.com/lil-lab/knotgym`.

Table 1: The three tasks of KNOTGYM. We show the final observation assuming the goal is achieved. Correctly completing tasks does not require getting the exact exemplar configuration (right rope in each image pair), but a configuration with the same Gauss code, a notation that describes the abstract spatial characteristics of a knot. For example, the final observations in the `tie` row are both [1+,1-,2+,2-], despite their different appearances.

Task name	Description	Initial→ Final Observations
`unknot`	Untangle a knot into a simple loop	
`tie`	Tie a goal knot from a simple loop	
`convert`	Tie a new knot from an old knot	

[1]While the number of crossings is a fitting measure of complexity to our manipulation task, the prime complexity measure of interest in mathematics is more likely the counting of crossing of *irreducible knots* (i.e., as part of the tabulation of prime knots). Under this perspective, all the knots in Figure 1 are equivalent, because they all can be reduced to a simple loop by applying a few (or many) Reidemeister moves. For us, the number of crossings is a more appropriate measure because we aim for manipulation and observation complexity.

2 Related Work

Spatial Reasoning in Vision-language Models Spatial reasoning has been studied with benchmarks focused on language-vision tasks, specifically relations between individual objects using synthetic [Johnson et al., 2017, Suhr et al., 2017] and natural [Suhr et al., 2019, Liu et al., 2023] images. The evaluation of the spatial reasoning of agents is often embedded in locomotion or robotics manipulation tasks [Yang et al., 2025, Shridhar et al., 2020]. These tasks prioritize diverse domains and spatial relationships, but they are static and require limited reasoning. In contrast, KNOTGYM, is more narrow, but demands long, complex spatial reasoning, opening up opportunities to apply test-time-scaling reasoning in a visual domain. Similar narrow focus to KNOTGYM is deployed by lilGym [Wu et al., 2023], but with focus relations between rigid objects expressed in natural language.

Axes of Generalization Generalization is studied along many axes, such as length in text sequence models [Anil et al., 2022], size in graph neural networks [Yehudai et al., 2021], combining parts in new ways [Hupkes et al., 2020], performing arithmetic [Lee et al., 2025b, Dziri et al., 2023], and symbolic operations [Welleck et al., 2022]. KNOTGYM proposes a new *visual* generalization axis. We leverages the well-studied mathematical construct of knots to provide an established formulation to structure and discuss generalization evaluation.

Deformable Object Manipulation Manipulating deformable objects, such as liquid, playdough, and ropes, is notorious for its infinite dimensional configuration space [Lin et al., 2020]. Existing work [Yan et al., 2020, Sundaresan et al., 2021, Shi et al., 2024] focuses on learning task-specific representations, and aims to generalize robustly across different textual and material in the real world. KNOTGYM takes the idea of rope manipulation and simplifies the apparatus to emphasize complexity generalization. This allows us to assess the complex visual reasoning abilities of both RL methods and general pretrained VLMs. KNOTGYM also has a different goal formulation compared to classic robotics tasks. Our goal is to reach an abstract topological structure determined by the Gauss code, as opposed to minimizing a distance (i.e., Euclidean) between one set of coordinates and the goal coordinates. The latter can be formulated as an optimization problem, while the former relies more on abstract reasoning (even a form of searching).

Machine Learning (ML) and Knots The intersection of ML and knot theory research is limited, but promising. Davies et al. [2021] discovered a new connection between the algebraic and geometric structure of knots, using ML to guide human intuition. Part of KNOTGYM is an instantiation of a knot-theoretic problem called unknotting, a special instance of the generic and open problem of knot equivalence. To solve the same unknotting task, Gukov et al. [2021] learns a reinforcement learning powered search algorithm in symbolic space via braid words. In contrast, we focus on raw image observations and are interested in assessing a model's ability to reason about intuitive physics.

3 KnotGym

Intuitively, knots are ropes whose ends are joined. In KNOTGYM, agents manipulate such objects in a continuous 3D space by pulling on specific points in the rope (i.e., exerting force at a location), without breaking the continuity of the rope. Each knot is embedded in 3D space. The 3D coordinates of a knot are its *configuration*. Each episode has a topological goal expressed by an underlying Gauss code. The goal is specified via an exemplar placed in the environment, adjacent to the manipulated knot. The goal of each episode is to manipulate an initial knot configuration, via a series of actions, such that the final knot has the goal Gauss code (Figure 2). The agent in KNOTGYM does not have access to the world state or the Gauss code representing the goal, but instead receives visual observations only, reflecting our research interest in visual spatial reasoning. KNOTGYM can be easily extended to reveal those signals to the agent as well.

We now define and discuss the design of the KNOTGYM environment and tasks. Table 2 summarizes key terms. KNOTGYM implements the Gymnasium interface [Towers et al., 2024] for ease of use. Appendix C provides implementation details.

 https://doi.org/10.52202/085713-2125

Table 2: Key concepts in KNOTGYM

Concept	Explanation
(Knot) configuration	3D coordinates of key points along a single knot
Goal	Abstract spatial relationship of a knot uniquely identified by a Gauss code (many configurations can share the same Gauss code)
Goal configuration	A knot configuration that has the goal Gauss code
State	The environment includes both the manipulated knot and the goal, so the state specifies both of their configurations
Observation	An RGB image that is a 2D projection of the state onto the z-plane

3.1 Environment

KNOTGYM is an episodic partially observable Markov decision process (POMDP) $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \Omega, \mathcal{O}, H, \gamma)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the set of actions, $\mathcal{T}$ is the transition function, $\mathcal{R}$ is the reward function, Ω is the observation space, $\mathcal{O}$ is a mapping between states and image observations,[2] H is a termination criteria, and γ is the discount factor.

States $\mathcal{S}$ A state $s_t \in \mathcal{S}$ in time t is a pair (c_t^m, c^g) of two rope configurations, c_t^m is the current configuration of the manipulated rope, and c^g is the configuration of the goal exemplar. Knot configurations are represented by a series of 3D coordinates of key points along the rope. The goal configuration c^g satisfies the goal Gauss code and remains the same throughout an episode; we include it in the state space to construct a Markovian state and to keep the policy conditioning on the state only. The agent operating in KNOTGYM does not have access to the world state, but receives partial observations.

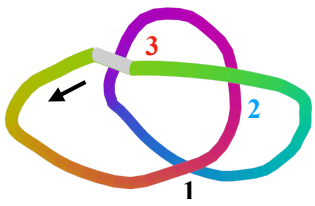


Figure 2: An episode is successful when the current knot configuration has the goal Gauss code. We obtain the Gauss code of any knot by traversing through the rope, starting from the white segment towards red (black arrow). When traversing, we denote an over-cross with +, and an under-cross with -, until we return to the starting segment.

Observations Ω An observation $o_t \in \Omega$ is an image generated by the observation mapping $\mathcal{O} : \mathcal{S} \to \Omega$. Concretely, o_t is rendered z-plane projections of both the current configuration c_t^m and the goal configuration c^g. Observation images have the shape $[3, 128, 2 \times 128]$ because we concatenate the rendered images of both configurations. Similar to [Lin et al., 2020], we use a larger image size than canonical deep RL environments because crossings are critical to determine the Gauss code, which is critical for the agent's reasoning. Crossings and self-occlusions also contribute to the partial observability of this problem. We reduce self-occlusions by tuning cable diameter and increasing the resolution.[3] We use 2D RGB observations because we are inspired by how humans perform predictive spatial reasoning from visual inputs only. KNOTGYM can be easily extended to symbolic observations or multiple cameras or RGBD observations.

Actions $\mathcal{A}$ An action $a \in \mathcal{A}$ is a 6-tuple (x, y, z, f_x, f_y, f_z), where (x, y, z) specifies a 3D location in state space that is rounded to the closest key point on the manipulated rope, and (f_x, f_y, f_z) is a force vector to apply to that key point. The action vector is tuned and normalized to $[-1, 1]^6$. Our main focus is spatial reasoning, so we abstract away end effectors that would introduce hardware-specific policies.[4] The transition model $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ captures the change in knot coordinates after applying the force for a short period of time, which is implemented by the MuJoCo physics simulator.

[2] KNOTGYM has a largely deterministic mapping from states to observations, even though there is negligible stochasticity because of the simulator.

[3] Cable diameter and resolution should be tuned when scaling to more complex knots (#X>8).

[4] Training with realistic end effectors is easy to add in the MoJoCo ecosystem, although tangential to our focus on abstract spatial reasoning.

https://doi.org/10.52202/085713-2125

Reward Function $\mathcal{R}$ The reward function $\mathcal{R} : \mathcal{S} \to \mathbb{R}$ is sparse and based on a symbolic oracle GC, which maps the rope coordinates to its Gauss code. The reward function is defined as

$$\mathcal{R}(c^m, c^g) = \mathbb{1}(GC(c^m) = GC(c^g)) \ .$$

The agent receives a single positive reward if the current knot configuration has the same Gauss code as the goal configuration. The agent needs to assess the Gauss code itself. The Gauss code is a formal description of the topology of a knot. It is computed by traversing through the rope and recording under-/over-crossings (Figure 2). Gauss codes are necessary but not sufficient to fully reconstruct a knot.[5] Table 3 shows how #X determines the number of possible Gauss codes. In addition, the reward function provides a negative penalty when reaching the horizon limit without success.

This criterion for success introduces a level of abstraction into KNOTGYM, where an agent must abstract the full set of correct termination states from the exemplar goal image. It is also more lenient than requiring reconstruction of the exact goal configuration, and as our experiments show, already challenging enough for existing methods (Section 4). Because the reward is based on topological equivalence, it does not always correspond to visual distance-based measures: two rope configurations can be completely different at first glance yet share the exact same underlying Gauss code; conversely two visually similar configurations can differ in their Gauss codes by just changing a single over-cross to an under-cross. This choice distinguishes KNOTGYM from other rope-configuration tasks, such as SoftGym [Lin et al., 2020], where dense rewards are computed from coordinate-wise distance that easily enable trajectory optimizations. KNOTGYM, in contrast, requires global and holistic spatial reasoning. It is not immediately clear, at least not to us, how to convert the tasks into smooth optimization problems.

Termination Criteria H There are two criteria for termination: completing the task (i.e., equal Gauss code, which also entails a positive reward) or reaching the horizon limit. An episode ends immediately upon reaching the goal Gauss code. An alternative, stricter reward function is to maintain the goal Gauss code for multiple frames (i.e., over a set time period), effectively learning a stopping behavior. Our preliminary experiments suggest that this variation makes the tasks even more difficult. It is easy to add to KNOTGYM.

Table 3: The factorial space of Gauss codes (GCs) with respect to the number of crossings (#X).

#X	# Possible GCs	Example GCs
0	1	{ [] }
1	2	{ [1+, 1-], [1-, 1+] }
2	12	{ [1+, 1-, 2+, 2-], [1+, 1-, 2-, 2+], [1-, 1+, 2+, 2-] ... }
n	$(2n-1)!!2^n$	...

3.2 Three Tasks

We design three tasks: `unknot`, `tie`, `convert`. They follow exactly the same environment interface as defined above, while only differing in the selection of initial state and the goal Gauss code, see Table 1. The complexity of each task can be tuned by setting the number of crossings in the initial rope configuration or the goal. For example, Table 4 shows the crossing settings we use in our experiments. We collect 40 knot configurations for each crossing setting (17 for the simple loop #X=0), reserve 20 configurations as the train split, and sample the initial goal configurations randomly at training and testing time. We set the horizon limit to 50 steps in our experiments to balance exploration and training costs. This is a hyperparameter that users can modify if they increase the number of crossings.

Table 4: Three tasks based on the number of crossings (#X) of the initial/goal knots.

Task name	Initial #X	Goal #X
`unknot`	{2,3,4}	{0}
`tie`	{0}	{2,3,4}
`convert`	{1,2,3}	{2,3,4}

The easiest of the three tasks is `unknot`, which requires untangling a knot into a simple loop (goal #X=0). The goal of `unknot` is shared across all episodes, so the policy does not have to reason about differing goals. `tie` is the inverse of `unknot`: it requires tying a knot from a simple loop. The policy is goal-conditioned: it needs to uncover the underlying goal Gauss code from an image, and

[5] An extended variation of Gauss code records additional chiral information.

https://doi.org/10.52202/085713-2125

manipulate the simple loop towards the identified goal Gauss code. `convert` is a combination of the other two and requires a wide range of abilities: identifying the goal Gauss code, recognizing the current topological structure, and planning accordingly.

3.3 Evaluation and Generalization

The primary evaluation metric is success rate: the percentage of episodes that with a positive reward (i.e., the manipulated knot configuration matches the goal Gauss code).

KNOTGYM offers at least two testable axes of generalization. The first is train/test generalization: we train the policy on initial/goal configurations from one split, and evaluate on unseen test configurations, keeping the task and the number of crossings #X the same. The second, and harder axis is complexity generalization, when the policy is trained on configurations up to a certain number of crossings (e.g., #X=2) and tested on configurations with a higher number (e.g., #X=3 and beyond). This notion of generalization is related to length generalization [Lee et al., 2025b, Dziri et al., 2023] or size generalization Yehudai et al. [2021]. A policy that generalizes well will succeed at the task even when the knot configurations are completely new and more complex. For training-free baselines (i.e., prompting proprietary VLMs), we assume that KNOTGYM is out of the pretraining and post-training distribution, and therefore it is always generalizing its training distribution.

3.4 Knot So Simple

What makes KNOTGYM both interesting and challenging? We identify three unique features KNOTGYM offers. These features are tightly integrated, making it suitable to test end-to-end systems.

Acute Perception Unlike the tasks in SoftGym [Lin et al., 2020], solving KNOTGYM relies heavily on correct identification of crossings, which often span only a few pixels. Effective policies must attend to such minute details to reason about both the goal and the current topological state.

Continuous Spatial Reasoning Unlike environments such as Ant-Maze [Fu et al., 2020], or ARC-AGI [Chollet, 2019], which have a discrete reasoning space, KNOTGYM, by its definition (a closed loop embedded in $\mathbb{R}^3$), is a naturally continuous environment with continuous states and action sets. There is no clear unit of action that would mark a branching point to enable the application of discrete planning methods. While it is conceptually possible to discretize the knot coordinates, for example, as a link diagram, and search over Reidemeister moves to achieve the goal Gauss code, KNOTGYM would still require mapping to grounded actions back in the continuous space.

Very Large Search Space The search space is large not only because KNOTGYM is a continuous space and the set of possible Gauss codes grows factorial with the number of crossings, but also because the knot equivalence problem is known to be hard. For the scope of this project, all knot configurations are reducible to the trivial knot, or the so-called unknot. Essentially, the policies are required to prove this by concretely finding a reduction path. As our experiments show, this already proves a significant challenge to state-of-the-art methods (Section 4). Whether we can decide if an arbitrary knot is reducible to a simple loop in polynomial time is an open math problem [Burton and Ozlen, 2012]. What's more, KNOTGYM can easily extend to include non-trivial knots, such as the trefoil and its variants. The north star of KNOTGYM is to drive the development of policies that learn to solve the knot equivalence problem, a generalization of the unknotting problem.

4 Experiments

We evaluate representative methods of different types on KNOTGYM: PPO (model-free RL) [Schulman et al., 2017], DreamerV3 (model-based RL) [Hafner et al., 2025a], TM-MPC2 (RL with test-time search) [Hansen et al., 2024], and VLMs via chain-of-thought prompting using GPT-4.1-nano [Achiam et al., 2023]. Our results characterize the strengths and weaknesses of each method. We discuss RL and prompting methods separately, as the experiments bring different insights.

Table 5: Benchmarking representative methods over nine KNOTGYM setups. Entries are training split success rates calculated over N rollouts. For RL, measurements are taken at 1M steps.

Task	#X	Random	RL (N=384)			Prompting (N=105±7)		
		(N=256)	DreamerV3	PPO	TD-MPC2	Open	Stateless	Stateful
`unknot`	2	11.1	93.3	65.7	71.3	0.0	12.0	20.9
`unknot`	3	8.4	93.4	63.3	55.4	0.0	6.7	17.0
`unknot`	4	6.7	89.3	37.0	50.3	0.0	6.1	7.4
`tie`	2	35.9	83.2	41.3	39.2	0.0	7.6	5.5
`tie`	3	2.5	16.1	3.3	4.6	0.0	1.0	0.9
`tie`	4	1.4	4.1	1.3	1.7	0.0	0.0	0.0
`convert`	2	36.8	71.5	47.7	40.8	0.9	5.7	2.8
`convert`	3	9.2	15.3	6.3	8.7	0.0	1.0	0.9
`convert`	4	2.9	5.4	4.7	3.8	0.0	0.0	0.0

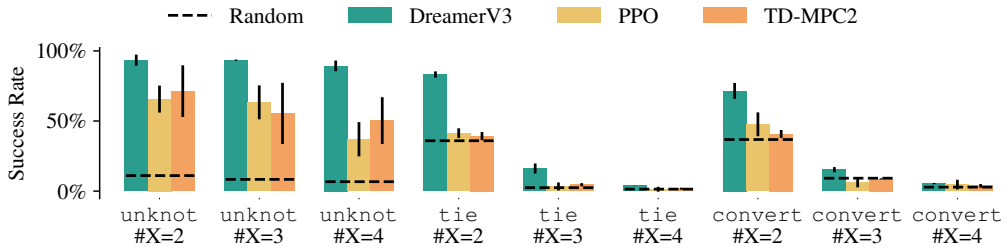


Figure 3: Train success rates of RL methods on nine different KNOTGYM setups after 1M environment steps during training. Error bars represent 95% confidence interval. All methods show non-trivial improvements on `unknot` via RL training, but struggle on `tie` and `convert`. No methods outperform a random policy at #X=4 of `tie` and `convert`, suggesting that increasing #X raises task difficulty significantly for tasks with many possible goals.

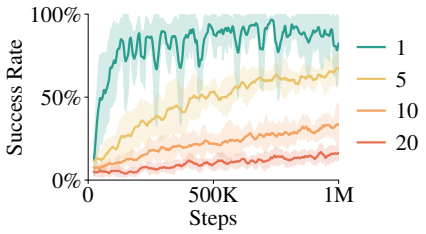


Figure 4: Training curves for different number of goal configurations in the training set (DreamerV3, `tie`, #X=3).

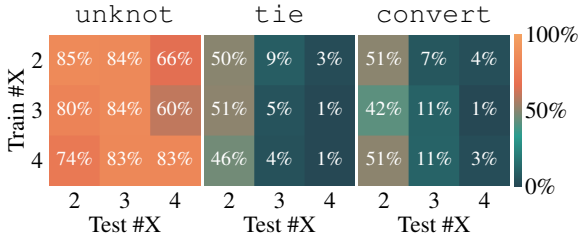


Figure 5: Generalization matrices for three tasks. Each entry of the matrices is success rate evaluated on the test split with N=128 episodes.

4.1 Reinforcement Learning Methods

We experiment with the vision variant of PPO implementation in Stable-Baselines3 [Raffin et al., 2021], and the official implementations of TD-MPC2 and DreamerV3, following the default configurations as much as possible. We include detailed hyperparameters in Appendix B and open-source benchmarking code for reproducibility. Each RL training run has a fixed budget of 1M environment steps. Table 5 and Figures 3–6 present the results and analysis. Appendix D presents additional results including all training curves in Figure 13.

Comparing the Three Tasks All methods show non-trivial performance on the easiest `unknot` task after training, with DreamerV3 performing particularly well. However, results on `tie` or `convert` are significantly weaker. A potential explanation is that `unknot` is slightly simpler because the goal is always the same, and the agent does not need to decode the Gauss code from the goal exemplar. Qualitative analysis (Figure 6) shows that all methods recover a simple strategy to solve `unknot`: dragging one section of the rope in one direction and letting inertia untangle the knot. This generalizes well to more crossings. In contrast, `tie` and `convert` require much more careful reasoning about the

https://doi.org/10.52202/085713-2125

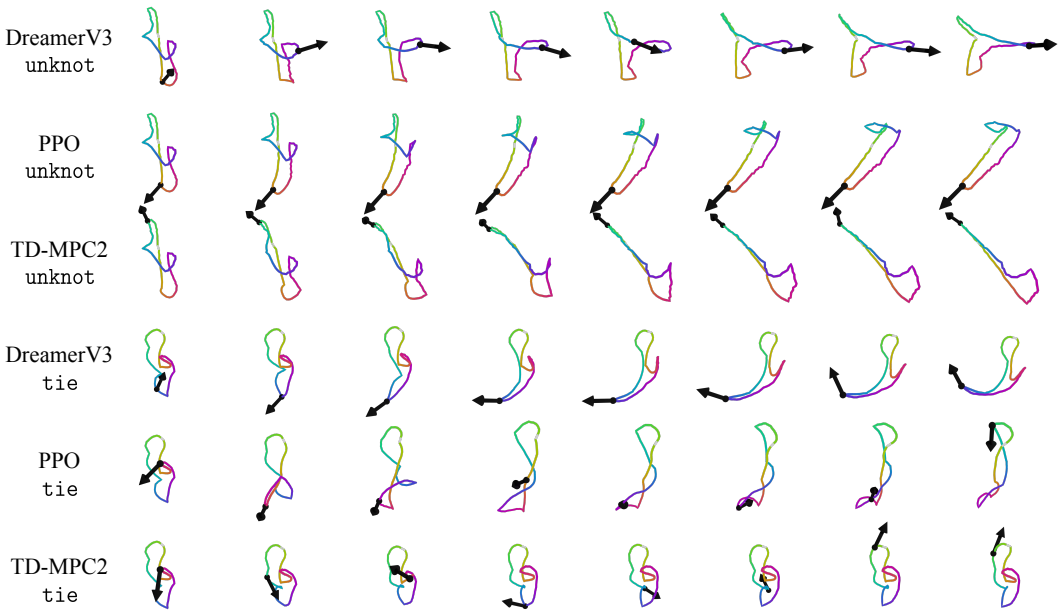


Figure 6: RL policy rollouts on `unknot` and `tie`. All methods find a solution to `unknot` by consistently pulling (black arrows) one segment of the rope. However, for `tie` the actions are more diverse and spread out without a consistent pattern, highlighting the difficulty of `tie`.

goal and condition the policy on the decoded goal Gauss code per episode, which contributes to the difficulty of `tie` and `convert`.

Training Difficulty and the Number of Crossings (#X) Increasing the number of crossings raises task difficulty significantly for `tie` and `convert`. Our random baseline policy samples actions uniformly. The random baselines for #X=2 solve a significant number of tasks because random walking for 50 timesteps likely stays in the space of #X=2. This is likely helpful for exploration early on during RL training for `convert` at #X=2. At #X=3, only DreamerV3 learned a policy clearly better than random for `tie` and `convert`. With one more crossing (#X=4), all methods are merely marginally better than random policy at `tie` and `convert`, even on the training tasks. This highlights the training challenges with larger #X. We hypothesize this challenge is related to the size of the training pool, so we examine the training curves when we constrain the diversity of goal configurations seen during training, reducing from the default 20 goal configurations to 10, 5, and 1 (Figure 4). Indeed, the variety of goal configurations presents a challenge for RL training.

Generalization We train three DreamerV3 policies for `unknot` and `tie` on a pool of 20 initial configurations with #X=2,3,4 respectively, and evaluate on held-out test initial configurations with #X=2,3,4. Figure 5 shows generalization performance. `unknot` policies generalize well: policies trained on #X=2 work well on new #X=2 knots, and generalize to new #X=3 knots, and even, to a lesser degree, to new #X=4 knots. For the harder `tie`, the #X=2 policy, despite reaching over 80% training success rate, only narrowly outperforms the random policy on the test set of new #X=2 knots, suggesting overfitting to training examples. There is also little goal-generalization on `tie`, as the test success rates are close to, if not lower than, the respective random policies.

4.2 Prompting

We evaluate GPT-4.1-nano on KnotGym. Because it is fair to assume that KnotGym is a new task for current VLMs, prompting naturally evaluates train/test and complexity generalization.[6] Each prompt is prefixed with detailed instructions about the task, definition of Gauss code, and tuples of before-after comparisons to convey system dynamics. We experiment with several prompting modes per task, with 100 episodes per task-mode pair.

[6]Early experiments show that "thinking" models (`o1`) do not have an edge over regular models in KnotGym.

https://doi.org/10.52202/085713-2125

 # System instructions and visual inputs are removed for brevity.
user: What are a series of actions that will transform the current knot such that it has the same gauss code as the goal knot? Think step by step, and end your answer in one <answer></answer> block, like: <answer> [-0.8, 0.8, 0.0, 0.9, 0.0, 0.0] [0.0, 0.2, 0.0, -0.7, 0.0, 0.0] </answer>. You can include multiple lists of six floats in the block.
model: Let's analyze the current and goal knots based on the provided images and gauss codes. The goal gauss code of the target knot appears to be an unknot (flat loop), which corresponds to an empty gauss code []. The current knot, with its filled gauss code, is more tangled, but the goal is to straighten it into an unknotted loop.
To achieve this, the main step is to loosen and straighten the loop, removing all crossings. This can be done by applying gentle forces to the segments to unfold and align the rope into a flat, unknotted circle.
Step-by-step plan:
1. Identify the segments roughly at the center of the loop and gently pull them outward to straighten the loop.
2. Slightly push the ends toward each other to ensure a smooth, flat configuration.
3. Apply minimal force to manipulate the loop into a flat state, removing intersections.
Assuming the rope is centered at (0,0,0), and segments are spread out, approximate actions are:
- Pull the top middle segment slightly upward.
- Pull the bottom middle segment slightly downward.
- Pull the right middle segment slightly to the right.
- Pull the left middle segment slightly to the left.
- Slightly push the segment near the point of crossing to straighten the loop.
Here are the approximate actions: <answer> [0.0, 0.5, 0.0, 0.0, 0.3, 0.0] [0.0, -0.5, 0.0, 0.0, -0.3, 0.0] [0.5, 0.0, 0.0, 0.3, 0.0, 0.0] [-0.5, 0.0, 0.0, -0.3, 0.0, 0.0] [0.0, 0.0, 0.0, 0.0, 0.0, 0.2] </answer>

Figure 7: Example prompt and response in an open mode query. The VLM recognizes the goal Gauss code and forms a reasonable plan to reach the goal. However, during rollouts the actions are too weak to achieve the desired effect, highlighting the brittleness of naive world modeling in VLMs.

Prompting Modes We evaluate three prompting modes. They differ in the control loop (open versus closed) and whether memory is retained for future queries: (a) *open* mode: the VLM produces a sequence of actions in one shot, given the initial observation o_0, including images of both current c_0^m and goal c^g configurations. (b) *stateless* mode: the VLM produces one action at a time, given the current observation $o_t \sim \mathcal{O}(c_t, c^g)$; and (c) *stateful* mode: the VLM also produces one action at a time, but it observes a windowed history of past observations and actions. The open/closed control loop (open vs. stateless/stateful) offers an opportunity to assess VLM dynamics without seeing the effects of its own actions. The stateful/stateless comparison is based on the assumption that memory would help further contextualize the system dynamics. Prompt examples are included in Appendix A.

Results Prompting-based methods perform worse than RL baselines in general (Table 5). Most prompting baselines are worse than a random policy, except for `unknot`, #X=2,3. The observation history offered by stateful prompting seems to help, as evidenced on the `unknot` task. Figure 7 presents one example VLM response to the open query. The VLM recognizes the goal Gauss code is [] (the task is `unknot`) and forms a reasonable plan to untangle the knot. However, during rollouts, the actions are too "weak" and not precise enough to achieve the desired effect. Action imprecision issues persist even with closed-loop feedback, highlighting the inefficiency of naive world modeling using VLMs to perform fine manipulations.

5 Discussion

We present KNOTGYM, a new interactive environment for complex spatial reasoning and manipulation. We benchmark general-purpose RL methods of different classes, and show a gradient of difficulty: `unknot` is relatively easy, but `tie` and `convert` pose a significant generalization challenge to state-of-the-art RL methods. Key to enabling this observation is the well-defined generalization ladder knot theory provides. KNOTGYM also presents a third type of generalization underexplored in this work that can be informally called *causal generalization*. For example, we can train the policy on `tie` task (the forward direction) trajectories, and evaluate whether the policy generalizes to the `unknot` task in the backward direction.

 https://doi.org/10.52202/085713-2125

KNOTGYM admits a broad range of solutions. RL methods posed by the vast goal space generally struggle with data inefficiency. Prompting VLMs with curated prompts, while succeeding at understanding the goal and performing some spatial reasoning, cannot produce fine, grounded actions. Including multi-turn interaction history in the prompt helps only to a limited extent. KNOTGYM provides an excellent testbed for evaluating other frontier visual reasoning agents, for instance, agents that verbalize action space reasoning [Lee et al., 2025a, MolmoAct], agents that rollout within the learned latent model [Hafner et al., 2025b, Dreamer4] or in image space based on unified multimodal models [Wu et al., 2024, VILA-U], agents that leverages pretrained video models as world models [Ball et al., 2025, Genie3], even visual coding agents that construct an explicit physics-based model of the rope through interaction.

KNOTGYM is now ready to enable the research of a broad set of research questions. We continue to develop it, with a plan to address several limitations in the near future. The first is simulator capacity, including improving simulation fidelity (e.g., to improve scaling to a high number of crossings and longer ropes), and throughput (e.g., currently 20 environment steps per second on a single CPU after applying frame skip) by moving from CPUs to GPUs.

Acknowledgments and Disclosure of Funding

This research was supported by NSF under grants No. 1750499, a gift from Open Philanthropy, an Nvidia Academic Grant, the National Artificial Intelligence Research Resource (NAIRR) Pilot, the Frontera supercomputer supported by the National Science Foundation (award NSF-OAC 1818253) at the Texas Advanced Computing Center (TACC) at The University of Texas at Austin, and the Delta advanced computing and data resource which is supported by the National Science Foundation (award NSF-OAC 2005572). We gratefully acknowledge use of the research computing resources of the Empire AI Consortium, Inc, with support from the State of New York, the Simons Foundation, and the Secunda Family Foundation [Bloom et al., 2025]. We thank Haochen Shi for helpful discussions and proofreading, and Steve Marschner and Kuan Fang for advice on 3D simulation.

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, NASA, or the other funders.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Cem Anil, Yuhuai Wu, Anders Andreassen, Aitor Lewkowycz, Vedant Misra, Vinay Ramasesh, Ambrose Slone, Guy Gur-Ari, Ethan Dyer, and Behnam Neyshabur. Exploring length generalization in large language models. *Advances in Neural Information Processing Systems*, 35:38546–38556, 2022.

Philip J. Ball, Jakob Bauer, Frank Belletti, Bethanie Brownfield, Ariel Ephrat, Shlomi Fruchter, Agrim Gupta, Kristian Holsheimer, Aleksander Holynski, Jiri Hron, Christos Kaplanis, Marjorie Limont, Matt McGill, Yanko Oliveira, Jack Parker-Holder, Frank Perbet, Guy Scully, Jeremy Shar, Stephen Spencer, Omer Tov, Ruben Villegas, Emma Wang, Jessica Yung, Cip Baetu, Jordi Berbel, David Bridson, Jake Bruce, Gavin Buttimore, Sarah Chakera, Bilva Chandra, Paul Collins, Alex Cullum, Bogdan Damoc, Vibha Dasagi, Maxime Gazeau, Charles Gbadamosi, Woohyun Han, Ed Hirst, Ashyana Kachra, Lucie Kerley, Kristian Kjems, Eva Knoepfel, Vika Koriakin, Jessica Lo, Cong Lu, Zeb Mehring, Alex Moufarek, Henna Nandwani, Valeria Oliveira, Fabio Pardo, Jane Park, Andrew Pierson, Ben Poole, Helen Ran, Tim Salimans, Manuel Sanchez, Igor Saprykin, Amy Shen, Sailesh Sidhwani, Duncan Smith, Joe Stanton, Hamish Tomlinson, Dimple Vijaykumar, Luyu Wang, Piers Wingfield, Nat Wong, Keyang Xu, Christopher Yew, Nick Young, Vadim Zubov, Douglas Eck, Dumitru Erhan, Koray Kavukcuoglu, Demis Hassabis, Zoubin Gharamani, Raia Hadsell, Aäron van den Oord, Inbar Mosseri, Adrian Bolton, Satinder Singh, and Tim Rocktäschel. Genie 3: A new frontier for world models. 2025.

Stacie Bloom, Joshua C. Brumberg, Ian Fisk, Robert J. Harrison, Robert Hull, Melur Ramasubramanian, Krystyn Van Vliet, and Jeannette Wing. Empire AI: A new model for provisioning AI and HPC for academic research in the public good. In *Practice and Experience in Advanced Research Computing (PEARC '25)*, page 4, Columbus, OH, USA, July 2025. ACM. doi: 10.1145/3708035.3736070. URL `https://doi.org/10.1145/3708035.3736070`.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Benjamin A Burton and Melih Ozlen. A fast branching algorithm for unknot recognition with experimental polynomial-time behaviour. *arXiv preprint arXiv:1211.1079*, 2012.

François Chollet. On the measure of intelligence. *arXiv preprint arXiv:1911.01547*, 2019.

Alex Davies, Petar Veličković, Lars Buesing, Sam Blackwell, Daniel Zheng, Nenad Tomašev, Richard Tanburn, Peter Battaglia, Charles Blundell, András Juhász, et al. Advancing mathematics by guiding human intuition with ai. *Nature*, 600(7887):70–74, 2021.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, et al. Faith and fate: Limits of transformers on compositionality. *Advances in Neural Information Processing Systems*, 36:70293–70332, 2023.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D Goodman. Stream of search (sos): Learning to search in language. *arXiv preprint arXiv:2404.03683*, 2024.

Sergei Gukov, James Halverson, Fabian Ruehle, and Piotr Sułkowski. Learning to unknot. *Machine Learning: Science and Technology*, 2(2):025035, 2021.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640(8059):647–653, 2025a. ISSN 1476-4687. doi: 10.1038/s41586-025-08744-2. URL https://www.nature.com/articles/s41586-025-08744-2.

Danijar Hafner, Wilson Yan, and Timothy Lillicrap. Training agents inside of scalable world models. *arXiv preprint arXiv:2509.24527*, 2025b.

Nicklas Hansen, Hao Su, and Xiaolong Wang. Td-mpc2: Scalable, robust world models for continuous control. In *International Conference on Learning Representations (ICLR)*, 2024.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. Compositionality decomposed: How do neural networks generalise? *Journal of Artificial Intelligence Research*, 67:757–795, 2020.

Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2901–2910, 2017.

Jason Lee, Jiafei Duan, Haoquan Fang, Yuquan Deng, Shuo Liu, Boyang Li, Bohan Fang, Jieyu Zhang, Yi Ru Wang, Sangho Lee, Winson Han, Wilbert Pumacay, Angelica Wu, Rose Hendrix, Karen Farley, Eli VanderBilt, Ali Farhadi, Dieter Fox, and Ranjay Krishna. Molmoact: Action reasoning models that can reason in space, 2025a. URL https://arxiv.org/abs/2508.07917.

Nayoung Lee, Ziyang Cai, Avi Schwarzschild, Kangwook Lee, and Dimitris Papailiopoulos. Self-improving transformers overcome easy-to-hard and length generalization challenges. *ArXiv*, abs/2502.01612, 2025b. URL https://api.semanticscholar.org/CorpusID:276107223.

Xingyu Lin, Yufei Wang, Jake Olkin, and David Held. Softgym: Benchmarking deep reinforcement learning for deformable object manipulation. In *Conference on Robot Learning*, 2020.

Fangyu Liu, Guy Emerson, and Nigel Collier. Visual spatial reasoning. *Transactions of the Association for Computational Linguistics*, 11:635–651, 2023.

Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22 (268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Haochen Shi, Huazhe Xu, Zhiao Huang, Yunzhu Li, and Jiajun Wu. Robocraft: Learning to see, simulate, and shape elasto-plastic objects in 3d with graph networks. *The International Journal of Robotics Research*, 43 (4):533–549, 2024.

 https://doi.org/10.52202/085713-2125

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Alane Suhr, Mike Lewis, James Yeh, and Yoav Artzi. A corpus of natural language for visual reasoning. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 217–223, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-2034. URL https://aclanthology.org/P17-2034/.

Alane Suhr, Stephanie Zhou, Ally Zhang, Iris Zhang, Huajun Bai, and Yoav Artzi. A corpus for reasoning about natural language grounded in photographs. In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6418–6428, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1644. URL https://aclanthology.org/P19-1644/.

Priya Sundaresan, Jennifer Grannen, Brijen Thananjeyan, Ashwin Balakrishna, Jeffrey Ichnowski, Ellen Novoseller, Minho Hwang, Michael Laskey, Joseph E Gonzalez, and Ken Goldberg. Untangling dense non-planar knots by learning manipulation features and recovery policies. *arXiv preprint arXiv:2107.08942*, 2021.

Alexander J Taylor and other SPOCK contributors. pyknotid knot identification toolkit. https://github.com/SPOCKnots/pyknotid, 2017.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulao, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024.

Sean Welleck, Peter West, Jize Cao, and Yejin Choi. Symbolic brittleness in sequence models: on systematic generalization in symbolic mathematics. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8629–8637, 2022.

Anne Wu, Kiante Brantley, Noriyuki Kojima, and Yoav Artzi. lilGym: Natural language visual reasoning with reinforcement learning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9214–9234. Association for Computational Linguistics, 2023. URL https://aclanthology.org/2023.acl-long.512.

Yecheng Wu, Zhuoyang Zhang, Junyu Chen, Haotian Tang, Dacheng Li, Yunhao Fang, Ligeng Zhu, Enze Xie, Hongxu Yin, Li Yi, et al. Vila-u: a unified foundation model integrating visual understanding and generation. *arXiv preprint arXiv:2409.04429*, 2024.

Mengyuan Yan, Gen Li, Yilin Zhu, and Jeannette Bohg. Learning topological motion primitives for knot planning. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9457–9464. IEEE, 2020.

Rui Yang, Hanyang Chen, Junyu Zhang, Mark Zhao, Cheng Qian, Kangrui Wang, Qineng Wang, Teja Venkat Koripella, Marziyeh Movahedi, Manling Li, et al. Embodiedbench: Comprehensive benchmarking multi-modal large language models for vision-driven embodied agents. *arXiv preprint arXiv:2502.09560*, 2025.

Gilad Yehudai, Ethan Fetaya, Eli Meirom, Gal Chechik, and Haggai Maron. From local structures to size generalization in graph neural networks. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 11975–11986. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/yehudai21a.html.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We claimed that we propose a new environment suitable for measuring generalization, and evaluate a range of methods. In the paper, we carefully define KNOTGYM, present and analyze results from benchmarking PPO, DreamerV3, and TDMPC2, as well as three prompting based methods. Indeed, our environment offers a gradient of generalization challenge.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations in the last paragraph of the paper and lay out plans to improve simulation quality when scaled to longer ropes and more crossings.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

https://doi.org/10.52202/085713-2125

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: We do not have theoretical results.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We describe detailed environment configuration and benchmark protocols in the appendix for reproducibility. We provide code for KNOTGYM and benchmarking during review, and will open source them upon publication.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility.

In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide code at `https://github.com/lil-lab/knotgym`, and training protocols in the appendix.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide key experimental settings in Section 3.2 and full experimental settings in the appendix. As we are benchmarking existing methods, we follow their setup as closely as possible.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We provide 95% confidence intervals with closed form formula for all training runs across three random seeds. We also provide number of rollouts (N) when reporting train/test task success rates.

   Guidelines:

https://doi.org/10.52202/085713-2125

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide a summary of computational resource requirements for each of our experiments in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics and confirm that the research conducted in the paper conform with the Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

Justification: We propose a scoped environment that lives completely in simulation. The proposed environment is intended for evaluating fundamental RL algorithms and existing pretrained models. We are not aware of immediate societal impact of designing and building this simulation environment.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: We propose a synthetic environment about knots, that does not require human data inputs, nor generate outputs subject to human consumption.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We created KnotGym on top of several open source libraries. Their creators are credited in the appendix and in the code base.

Guidelines:

 https://doi.org/10.52202/085713-2125

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: KNOTGYM implements the standard gym API and we provide documentations and example usages.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: This research does not involve crowdsourcing with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

Justification: This paper does not involve research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs as an important, original, or non-standard components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

https://doi.org/10.52202/085713-2125

A Prompt Examples

We provide the prompt instances for each prompting mode: open prompt (Figure 8), stateless prompt (Figure 9), and stateful prompt (Figure 10). The first image of the user prompt (i.e., first <image> tag) is the same across the three, and is shown in Figure 11.

user: Output a series of actions to transform the knot from its initial configuration to the goal gauss code.
user: <image>
user: Goal specification: The conversion is considered successful, when the current knot has the same gauss code as the goal knot. When determining the gauss code, always start from the white segment and traverse the rope towards the red segment, record positive for over-cross and negative for under-cross. A visual example is included in the image. An flat loop has gauss code of [].
user: Action specification: We follow a right-hand coordinate system, centered in the figure. Each action is in the form of [x,y,z,fx,fy,fz] where (x,y,z) are 3D coordinates which will be rounded to the closest rope segment, and (fx,fy,fz) are force vectors to be applied to that rope segment. x,y,z,fx,fy,fz are floating points bounded by [-1, 1]. In the image are three examples of before-and-after pairs of the unit directions. Use them as a reference. You can compose an action, for example, [1.0,1.0,0.0,0.9,0.0,-0.7] means pulling the most upper-right segment with 0.9 unit force in +x direction and 0.7 unit force in -z direction. You can select a segment somewhere in the middle of the rope, for example, let [x,y,z]=[-0.5,0.5,0.0] would be in the center of second quadrant.
user: Now consider a goal knot of the goal gauss code (what's the gauss code of the following knot?):
user: <image>
user: Here is the current knot:
user: <image>
user: What are a series of actions that will transform the current knot such that it has the same gauss code as the goal knot? Think step by step, and end your answer in one <answer></answer> block, like: <answer> [-0.8, 0.8, 0.0, 0.9, 0.0, 0.0] [0.0, 0.2, 0.0, -0.7, 0.0, 0.0] </answer>. You can include multiple lists of six floats in the block, separated by new lines.

Figure 8: Example open prompt and response.

user: Output a series of actions to transform the knot from its initial configuration to the goal gauss code.
user: <image>
user: Goal specification: The conversion is considered successful, when the current knot has the same gauss code as the goal knot. When determining the gauss code, always start from the white segment and traverse the rope towards the red segment, record positive for over-cross and negative for under-cross. A visual example is included in the image. An flat loop has gauss code of [].
user: Action specification: We follow a right-hand coordinate system, centered in the figure. Each action is in the form of [x,y,z,fx,fy,fz] where (x,y,z) are 3D coordinates which will be rounded to the closest rope segment, and (fx,fy,fz) are force vectors to be applied to that rope segment. x,y,z,fx,fy,fz are floating points bounded by [-1, 1]. In the image are three examples of before-and-after pairs of the unit directions. Use them as a reference. You can compose an action, for example, [1.0,1.0,0.0,0.9,0.0,-0.7] means pulling the most upper-right segment with 0.9 unit force in +x direction and 0.7 unit force in -z direction. You can select a segment somewhere in the middle of the rope, for example, let [x,y,z]=[-0.5,0.5,0.0] would be in the center of second quadrant.
user: Now consider a goal knot of the goal gauss code (what's the gauss code of the following knot?):
user: <image>
user: Here is the current knot:
user: <image>
user: What is the next action to take? Think step by step, and end your answer in one <answer></answer> block, like: <answer> [0.0, 0.2, 0.0, -0.7, 0.0, 0.0] </answer>. You should only include one list of six floats in the block.

Figure 9: Example stateless prompt and response.

user: Output a series of actions to transform the knot from its initial configuration to the goal gauss code.
user: <image>
user: Goal specification: The conversion is considered successful, when the current knot has the same gauss code as the goal knot. When determining the gauss code, always start from the white segment and traverse the rope towards the red segment, record positive for over-cross and negative for under-cross. A visual example is included in the image. An flat loop has gauss code of [].
user: Action specification: We follow a right-hand coordinate system, centered in the figure. Each action is in the form of [x,y,z,fx,fy,fz] where (x,y,z) are 3D coordinates which will be rounded to the closest rope segment, and (fx,fy,fz) are force vectors to be applied to that rope segment. x,y,z,fx,fy,fz are floating points bounded by [-1, 1]. In the image are three examples of before-and-after pairs of the unit directions. Use them as a reference. You can compose an action, for example, [1.0,1.0,0.0,0.9,0.0,-0.7] means pulling the most upper-right segment with 0.9 unit force in +x direction and 0.7 unit force in -z direction. You can select a segment somewhere in the middle of the rope, for example, let [x,y,z]=[-0.5,0.5,0.0] would be in the center of second quadrant.
user: Now consider a goal knot of the goal gauss code (what's the gauss code of the following knot?):
user: <image>
user: Here is the current knot:
user: <image>
user: What is the next action to take? Think step by step, and end your answer in one <answer></answer> block, like: <answer> [0.0, 0.2, 0.0, -0.7, 0.0, 0.0] </answer>. You should only include one list of six floats in the block.
model: First, observe the current knot configuration and note where the segments are located in 3D space. The goal is to simplify the knot or move it toward a configuration with a known gauss code.
Next, identify a segment that can be manipulated to straighten the knot or reduce crossings. The middle segment appears to be at about [-0.2, 0.3, 0.0], which is close to the center and can be pulled outward or inward.
A logical move is to gently pull the middle part of the knot to help straighten or reposition it, making subsequent manipulations easier.
Therefore, the next action should be to apply a force to the central segment, pulling it outward along the x-axis to reduce complexity.
<answer>[0.0, 0.3, 0.0, 0.8, 0.0, 0.0]</answer>
user: Here is the current knot:
user: <image>
user: What is the next action to take? Think step by step, and end your answer in one <answer></answer> block, like: <answer> [0.0, 0.2, 0.0, -0.7, 0.0, 0.0] </answer>. You should only include one list of six floats in the block.

Figure 10: Example stateful prompt and response.

B Benchmarking Details

This sections details to help with reproducibility. When a hyperparameter is not specified, we use the default value from the official implementation. Deviations from the default settings are **bolded**.

B.1 PPO

We use the Stable-Baselines3 [Raffin et al., 2021] implementation of vision PPO. Table 6 lists hyperparameters values. The total number of trainable model parameters is 16.5 M. We have 32 vectorized environments. Each training run was conducted on 32 CPU cores, 96 GB RAM, and one NVIDIA GTX 2080 Ti GPU (11GB). Each training run took around 4 hours.

B.2 DreamerV3

We use the official code base released by Hafner et al. [2025a]. We use the version with 12M trainable parameters, roughly the same size as the PPO model-free policy. We use the exact same default hyperparameters as the released repository (Table 7). We have 32 vectorized environments. Each training run was conducted on 33 CPU cores, 125 GB RAM, and one NVIDIA H100 GPU (80GB). Each training run took around 14 hours.

 https://doi.org/10.52202/085713-2125

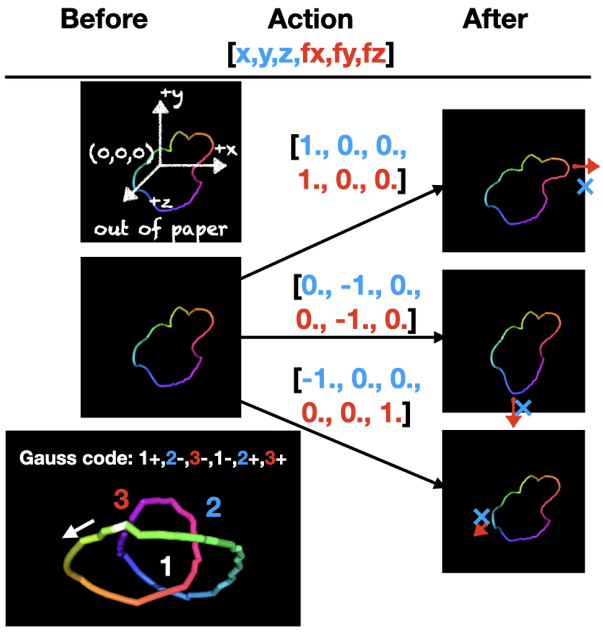


Figure 11: The first image of the user prompt. This image is used to introduce the task structure (together with text instructions), and to prime the model of system dynamics.

Table 6: PPO hyperparameters.

Name	Value	Comment
Learning rate	**1e-5**	Sweeping over {1e-5, 3e-5, 1e-4, 3e-4 (default)}.
Feature dimension	**768**	Sweeping over {512 (default), 768, 1024}
Batch size	64	Default
Number of epochs	10	Default
Gamma	0.99	Default
GAE lambda	0.95	Default
Clip range	0.2	Default

B.3 TD-MPC2

We use the official code base released by Hansen et al. [2024]. We take the default architecture with 5M trainable parameters. Although we experimented with policies as large as the 48M model, we did not observe significant improvements over the default 5M model on KNOTGYM. Table 8 lists hyperparameters.

We alter the convolutional encoder because the original code base only supports $[c, 64, 64]$ observations, while we need [3, 128, 256] for a fair comparison across methods. We added one additional pair of `nn.Conv2d(num_channels, num_channels, 5, stride=2), nn.ReLU(inplace=False)` after the existing `Conv2d` layer of kernel size 5, and one fully connected linear layer between the last flatten layer and the final activation layer.

With considerations of training wall time, we use the experimental vectorized_env branch that supports parallel environments. We use 16 parallel environments. Each training run was conducted on 16 CPU cores, 148 GB RAM, and one NVIDIA GTX 2080 Ti GPU (11GB). Each training run took up to 20 hours.

Table 7: DreamerV3 hyperparameters.

Name	Value	Comment
Learning rate	4e-5	Default
Train ratio	256	Default
Hidden layer size	256	Default
Number of classes	16	Default
Replay buffer size	5e6	Default

Table 8: TD-MPC2 hyperparameters.

Name	Value	Comment
Number of environments	**16**	Sweeping over { 1 (default), 2, 4, 8, 16, 32, 64}
Steps per update	**4**	Sweeping over { 1 (default), 2, 4, 8, 16}
ρ	**0.7**	{ 0.5 (default), 0.7 (suggested for episodic tasks)}
Learning rate	3e-4	Default
Batch size	256	Default
MPC	True	Default
Iterations	6	Default
Number of samples	512	Default
Number of elites	64	Default
Number of trajectories	24	Default
Horizon	3	Default

C KnotGym Implementation Details

The Environment KnotGym uses the MuJoCo physics simulator [Todorov et al., 2012]. Each knot is modelled as a chain of beads (rigid bodies). The knot "floats" in a viscous medium – a performance related design decision to reduce collision checking between the rope and the plane. The pixel observation is generated by MuJoCo's default renderer. The camera is set to track the center of mass of the knot from a fixed distance and orientation. We sample all knot configurations with a combination of periodic saving on a random policy and manual filtering for configurations that are too twisted thus not suitable to be goal exemplars. We normalize the action space by default. Table 9 lists the environment specifications.

Table 9: KnotGym environment specifications.

Name	Value
Max n crossings	One of {1,2,3,4}
Observation Space	Shape: [3, 128, 256], dtype: uint8
Action Space	Shape: [6], dtype: float 32, range: [-1,1]
Frame skip	24
Reward for Gauss code equality	+5
Punishment for timeout	-5
Max episodic steps	50
Reset noise scale	0.015

Software Credit We use the following software: Stable-baselines3 [Raffin et al., 2021], DreamerV3 [Hafner et al., 2025a], TD-MPC2 [Hansen et al., 2024], PyKnotId [Taylor and other SPOCK contributors, 2017], Gymnasium [Towers et al., 2024], and MuJoCo [Todorov et al., 2012].

Stable-baselines3 (`https://github.com/DLR-RM/stable-baselines3`) has MIT license.

DreamerV3 (`https://github.com/danijar/dreamerv3`) has MIT license.

TD-MPC2 (`https://github.com/nicklashansen/tdmpc2`) has MIT license.

PyKnotId (`https://github.com/SPOCKnots/pyknotid`) has MIT license.

https://doi.org/10.52202/085713-2125

Gymnasium (`https://github.com/Farama-Foundation/Gymnasium`) has MIT license.

MuJoCo (`https://github.com/google-deepmind/mujoco`) has Apache license 2.0.

D Additional Results

Model Selection We select the model closest to 1M environment steps for generalization analysis. How does training affect generalization? Figure 12 shows the generalization dynamics of a single `tie` #X=2 training run with 20 goal configurations. While the train success rate increases drastically, the success rate evaluated on the test split configurations with same #Xincreases only modestly. This suggests that the policy has difficulty generalizing to different goals even with the same level of complexity (same #X).

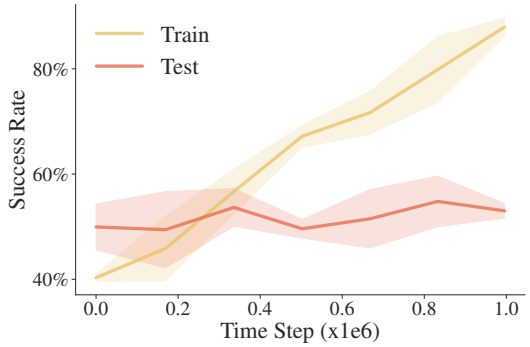


Figure 12: Generalization dynamics of a DreamerV3 `tie` #X=2 training run. Error shades are 95% confidence intervals across three seeds.

Training Curves Figure 13 shows all RL training curves in the first million steps. For the `unknot` task, all methods manage to learn, despite different sample efficiency. For the harder `tie` and `convert`, none of the methods surpass random baseline when #X>2, showcasing the learning challenges KnotGym presents to RL methods.

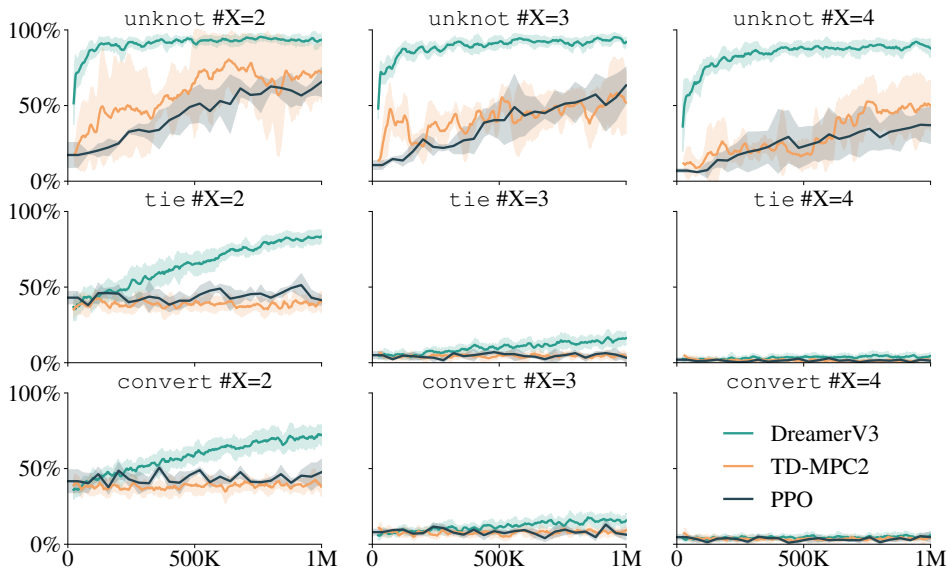


Figure 13: RL training curves: success rates on the train split versus number of steps across nine KnotGym tasks. Error shades are 95% CI computed over three seeds.