2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Learning Topological Motion Primitives for Knot Planning

Mengyuan Yan[1] and Gen Li[2] and Yilin Zhu[1] and Jeannette Bohg[1]

***Abstract*— In this paper, we approach the challenging problem of motion planning for knot tying. We propose a hierarchical approach in which the top layer produces a topological plan and the bottom layer translates this plan into continuous robot motion. The top layer decomposes a knotting task into sequences of abstract topological actions based on knot theory. The bottom layer translates each of these abstract actions into robot motion trajectories through learned topological motion primitives. To adapt each topological action to the specific rope geometry, the motion primitives take the observed rope configuration as input. We train the motion primitives by imitating human demonstrations and reinforcement learning in simulation. To generalize human demonstrations of simple knots into more complex knots, we observe similarities in the motion strategies of different topological actions and design the neural network structure to exploit such similarities. We demonstrate that our learned motion primitives can be used to efficiently generate motion plans for tying the overhand knot. The motion plan can then be executed on a real robot using visual tracking and Model Predictive Control. We also demonstrate that our learned motion primitives can be composed to tie a more complex pentagram-like knot despite being only trained on human demonstrations of simpler knots.**

I. Introduction

Autonomous manipulation of deformable objects has many potential applications, including robotic surgery, assistive dressing, textile and clothing manufacturing, etc. Tying knots with linear objects, e.g. ropes, is a common but challenging task in this research direction. Various types of knots are used in surgery or search and rescue. By teaching robots to accomplish these tasks, we could aid surgeons in saving more patients, and replace rescue task forces in high-risk environments. However, teaching robots to manipulate ropes and tie knots is hard. Ropes have a high-dimensional state space which makes visual perception challenging. They have a large action space and under-actuated dynamics, making common planning algorithms either non-applicable or computationally expensive. In addition, knot tying is a long-horizon task for which some local planners for deformable linear objects could fall into local minima [1, 2].

Knot theory and topology can help to guide motion planning for tying knots. For example, it can provide an abstract action skeleton, i.e. a sequence of topological actions, to decompose the long-horizon knotting task into shorter subtasks [3]. It can also help to prune out unpromising edges in tree-based or roadmap-based motion planning methods [4]. However, motion planning for each topological action remains challenging, and brute-force search for the physical robot motions is prohibitively expensive due to the high-dimensional action space and lack of intuitive cost functions to abstract topological goals.

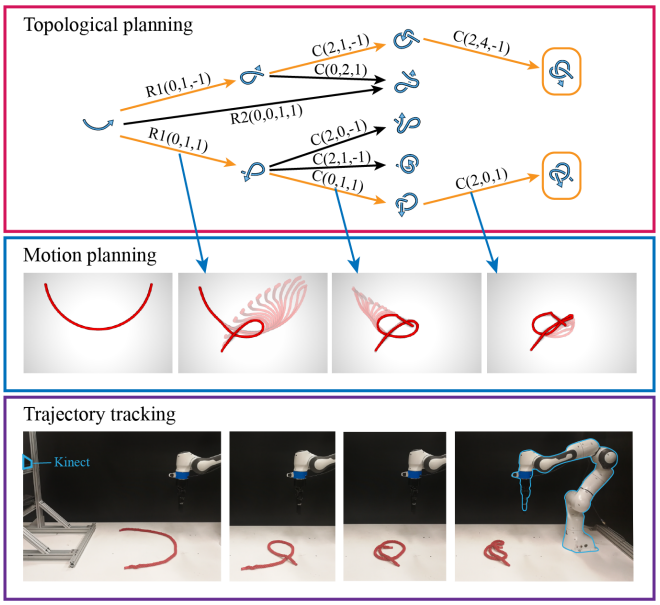


Fig. 1: The 3-level inference process for tying a knot. On the top level, we search for topological paths from the start topology to the goal topology in the graph defined by knot theory. On the second level, we ground each edge in the topological path, which is a topological action, into robot motion trajectories using trained motion primitives. Each motion primitive will predict a robot motion spline curve conditioned on the geometric configuration of the rope at the end of the previous stage, and the predicted motion is simulated to obtain rope configurations during the manipulation. On the bottom level, the simulated rope configurations will be used as reference states for MPC to track on the real robot.

Learning from demonstrations provides a promising alternative. van den Berg et al. [5], Berenson and Abbeel [6] learn to clone and enhance the robot trajectories for tying surgical knots from human teleoperation. However, the learned robot motion does not generalize to new string configurations. We try to benefit from both knot theory and data-driven methods, and propose to learn a library of topological motion primitives which ground abstract topological actions in robot motions and rope geometries. The motion primitives can be composed to plan for various knots. We design the learning algorithm to facilitate learning from few human demonstrations and generalize to new rope configurations and new knotting tasks.

Our key contributions are:

*Toyota Research Institute ("TRI") provided funds to assist the authors with their research but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity.

[1]Mengyuan Yan, Yilin Zhu and Jeannette Bohg are with School of Engineering, Stanford University, CA, USA, 94305 {mengyuan, ylzhu, bohg}@stanford.edu

[2]Gen Li is with Department of Computer Science and Technology, Tsinghua University, Beijing, China, 100084. This work was done while he was a research intern at Stanford University. lig16@mails.tsinghua.edu.cn

- We define *topological motion primitives*, which translate abstract topological actions into concrete robot motion trajectories, conditioned on input rope geometric configurations. They can be composed to solve a variety of knotting tasks.
- We encode topological actions such that each topological motion primitive can instantiate a range of them. Human demonstrations are only needed for some topological actions and the topological motion primitives learn to generalize to unseen rope shapes and topological actions through reinforcement learning.
- We demonstrate that the topological motion primitives can be composed to accomplish knot tying tasks on a real robot. We also demonstrate that we can plan for more complex knotting tasks than seen during training, which has not been shown in previous works.

II. Related Work

a) Planning for rope manipulation: Planning for rope manipulation is a challenging task because of its high dimensional state space and highly under-actuated dynamics. Moll and Kavraki [1], Tandon et al. [2] propose local planners for Deformable Linear Objects (DLOs) based on minimum energy curves. They can be used with tree-based and roadmap methods to search for longer plans. Although [1, 2] restrict the action space to the two ends of the rope, planning can take hours for a simple deformation task. For the long horizon and complex task of knotting, Saha and Isto [4] build search trees by sampling robot motions, and use knot theory to prune unpromising branches. The high computational cost for search-based methods call for the use of human knowledge to bias the search space. Wakamatsu et al. [3] design strategies for choosing grasping points and robot motions for each topological action step in knotting tasks, but only verified their strategy in an untangling task. Our work proposes to learn such strategies from human demonstrations, which we call *topological motion primitives*. By learning such primitives we generalize from very few human demonstrations to different geometrical and topological conditions, and greatly improve the efficiency of tree-based planning by intelligently biasing the sampling function.

b) Learning for rope manipulation: Several recent works have learned dynamics models for ropes and used them with *Model Predictive Control* (MPC) for manipulation tasks. Li et al. [7] and Battaglia et al. [8] model ropes as mass-spring systems, and use graph networks to learn rope dynamics. Ebert et al. [9] learn a video prediction model, without any physical concept of objects or dynamics. In follow-up works, Ebert et al. [10, 11] investigate different image losses. Han et al. [12] uses model-based reinforcement learning (RL) for deforming a rope in 2D with fixed start and goal configurations. While these works have demonstrated short-horizon deformation tasks, the method does not directly extend to knot planning, where the goal state is abstract, represented by topology. Pathak et al. [13] learn an inverse model that can predict robot actions for local deformation tasks, however humans are responsible for providing a visual plan that the inverse model will follow. Most related to our work, Wang et al. [14] tackles the long-horizon planning problem by embedding images into a plan-able latent space where neighboring points in the latent space correspond to images that are temporally close. Plans are generated using A* search in the latent space and decoded to observations. van den Berg et al. [5], Berenson and Abbeel [6] learn surgical knotting skills from human demonstrations, however only the robot end-effector trajectory is considered, and the state of the thread being manipulated is not included in the algorithm. Our work is different from the above works in the following aspects. (1) We address the long-horizon problem of knot tying, while [7, 8, 9, 12] only addresses the problem of deforming a rope to a goal shape. (2) We build on the previous work [15] which demonstrates estimation and tracking of rope states from images, so that planning and control can be done in state space instead of pixel space [9, 13, 14]. (3) We learn from human demonstrations and generalize to unseen situations by reinforcement learning, while previous works [7, 8, 9, 13, 14, 12] only use self-supervised robot exploration data and are unlikely to have seen successful multi-cross knots, and [5, 6] disregards the rope states and cannot generalize to new rope states. (4) Having access to the physical state of ropes, we use knot theory to decompose various knotting tasks into the same library of topological motion primitives, so that we can learn them with demonstrations on simpler knotting tasks, and compose the learned primitives to complete more complex knotting tasks.

III. Problem Statement

We define the task of robotic rope knotting as follows. The robot observes an untangled rope in its work space. The task is specified by a goal topological state, whose representation we will define in Sec. IV. The robot needs to find a plan of grasping points and motion trajectories that can bring the current rope configuration to a goal configuration with the desired goal topology, and execute the motion plan with visual tracking and MPC.

As shown in Fig. 1, we decompose the task of tying knots of user-specified topology into 3 levels. At the top level, knot theory defines a graph with topological states as nodes and topological actions as edges. Possible topological action sequences (topological plans) are obtained by graph search. At the second level, we translate each topological action into a motion trajectory of the robot and rope, conditioned on the observed configuration of the rope. We call the mapping functions *topological motion primitives*. Once the preceding topological action has been translated into a robot motion trajectory, the rope trajectory is obtained via simulation, and the end configuration serves as the start configuration for the subsequent topological action. At the bottom level, we execute the motion plan on the real robot. We use a visual tracking algorithm together with MPC to track the planned rope trajectory in a closed-loop manner, adjusting the robot motion commands to account for errors in rope dynamics and for noise in perception and actuation.

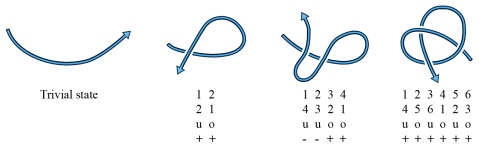
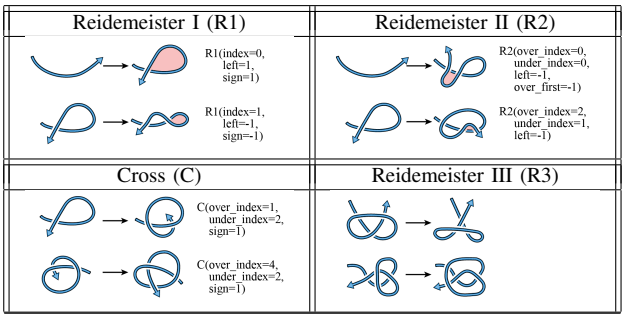


Fig. 2: Example geometric configurations and the representations of corresponding topological states. The arrow represents the direction from head to tail. "o"/"u" indicate the vertical order of each intersection (short for over/under), and "+"/"-" indicate the sign of each intersection.



index	For R1 only. Index of the segment forming the new intersection
over_index	For R2 and C. Index of the segment that will be over the new intersection(s).
under_index	For R2 and C. Index of the segment that will be under the new intersection(s).
left	For R1 and R2. Binary. Whether the new face (shaded) is on the left side of the top segment.
sign	For R1 and C. Binary. Sign of the new intersection.
over_first	For R2 if over_index = under_index. Binary. Whether the half segment closer to head will be over the new intersections.

TABLE I: Top: examples actions in each category and their parameter values. Bottom: Definitions of the parameters.

IV. Background on Topology

We follow the representation of a rope's topological state in [16]. A topological state is defined based on a 2D projection of the 3D curve. We use the horizontal plane in the world frame as the projection plane. Some examples of projected curves and their topological state representation are shown in Fig. 2. Given a projection, we pick one end of the rope as the head and the other end as the tail. As we trace the curve from head to tail, we number the intersections starting from 1. Each intersection will be encountered twice and therefore receives two numbers. Then, we retrace the curve, and when we encounter an intersection, record the two numbers from the current and the other intersecting strand, as well as a sign plus/minus and a relative vertical position (over/under) between the current and the other intersecting strand. A sign is determined by the following equation:

$$sign = \frac{\vec{l}_{over} \times \vec{l}_{under}}{|\vec{l}_{over} \times \vec{l}_{under}|} \cdot \vec{e_z}, \tag{1}$$

where l_{over} and l_{under} are the directional vectors for the two strands, and e_z is the unit normal of the projection plane.

To transition between the topological states, there are four categories of topological actions, and each category has many topological action instances, indexed by a few discrete parameters. Examples for each category are shown in Table I (top) and the definition of parameters are listed in Table I (bottom). Segments of the projected curve, separated by intersection points, are indexed starting from 0 when tracing the curve from head to tail.

- The Cross (C) action makes a new intersection using the head/tail segment with another segment.
- The Reidemeister I (R1) action makes a new loop using one segment of the curve.
- The Reidemeister II (R2) action makes two new intersections of opposite signs, by pulling the middle of one segment on top of another segment.
- The Reidemeister III (R3) action moves two neighboring intersections to the other side of a cross. We do not consider this more complex action category in our planning.

Note that these topological action categories are different from the grouping into topological motion primitives in Sec. V-A.

All topological states and actions form a directed acyclic graph (DAG), with the trivial topological state (i.e. untangled) as the root. Given a goal state, we can find one or more paths from the trivial topological state. Fig. 1 (top) shows a partial graph for the example of an overhand knot. The overhand knot has two topologies with opposite chirality. We show 2 of the 8 possible different topological paths.

V. Learning Topological Motion Primitives

Now we want to translate the abstract topological actions obtained from graph search into concrete robot motion trajectories and rope motion. The downstream closed-loop controller will use the resulting rope and robot trajectories as a reference.

We made two design choices in learning the topological motion primitives. First, the action space of the motion primitives will be long robot motion trajectories parameterized by spline curves, instead of small delta positions commonly used for feedback policies. This choice drastically shortens the horizon of the resulting RL problem so that the training process will be stable and more data-efficient, even with simple RL algorithms. Meanwhile, spline curves are complex enough to allow successful knotting with only a few re-grasps. Second, each topological motion primitive needs to work for a group of topological actions in order to scale to more complex knotting tasks. Thus we encode the discrete topological action parameters, listed in Table I, as an input to the motion primitives. The encoding is designed to facilitate learning of the similarities among motion policies that humans demonstrate for different topological actions.

We want the topological motion primitives to learn distributions of successful motion splines, conditioned on the current rope shape and the topological action to instantiate. Samples can be drawn from the learned distribution and used in tree-search during inference. Each spline curve has 3 control points. The first control point will be on the rope and serves as the grasping point. The last control point will be on the supporting table plane so that the robot gently releases the rope instead of dropping. The middle point can be above the table plane and determines the maximum height of the spline. This formulation leads to a 6D action space.

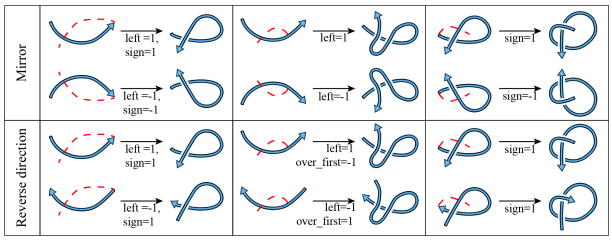


Fig. 3: Examples applying the *Mirror* or *Reverse* transformation on the geometric configurations and predicted robot motion splines. From Left to Right: Examples for the R1, R2 and Cross category. Applying mirror transformation to rope shapes and robot trajectories will change the corresponding topological action to have the opposite "sign" and "left" parameters. Reversing the rope geometries will change the corresponding topological action to have the opposite "left" and "over_first" parameters.

We observe that in most cases one motion spline is enough to accomplish a topological action. For the cases where regrasping is necessary, we expect that sequencing a few such spline curves could solve the problem. However, we leave that to future work.

A. Two types of similarities in motion policies

In this section, we will introduce two types of similarities in motion policies for different topological actions, to help the topological motion primitives generalize. Based on these similarities, we group all topological actions in the R1, R2 and Cross category into 4 topological motion primitives:

Set 1: All R1 actions.
Set 2: All R2 actions.
Set 3: Cross actions where the top segment is one end.
Set 4: Other Cross actions.

Cross actions are split into 2 sets based on the similarities in grasping position on the rope.

The first type of similarity is based on spatial symmetries in state transitions. The symmetries allow to reduce the number of topological actions that need to be learned. The topological actions that are not directly learned can be instantiated by using their learned counterparts, and applying geometric transformations to the rope configurations and predicted splines. Fig. 3 shows examples for the two transformations: *Mirror* and *Reverse*. Each cell in the figure shows a pair of transitions (s, a, s') before (top in each cell) and after (bottom in each cell) applying the indicated transformation. Here s and s' refer to the geometric rope configuration and a is the robot motion spline shown as red dashed curves. Each transition is also described, in the topological level, as a transition $(s_{\text{topo}}, a_{\text{topo}}, s'_{\text{topo}})$, and binary-valued parameters for the topological action a_{topo} are shown below the arrows. The *Mirror* transformation reflects the rope geometries and the robot trajectory about the horizontal axis (in fact, reflecting about any line in the plane has the same effect, but we choose the horizontal axis in our implementation). The corresponding topological actions before and after this transformation will have opposite "left" and "sign" parameters, but the other parameters will be the same. The *Reverse* transformation switches the head and tail of the rope, but keeps the robot trajectory unchanged. The corresponding topological actions before and after this transformation will have opposite "left" and "over_first" parameters, segment indices will also change. Either transformation, when applied twice, will be identity and produce no change.

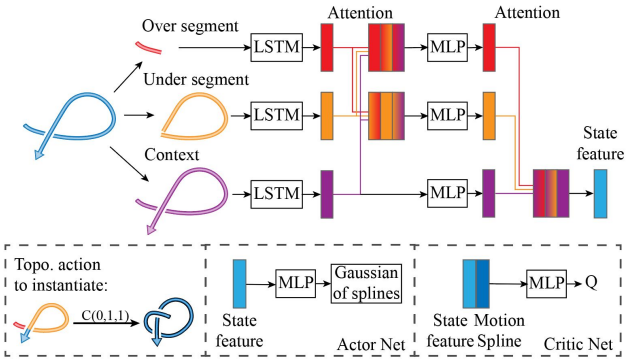


Fig. 4: Illustration of input encoding and network structure for topological motion primitives. Top: The trunk network for both the actor and critic network. The rope's geometric configuration is encoded in 3 parts according to the topological action to be instantiated, which is illustrated on the bottom left. The rectangles with color gradients represent the output features of attention modules [17], the color towards the left (right) edge of the rectangle is the color of the query (value) feature to the attention module respectively. The LSTM and MLP for each stream do not share weights. Bottom middle and right: prediction for actor network and critic network, based on the feature extracted from the trunk network.

As a result, it is sufficient to learn, for example, the topological action R1(index=0, left=1, sign=1), and we can use the same network to infer robot motion splines for R1(index=0, left=$\pm$1, sign=$\pm$1). For example, to infer the motion spline for R1(index=0, left=-1, sign=-1), we mirror the start state, feed the transformed state into the network to predict spline parameters, and then apply the inverse transformation, i.e. mirroring, to the predicted spline parameters. To infer the motion spline for R1(0, -1, 1), we use the *Reverse* transform instead, and for R1(0, 1, -1) we use the combination of both transformations.

The second type of similarity is more conceptual. When making a new intersection, humans will grasp the segment that will be over the newly formed intersection, and move towards the segment that will be underneath. This common strategy is captured by encoding the rope's geometric configuration into three parts: The segment which will be over the newly formed intersection, the segment which will be underneath, and the whole state which acts as task context to e.g. avoid undesired self-collision. For R1 actions, the over and under segment will be the same. The network structure to process this input is shown in Fig. 4. Pairwise attention modules [17] are used to propagate information between the three streams.

B. Training and inference process

We will train the topological motion primitives in two stages: (i) reinforcement learning for single-step tasks, where each motion primitive is trained independently and demonstration data is used to bias initial exploration, and (ii) reinforcement learning for multi-step knotting tasks, where the motion primitives are sequenced to form the policy and are trained jointly.

a) Collecting demonstration: With our action space formulation, demonstration is simply given by clicking on three points on an image of the start geometric configuration,

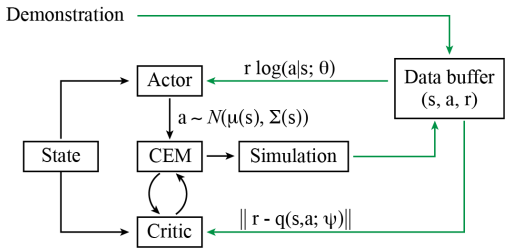


Fig. 5: Illustration of the training and inference process for motion primitives. States s are geometric configurations of the rope. Actions a are 6D vectors parameterizing spline curves. Rewards r are binary, indicating success of the desired topological action. The critic network with weights ψ predicts Q values and the actor network with weights θ predicts the mean and variance μ, Σ for Gaussian action distributions. The black arrows constitutes the policy during inference, as well as the behavior policy during training. Green arrows indicate data flow and objectives during training.

indicating the position of the three control points of the spline. Gaussian noise is added to the annotation to generate sample motion splines, and the motions are simulated to verify if the desired topological action is accomplished. All the trials are saved and we refer to them as demonstration data.

b) Independent training on single-step tasks: We adapt QT-Opt [18] to train each topological motion primitive. The training process is summarized in Fig. 5. QT-Opt trains a critic neural network to approximate the Q function, and uses the *Cross-Entropy Method* (CEM) to find the action with maximum Q value as the policy. However, we find this to be in-sufficient for our high dimensional action space. Thus, we also train an actor network to predict Gaussian distributions conditioned on the start configuration of the rope to initialize the CEM process. The actor network is trained with REINFORCE [19] with data from the replay buffer. Although other methods such as Trust Region Policy Optimization or Proximal Policy Optimization can be used, we expect them to bring little benefit, since with our formulation of the action space, the resulting RL problem has a short horizon. The replay buffer is initially populated with demonstration data, and this training stage will be referred to as imitation learning. New data are added to the replay buffer once imitation learning has converged, and this training stage will be referred to as reinforcement learning.

c) Joint training on multi-step tasks: There are infinitely many geometric configurations that correctly instantiate a topology. Some configurations are more favorable than others because they make the next action easier. It is hard to engineer a continuous reward function which can bias the motion primitives towards generating more favorable end states. Luckily, such a reward function is available once we trained the library of motion primitives. Let us say that the topological path found for a task is $(s_{\text{topo}}, a_{\text{topo}}, s'_{\text{topo}}, a'_{\text{topo}}, s''_{\text{topo}})$, and geometric instantiations of this topological path is (s, a, s', a', s''). The state value function $V_{a'_{\text{topo}}}(s') = \max_{a'} Q_{a'_{\text{topo}}}(s', a')$ is the predicted success rate of a'_{topo} starting from configuration s'. This value thus is a reward for a_{topo}, $r_{a_{\text{topo}}}(s, a)$. Based on this observation, we finetune the topological motion primitives jointly on multi-step knotting tasks, where the RL horizon is set to the number of topological actions in the path found during topological graph search. The policy only receives a reward of 1 at the end of the episodes if the desired knot topology is achieved. Different from a normal RL problem, different motion primitives are used for each time step in an episode according to the topological plan, and each motion primitive receives different transitions from the replay buffer as training data. The same adapted QT-Opt algorithm is used for this training process.

d) Grounding the topological path: tree search: Once the four topological motion primitives are trained, we use them as action samplers in tree search, to guarantee finding a motion plan. From the top level topological planning, we have found one or more topological paths from the untangled state to the given knot topology. The paths form a DAG, which we call the solution DAG. We start from the geometric configuration of the untangled state as the tree root. At each tree expansion step, we randomly select a node in the tree (including both leaf nodes and non-leaf ones), biasing the selection probability towards nodes whose topological states are closer to the knot topology in the solution DAG (measured by the number of edges on the shortest path in the solution DAG). From the selected node, we randomly select an outgoing edge in the solution DAG as the next topological action to instantiate. We use the corresponding topological motion primitive to predict a motion spline for the robot, and this spline is executed in simulation to obtain the rope motion trajectory. Note that the predicted spline is not guaranteed to reach the desired rope topology prescribed by a topological action. Therefore, the rope's end configuration together with its topology is only added to the search tree if the goal topology of the topological action is reached. The tree keeps expanding until the final knot topology is reached.

VI. Closed-loop control and visual tracking

To execute the planned robot motion on the real robot, we sample way points from the robot motion spline, and record corresponding rope geometric configurations from the simulator. We use the rope motion trajectories as reference states for MPPI [20] that tracks this rope motion. The planned robot motion spline is used to initialize the actions used in MPPI. We use the perception network in [15] to estimate the initial rope state, and use the image space loss and LSTM dynamics model in [15] to track the 3D rope state across time. We extended the differentiable rendering of rope states to 3D, by additionally taking the Kinect depth images.

VII. Experiments

We evaluate four aspects of our proposed method. First, we quantitatively verify two hypotheses underlying our method: using human demonstrations to bias the search for robot motion splines can drastically reduce the number of samples compared to brute-force search, and using reinforcement learning on a wider range of start configurations improves generalization compared to only using imitation learning. Second, we show ablation studies that validate two design choices: using both the actor and critic networks for learning the motion primitives, and finetuning the motion primitives

jointly on multi-stage knotting tasks. Both decisions improve success rates of the motion primitives and reduce search costs. Third, we compare our method to Causal InfoGAN [14] as a baseline on knot planning. Finally, we qualitatively demonstrate that our generated plan can be executed on a real robot, and our method can plan for a knot more complex than seen during training.

A. Value of using demonstrations and learning

1) Data: We evaluate on eight topological actions, listed in Fig. 6. They are visualized in Fig. 1 (Top). For R1(0,1,1) and R2(0,0,1,1), only one start configuration is demonstrated, which is a straight line. For each demonstration, 480 splines are generated by adding Gaussian noise, and simulated. For each of the other six topological actions, we demonstrate on eight start configurations and simulate 320 trials from each configuration. All the trials are saved as demonstration data. The eight topological actions are categorized into the four topological motion primitives as described in Sec. V-A. Thus, the demonstration data is also separated into four buffers to train four pairs (actor and critic) of networks. The categorization is reflected by vertical lines in Fig. 6.

2) Baseline: Our first baseline is to randomly sample a motion spline from the whole parameter space. Our second and third baselines use the demonstration data but no further reinforcement learning. For the second baseline, generalization is achieved by interpolating demonstrations. For each demonstrated configuration, we fit a Gaussian distribution to all successful spline parameters. For new configurations, the *Iterative Closest Point* (ICP) distance to each demonstrated configuration is calculated. we use the exponential of negative ICP distances as the weights to interpolate the Gaussian parameters. Splines are sampled from the interpolated Gaussian distributions to evaluate the success rate. For third baseline, we train our topological motion primitives with imitation learning only.

3) Single-stage evaluations: For each topological action we evaluate the success rate of predicted spline samples for an unseen set of geometric configurations, but of the same topology as seen during training. The results are shown in Fig. 6. Comparing imitation-only motion primitives to brute force search, there are up to 100x improvements in success rates, thus reduction in search time, by using human demonstrations. The imitation-only motion primitives also have higher success rates than ICP interpolation on most topological actions, especially R1(0,1,1) and R2(0,0,1,1), suggesting the advantage of our neural network compared to hand-designed methods. By further training the motion primitives with reinforcement learning on a wider range of rope shapes, we improve the success rate by 1.14x to 2.33x compared to imitation-only.

4) Multi-stage knotting evaluations: We also evaluate the number of branches in the search tree when planning for an overhand knot. For simplicity we only follow one topological path instead of all possible paths. This path involves a sequence of 3 topological actions, R1(0,1,1), C(0,1,1) and C(2,0,1). We use two start configurations and repeat 10

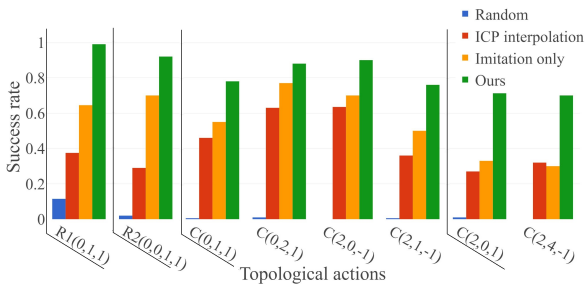


Fig. 6: Success rate of predicted spline samples for eight topological actions, using a random sampler, the ICP interpolation of human demonstrations, our motion primitive networks only trained with imitation learning, and our motion primitives trained with imitation and reinforcement learning (Ours). Black vertical lines indicate categorization of topological actions into the four motion primitives.

experiments for each configuration, with different random seed: (i) the start configuration is a straight line, which has been demonstrated when training the motion primitives, and (ii) the start configuration is an arc curve and generalization is necessary. When using the ICP interpolation, the tree size shows significant variance. Although the smallest trees only have 3 branches, which is the theoretical lower limit, the largest tree has 190 branches for the straight configuration and 1428 branches for the arc. When using the imitation-only motion primitives, the largest tree has 43 branches for the straight configuration and 9 branches for the arc. When using our motion primitives trained with both imitation learning and RL, the tree size is kept below 5 for all 10 seeds, and reaches the lower limit of 3 branches more than half of the time. This greatly reduced search time demonstrated our method's superior generalization to new geometries.

B. Ablation studies

In this section we evaluate two design choices: (1) Using both actor and critic networks for learning topological motion primitives, as compared to only using one of the network, and (2) finetuning the topological motion primitives jointly on the overhand knotting task.

1) Benefit of using actor-critic: We evaluate the success rate on the same sets of topological actions and rope configurations as used in Sec. VII-A, and report the results in Fig. 7. When training the critic only, CEM is initialized with a large Gaussian covering the whole space of spline parameters, and is run for 10 iterations to find the best action. When training the actor only, we take samples from the predicted Gaussian distribution. When training both networks, the actor prediction is used to initialize the CEM, and CEM is run for only 1 iteration.

From Fig 7, training with both actor and critic networks achieves the best results for 7 out of 8 topological actions. The advantage of training both networks, over using the critic only, confirms that CEM is not effective enough in finding maxima in a high dimensional action space, which is also noted in [21]. The difference in success rate is particularly big for topological actions C(2,0,1) and C(2,4,-1), which require high accuracy robot motion. Using both actor and critic also performs better than training the actor alone, possibly for two reasons: (1) Gaussian distributions

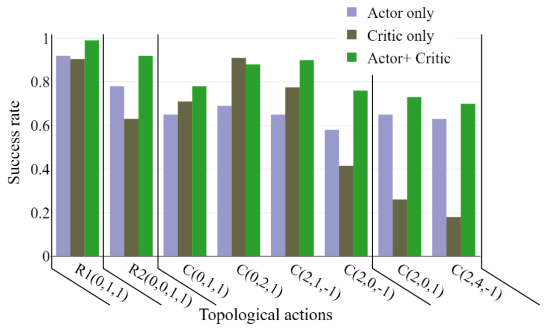


Fig. 7: Success rate of predicted spline samples for 8 topological actions, using topological motion primitives trained using actor network only, critic network only, and both networks together.

are not expressive enough to approximate the successful spline parameter distribution. (2) REINFORCE is an on-policy RL algorithm, but we are using it in our off-policy setting with replay buffers, because we can only afford a small amount of interaction with simulation. Therefore the actor is overly optimistic near the boundary of successful and failing spline parameters. We have experimented with correcting this bias with importance sampling, but could not stabilize the training.

2) Benefit of joint finetuning: We repeat the planning process from randomly generated untangled start configurations to an overhand knot and evaluated the number of evaluated branches. When using the motion primitives trained independently, the number of evaluated branches is 4.3 ± 2.6 over 50 experiments. After finetuning, the number drops to 3.4 ± 0.6, showing notable improvement, especially for the worst cases.

C. Comparison to Causal InfoGAN

We compare our method to using Causal InfoGAN [22] on the same tasks of planning for single-stage topological actions and the multi-stage overhand knot. We modified open source code [22] to use the configurations of the rope instead of images as observations. Causal InfoGAN learns to embed the rope configurations into a latent space where a linear-Gaussian stochastic transition model is prescribed. When planning a path between given start/goal configurations, the algorithm first project the given configurations into the latent space, i.e. search for a point in the latent space that maps to a configuration closest to the given start/goal, then interpolate the straight line in the latent space from the projected start point to the projected goal. The interpolation points are mapped to rope configurations as way points that constitute a plan. We trained the network using the demonstration data for only one topological action, C(0,1,1). We also tried to train the network on all demonstration data but cannot get visually plausible results, due to commonly known training instabilities of GANs.

We show visualization of the generated plans from the single-stage model in Fig. 8. The top two rows are from the training data and the bottom two rows are from the same evaluation set of rope configurations used in previous experiments. Within each row, the first and last images are the given start and end configurations, and the second and

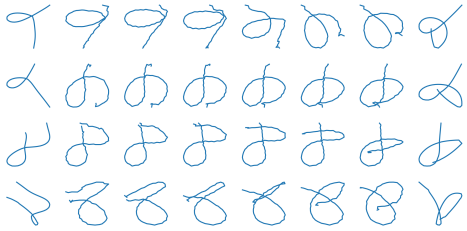


Fig. 8: Visual plan generated by Causal InfoGAN. This model is trained on demonstration data for only one topological action, C(0,1,1). The plans are visually plausible, although the bottom two lines' motion strategy are different from the demonstration data.

second last are the projected configurations. Different from our method, Causal InfoGAN requires the end configuration to be given, and we guarantee there is a feasible path for the start and end configurations shown in Fig. 8. The distances from the given start and end configurations to their projected configurations are quite large. In the first row of Fig. 8, the projected end configuration has a different topological state than the given one. Between the projected start and end configurations, the generated visual plans look plausible, although the bottom two rows adopted a motion strategy different from demonstration. The manipulation progress also seems to be uneven, e.g. middle columns in the first row have larger shape changes.

D. Plan execution on real robot

We show in the supplementary video that plans generated by our motion primitives can be executed on the real robot, with the help of visual tracking and MPC. Snapshots during the execution process are shown in the bottom row of Fig. 1.

E. Generalization to more complex tasks

We show that the topological motion primitives trained with the 8 topological actions above, where no more than three intersections are present in the rope states, can be used to find a plan for the more complex task of tying a pentagram-like knot starting from the overhand knot. The rope configurations (rainbow-colored solid lines) and the predicted robot motion splines (green dashed lines) are visualized in Fig. 9. This plan is found when only 12 branches are evaluated in the tree. We remark that the start configuration for this more complex task is different from the ones generated in the overhand knotting task, e.g. the one shown on the bottom-right corner of Fig. 1. We manually pulled out the head segment in order to demonstrate the more complex pentagram-like knot task. The trained topological motion primitives have not learned such pulling behavior, since such behaviors are not required in the overhand knot and not present in the demonstration data. We leave this to future work.

VIII. Conclusion

We propose a 3-level motion planning and control algorithm for knot tying. At the top level, knot theory decomposes knotting tasks into sequences of abstract topological actions. The set of topological actions is shared across all knotting tasks. Our key contribution is at the second level, which is a set of topological motion primitives that generates

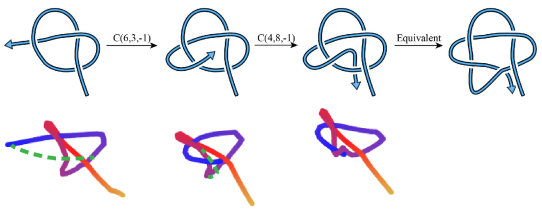


Fig. 9: Topological path to a pentagram-like knot (top), and corresponding motion plan (bottom). Rope configurations are shown with color gradients, with blue as heads and orange as tails. Robot motion splines are shown in green.

robot motion trajectories for each topological action, conditioned on the observed rope configurations. At the bottom level, the predicted robot motion trajectories are simulated, and MPC tracks the resulting rope trajectory on real robots. We train the topological motion primitives by imitation and reinforcement learning. To generalize human demonstrations of simple knots into more complex knots, we observe similarities in the motion strategies of different topological actions, and designed the network architecture accordingly to learn such similarities. We demonstrate that our learned motion primitives have significantly higher success rate in completing the required topological actions, compared to baseline methods. When the motion primitives are used as samplers in a tree-search framework, a much smaller tree is grown before a valid motion plan is found. We verified that the generated robot motion plan can be executed on a real robot using visual tracking and MPC. We also demonstrated that our learned motion primitives can plan for a more complex pentagram-like knot, even though the human demonstrations are only on simpler tasks. This suggest that it is possible to scale the learned motion primitives to increasingly more complex knotting tasks without human intervention.

There may be several challenges in extending this work to a wide range of object types and knot types. Although we are confident this method applies to ropes with varying thickness and length, other object types such as cables, may be very stiff such that they return to their original shapes when released from the gripper. Very thin threads such as surgical suture may be hard to observe from depth images thus making 3D perception hard. To extend to more complex knot types, the robot would need to learn additional skills, such as pulling one segment from underneath existing intersections, or rearranging the rope to make next steps easier. The former may also require two robot arms to cooperate. These challenges may be addressed by some changes to the method. For example, by allowing each motion primitive to predict more than one robot motion spline when necessary, and providing demonstrations of re-arranging or pulling. To use multiple arms, the actions space may be extended. For example to acquire the pulling skill, the motion primitive could additionally predict an end-effector pose for the second arm to hold one point of the rope, while the first arm pulls another segment.

REFERENCES

[1] M. Moll and L. E. Kavraki, "Path planning for deformable linear objects," *IEEE Transactions on Robotics*, vol. 22, pp. 625–636, Aug 2006.

[2] S. Tandon, S. Javdani, J. F. O'Brien, and P. Abbeel, "Motion planning for deformable one-dimensional objects."

[3] H. Wakamatsu, A. Tsumaya, E. Arai, and S. Hirai, "Planning of one-handed knotting/raveling manipulation of linear objects," *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, pp. 1719–1725, 2004.

[4] M. Saha and P. Isto, "Manipulation planning for deformable linear objects," *IEEE Transactions on Robotics*, vol. 23, pp. 1141 – 1150, Jan. 2008.

[5] J. van den Berg, S. Miller, D. Duckworth, H. Hu, A. Wan, X. Fu, K. Goldberg, and P. Abbeel, "Superhuman performance of surgical tasks by robots using iterative learning from human-guided demonstrations," in *IEEE International Conference on Robotics and Automation (ICRA)*, May 2010, pp. 2074–2081.

[6] S. T. A. L. D. Berenson and J. F. P. Abbeel, "Tying surgical knots from demonstration: Enhancing demonstrations and correcting errors during execution."

[7] Y. Li, J. Wu, J.-Y. Zhu, J. B. Tenenbaum, A. Torralba, and R. Tedrake, "Propagation networks for model-based control under partial observation," *IEEE International Conference on Robotics and Automation (ICRA)*, May 2019.

[8] P. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, and K. kavukcuoglu, "Interaction networks for learning about objects, relations and physics," *International Conference on Neural Information Processing Systems (NIPS)*, Dec. 2016.

[9] F. Ebert, C. Finn, S. Dasari, A. Xie, A. X. Lee, and S. Levine, "Visual foresight: Model-based deep reinforcement learning for vision-based robotic control," *arXiv: 1812.00568*, 2018.

[10] F. Ebert, C. Finn, A. X. Lee, and S. Levine, "Self-supervised visual planning with temporal skip connections," *Conference on Robot Learning*, pp. 344–356, 2017.

[11] F. Ebert, S. Dasari, A. X. Lee, S. Levine, and C. Finn, "Robustness via retrying: Closed-loop robotic manipulation with self-supervised learning," *Conference on Robot Learning*, vol. 87, pp. 983–993, 2018.

[12] H. Han, G. Paul, and T. Matsubara, "Model-based reinforcement learning approach for deformable linear object manipulation," in *13th IEEE Conference on Automation Science and Engineering (CASE)*, 2017, pp. 750–755.

[13] D. Pathak, P. Mahmoudieh, M. Luo, P. Agrawal, D. Chen, F. Shentu, E. Shelhamer, J. Malik, A. A. Efros, and T. Darrell, "Zero-shot visual imitation," in *International Conference on Learning Representations*, 2018.

[14] A. Wang, T. Kurutach, A. Tamar, and P. Abbeel, "Learning robotic manipulation through visual planning and acting," in *Robotics: Science and systems*, 2019.

[15] M. Yan, Y. Zhu, N. Jin, and J. Bohg, "Self-supervised learning of state estimation for manipulating deformable linear objects," *International Conference on Robotics and Automation(ICRA)*, 2020.

[16] T. Morita, J. Takamatsu, K. Ogawara, H. Kimura, and K. Ikeuchi, "Knot planning from observation," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 3, Sep. 2003, pp. 3887–3892.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *International Conference on Neural Information Processing Systems (NIPS)*, Dec. 2017, pp. 5998–6008.

[18] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Proceedings of The 2nd Conference on Robot Learning*, Oct 2018, pp. 651–673.

[19] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, no. 3–4, p. 229–256, May 1992.

[20] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1433–1440, May 2016.

[21] M. Yan, A. Li, M. Kalakrishnan, and P. Pastor, "Learning probabilistic multi-modal actor models for vision-based robotic grasping," *International Conference on Robotics and Automation (ICRA)*, May 2019.

[22] T. Kurutach, A. Tamar, G. Yang, S. Russell, and P. Abbeel, "Learning plannable representations with causal infogan," *International Conference on Neural Information Processing Systems (NIPS)*, p. 8747–8758, Dec. 2018.