The Grasp Loop Signature: A Topological Representation for Manipulation Planning with Ropes and Cables

Peter Mitrano[1] and Dmitry Berenson[1]

***Abstract*— This paper studies robotic manipulation of deformable, one-dimensional objects (DOOs) like ropes or cables, which has important potential applications in manufacturing, agriculture, and surgery. In such environments, the task may involve threading through or avoiding becoming tangled with other objects. Grasping with multiple grippers can create closed loops between the robot and DOO, and if an obstacle lies within this loop, it may be impossible to reach the goal. However, prior work has only considered the topology of the DOO in isolation, ignoring the arms that are manipulating it. Searching over possible grasps to accomplish the task without considering such topological information is very inefficient, as many grasps will not lead to progress on the task due to topological constraints. Therefore, we propose the $\mathcal{G}_L$-signature which categorizes the topology of these grasp loops and show how it can be used to guide planning. We perform experiments in simulation on two DOO manipulation tasks to show that using the $\mathcal{G}_L$-signature is faster and more successful than methods that rely on local geometry or additional finite-horizon planning. Finally, we demonstrate using the $\mathcal{G}_L$-signature in a real-world dual-arm cable manipulation task.**

I. Introduction

Manipulation planning for deformable one-dimensional objects (DOOs) like ropes and cables is challenging due to the high-dimensional state representation of these objects and the cost of simulating their motion. Furthermore, most tasks benefit from multiple arms to control DOO shape and avoid becoming tangled with the environment. Therefore, the planner needs to consider the DOO, the arms manipulating it, and the environment. A task and motion planning (TAMP) approach to this problem would decompose planning into a grasp selection problem and a motion planning problem for the DOO given a specific grasp, as in [1]–[3]. However, the DOO planning problems are often expensive to solve. To reduce the space of grasps we need to search, we borrow the idea of a *signature* from the field of topology.

To explain what this signature represents, consider how the robot should grasp the tip of the cable in Figure 1. By grasping we form a loop, which we call a *grasp loop* and show as blue and red dashed lines in Figure 1. It is possible to grasp either around the left side or the right side of the frame, but these two grasps are categorically different in that we cannot smoothly deform from one to the other without breaking the grasp or the frame. The frame also forms a loop, called an obstacle loop (S_1). When grasping from the left (red), these two loops are linked, but when grasping from the right (blue) they are not. Our key insight is that the robot, DOO, and environment form a *graph of grasp loops* and we can use this graph to construct a signature, $\mathcal{G}_L$-signature, which captures topological information relevant for planning. To be clear, we do not address knots in the DOO. Our work is complimentary to work on tying or untying knots [4]–[7].

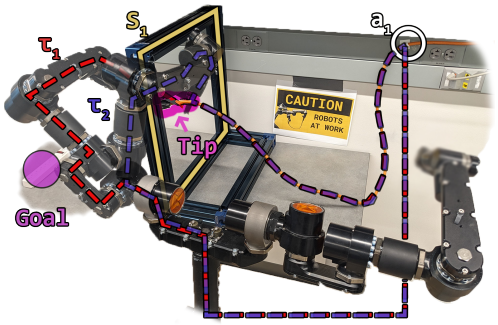

Fig. 1: Annotated image of our real world cable threading setup. The red dashed line shows a grasp loop τ_1 that is linked with the skeleton S_1. The blue grasp loop is not linked with S_1. This distinction is captured by the proposed $\mathcal{G}_L$-signature and is used in planning.

The main contribution of this paper is the $\mathcal{G}_L$-signature which compactly represents the topological relationship between the object, the arms manipulating it, and the environment. We demonstrate that this signature is useful for manipulation planning with DOOs. In simulation, we show that methods using the $\mathcal{G}_L$-signature outperform baselines and ablations which search for grasps without using topological information. Finally, we demonstrate a threading and point reaching task on a physical robot. Videos and animations can be found on our Project Website[1].

II. Related Work

Topology in Motion Planning: Topology and homotopy have been used in path planning for flying and driving robots [8], [9], as well as tethered robots [10]. [10] operates only in 2D and [11] considers an approximation of homotopy for 3D path planning. [8] introduces a simple-to-compute and exact signature for characterizing the homotopy of 3D paths with respect to 3D obstacles with holes in them, called the h-signature. We build on [8] to define our $\mathcal{G}_L$-signature.

Manipulation Planning for Deformable Objects: Prior work on knot tying and untying has also applied knot theory to DOO manipulation [4]–[7], [12]–[14]. These methods use planar crossing representations, which project a curve

This work was supported in part by Toyota Research Institute, the Office of Naval Research Grant N00014-21-1-2118, and NSF grants IIS-1750489, IIS-2113401, and IIS-2220876.[1]Department of Robotics, University of Michigan

[1]https://sites.google.com/view/doo-manipulation-signature/home

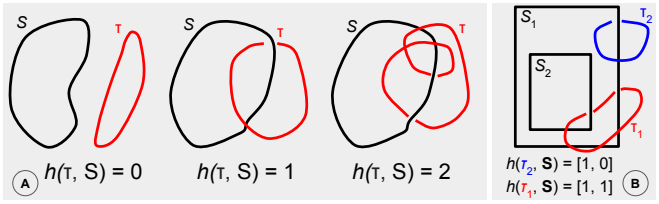

Fig. 2: (A) Illustration of the h-signature for a loop representing the robot and DOO (red/blue) and a loop representing an obstacle (black). (B) Two examples of the h-signature for a skeleton with two obstacle loops S_1 and S_2.

into a specified plane and count the sequence and type of crossings. [5] used this method for robotic knot tying, and extended this to tying around obstacles by specifying connections between obstacles and the DOO. However, these methods do not consider how the manipulator effects the topology, and fail in some cases with non-planar obstacles. A method for threading surgical needles was proposed in [15], but uses floating grippers and does not address planning for the robots' arms or obstacles, and is limited to tight-tolerance insertion tasks. [16]–[20] all address manipulation planning for DOOs assuming the grasp is fixed, which is complementary to this work.

Grasping and Regrasping: With rigid grasping, the challenge is primarily in achieving a stable grasp [21], [22]. With deformables, the challenge is where to grasp, as studied in cloth smoothing or folding [23]–[25]. These works use pick-and-place primitives with a single manipulator, which is too restrictive for many tasks. In contrast, we use a dual-arm manipulator and use joint velocities as our action space. In [26], a dual arm manipulator autonomously dresses a mannequin. Their method for grasp planning is based on learned visual models of the garment, and only considers grasps near keypoints such as the elbow or shoulder. The methods in [1]–[3] plan grasps on the DOO, but they assume the rope is planar (flat on a table), use one manipulator, and do not consider obstacles for the manipulator. [27] describes a method that produces pick, place, and sliding paths in the configuration space without explicit task planning. However, this method does not address underactuated kinodynamic systems such as DOOs.

III. Defining the $\mathcal{G}_L$-signature

A. Preliminaries

We primarily use notation that is consistent with [8]. We call a closed one-dimensional curve in 3D a *loop*. The environment is assumed to be decomposed into a *skeleton* made up of multiple *obstacle loops* $\mathbf{S} = \{S_1, \ldots, S_n\}$. Each obstacle loop is made up of line segments $S_i = \{\mathbf{s}_i^1, \ldots, \mathbf{s}_i^{n_i}\}$. In practice, the skeleton can either be specified manually or computed automatically from a medial axis transform of a mesh or pointcloud of the environment. The state $s = (q, o)$ contains the robot state q (joint angles) and the DOO state o (ordered list of points in $\mathbb{R}^3$).

[8] plans paths that are in a given homotopy class or avoid a certain homotopy class. They compare two paths by considering the homotopy class of the closed loop τ formed

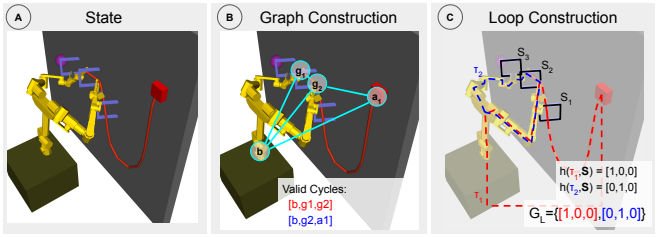

Fig. 3: The process of constructing the $\mathcal{G}_L$-signature. (C) There are 2 grasp loops and 3 object loops, so the $\mathcal{G}_L$-signature is a set with two elements, and each element is a vector of 3 non-negative integers.

by joining the two paths at their shared start and end points. For a path loop τ and an obstacle loop S, they define the h-signature $h(\tau, S) \in \mathbb{Z}$, which counts the number of times τ passes through S. The sign of h in this case is determined by the direction of τ. The h-signature can be extended to a list of the h-signatures with respect to each obstacle loop in the skeleton $h(\tau, \mathbf{S})) = [h(\tau, S_1), \ldots, h(\tau, S_n)]$. These cases are illustrated in Figure 2. The equation for computing $h(\tau, S)$ is reproduced from [8]. The point $\mathbf{s}_i^{j'}$ is the point that follows $\mathbf{s}_i^j$, and r is a point on the loop τ. The integration over τ is done numerically.

$$h(\tau, S) = \frac{1}{4\pi} \int_\tau \sum_{j=1}^{n_i} \Phi(\mathbf{s}_i^j, \mathbf{s}_i^{j'}, r) \Delta r$$

$$\Phi(\mathbf{s}_i^j, \mathbf{s}_i^{j'}, r) = \frac{1}{||d||^2} \left(\frac{d \times p'}{||p'||} - \frac{d \times p}{||p||} \right) \quad (1)$$

$$p = \mathbf{s}_i^j - r, \quad p' = \mathbf{s}_i^{j'}, \quad d = \frac{(\mathbf{s}_i^{j'} - \mathbf{s}_i^j) \times (p \times p')}{||\mathbf{s}_i^{j'} - \mathbf{s}_i^j||^2}$$

We take this idea but apply it to grasp loops, instead of paths. Unlike in path planning, where the direction of τ matters, we only care how or whether loops are linked. Accordingly, we assert that h is always non-negative.

B. Computing the $\mathcal{G}_L$-signature

The $\mathcal{G}_L$-signature is composed of the h-signatures $h(\tau, \mathbf{S})$ of grasp loops τ formed by the robot and DOO. The grasp loops are constructed based on a graphical model of the state where vertices are the robot base, its grippers, and attach points, and edges are paths between them. Attach points are used to represent locations on the DOO which are fixed relative to the robot (e.g. plugged into the wall or tethered to the robot itself). Figure 3 illustrates how the graph construction step works. The robot base vertex is connected to all gripper and attach vertices because it connects to the grippers directly (via the robot geometry) and the attach points indirectly (via the environment). Edges connect grippers/attach points to one another if they are adjacent on the DOO. In Figure 3, the vertices (g_1, g_2) are adjacent, as are (g_2, a_1), but (g_1, a_1) are not.

From this graph, we extract all cycles of 3 vertices which contain a gripper, and convert each cycle to a grasp loop τ. We consider only length-3 cycles containing a gripper in our signature because these contain the relevant information

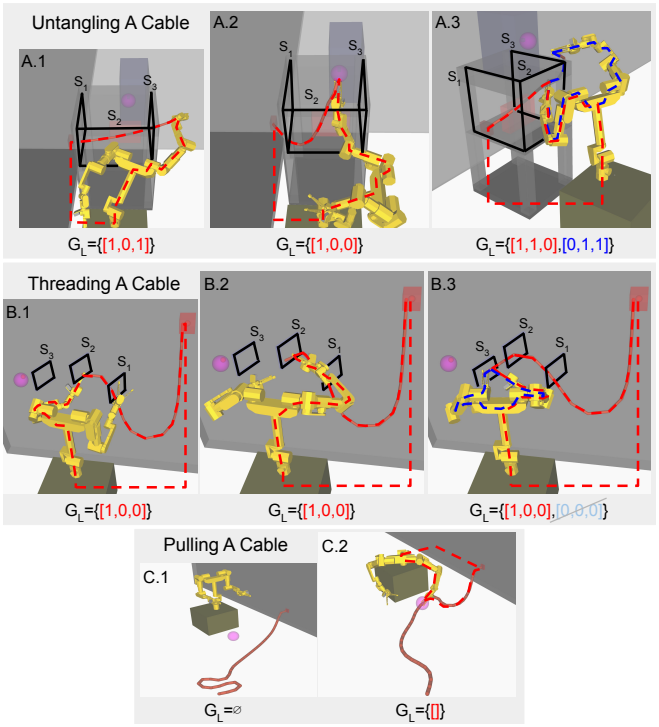

Fig. 4: Example scenes and their $\mathcal{G}_L$ values. The dashed blue and red lines are grasp loops, and the solid black lines are obstacle loops. (B.3) The blue grasp loop is omitted from the signature, as described in Section III-B. The purple sphere shows the goal region.

for planning how to manipulate the DOO. Including longer cycles would add redundant information. Likewise, cycles not containing a gripper are omitted for compactness, since we assume that attach points cannot be changed by the planner. To make a grasp loop from a cycle, we concatenate the 3D paths represented by the cycles' edges. This requires a skeletonized representation of the robot geometry, which can be constructed from the kinematic tree and the origins of the links, as well as the points representing the DOO. A path between the robot base and an attach point ((b, a_1) in Figure 3) can be chosen arbitrarily, as long as it is the same for all states.

For each grasp loop τ_i, we compute the associated h-signature $h(\tau_i, \mathbf{S})$. The $\mathcal{G}_L$-signature of the state, denoted $\mathcal{G}_L(s)$, is the *multiset* of the h-signatures of each grasp loop. In a multiset the order does not matter, but elements may repeat. The number of repetitions of an element is called its multiplicity. Two multisets are equivalent if their elements and multiplicities are equal. Preserving repetitions in the $\mathcal{G}_L$-signature allows us to represent multiple grasp loops that go through the same obstacle loop.

This may result in a grasp loop τ containing two grippers for which $h(\tau, \mathbf{S}) = \mathbf{0}$ (i.e. not linked $\mathbf{S}$). The blue dashed grasp loop shown in Figure 4 B.3 shows this. Releasing one of the grippers does not categorically change what we can do with the object, and neither would grasping with an additional gripper right next to two already grasping. Therefore, if there is a cycle with $h(\tau, \mathbf{S}) = \mathbf{0}$ containing two grippers, one of the grippers is removed from the graph and the process restarts from the cycle extraction step.

C. Computational Complexity

The complexity of computing the $\mathcal{G}_L$-signature can be written in Big-O notation based on the number of skeletons n_s, line segments in the skeleton l_s, arms and/or attach points n_a, and the length of the arms and/or DOO l_a. Due to the rules of construction defined above, in the worst case where all arms are grasping, the graph of grasps loops is a fan graph (F_{1,n_a}) [28]. In the base case of $n_a = 2$, the graph has 3 vertices and one cycle of length 3. Adding another vertex adds one cycle, so in the worst case n_a arms/attach points create $n_a - 1$ loops. Each cycle (loop) is compared with each skeleton, and the number of comparisons scales linearly with both the number of line segments and the length of the loop, giving a total complexity of $O\big((n_a - 1) n_s l_s l_a\big)$. Our Python implementation, using the NetworkX library [29] for computing the $\mathcal{G}_L$-signature for a state, takes $\leq$10ms in the Untangling and Threading environments.

IV. DOO Manipulation with $\mathcal{G}_L$-signature

A. Problem Statement

In this section, we define the DOO manipulation problem which our proposed planning method addresses. In our experiments, the robot has two 7-dof arms attached to a 2-dof torso with parallel-jaw grippers, but the $\mathcal{G}_L$-signature can be applied to other robot morphologies. We assume we have a complete geometric model and skeleton of the environment. When manipulating with the current grasp, the action space is joint velocities $\dot{q}$. We describe points on the DOO primarily by their location $l \in [0, 1]$, where $l = 0$ is one end of the DOO and $l = 1$ is the other. Each location also corresponds to a point $p(l) \in \mathbb{R}^3$. Grasps are represented by a vector of locations $\boldsymbol{l} = [l_1, l_2]$, one for each gripper. A set of grasp locations $\boldsymbol{l}$ must also be paired with a collision-free motion of the robot to the new grasp locations, which may be reachable by many distinct joint configurations.

The goal of the manipulation is to bring a *keypoint* l_k on the DOO to a goal region with position p_{goal} and radius d_{goal}. This is a useful skill for plugging in cables, or for using tools with an attached cable or hose, and more complex tasks like cable harnessing can be described as a sequence of these point reaching goals. Additionally, one can specify a desired $\mathcal{G}_L$-signature for the goal $\mathcal{G}_{L\,\text{goal}}$. This type of DOO manipulation is complementary to tying or untying knots, which has been addressed in prior work [5]–[7].

B. DOO Point Reaching Method

Algorithm 1 describes our method for point reaching tasks. This algorithm serves to demonstrate the utility of the $\mathcal{G}_L$-signature in planning. It uses cost functions that are designed for our robot and our scenarios, and these functions may need to be modified for other applications. Given the current grasp, we use MPPI [30] to find an action $\dot{q}$ that minimizes the goal cost C_{goal}, shown in Eq. (2). MPPI runs until the goal is reached or progress stops. If progress stops, we plan

Algorithm 1 DOO Point Reaching with the $\mathcal{G}_L$-signature

1: **procedure** POINTREACHING($s, n_x, l_k, p_{\text{goal}}, d_{\text{goal}}$)
2: **for** $i < i^{\max}$ **do**
3: $\dot{q}$ =MPPI($s, p_{\text{goal}}, C_{\text{goal}}$)
4: $s = f(\dot{q})$ ▷ Execute and get state
5: **if** $||p_{\text{goal}} - p(l_k)|| < d_{\text{goal}}$ **then**
6: break
7: **if** trapped **then**
8: $d_0 = \min(|\boldsymbol{l}_0 - l_k|)$ ▷ Initial geodesic
9: $\boldsymbol{l}^* = \textit{PlanGrasp}(s, n_x, C_{\text{grasp}}, l_k, \mathcal{B})$
10: $d^* = \min(|\boldsymbol{l}^* - l_k|)$
11: **if** $d^* \geq d_0$ **then** ▷ Unable to grasp closer
12: Add $\mathcal{G}_L(s)$ to $\mathcal{B}$
13: $\boldsymbol{l}^* = \textit{PlanGrasp}(s, n_x, C_{\text{grasp}}, l_k, \mathcal{B})$
14: ExecuteGraspChange($\boldsymbol{l}^*$)

and execute a grasp change, and resume running MPPI. This process is repeated until the goal is reached (trial success) or for $i^{\max}$ iterations (trial failure). For both MPPI and grasp planning, we model the dynamics of the robot, rope, and obstacles in MuJoCo [31].

The method for determining if MPPI is trapped, called trap detection, is adapted from [32]. Trap detection operates on a window of recent joint configurations $q_1, \ldots, q_T$, and computes the average one-step state difference $\bar{q} = \frac{q_T - q_1}{T}$ and keeps a running maximum of this value $\bar{q}^+$ over the trial. MPPI is considered trapped when the ratio $\frac{\bar{q}}{\bar{q}^+}$ is below a threshold (0.2-0.3 in our experiments).

The goal cost used for MPPI is shown in Equation (2), where the state s is used to compute the grasp locations $\boldsymbol{l}$, grasping state $\mathbb{1}_{\boldsymbol{g}}$, keypoint position $p(l_k)$, grasp positions $p(\boldsymbol{l})$, and number of contacts n_{con}.

$$C_{\text{goal}}(s, \dot{q}) = ||p(l_k) - p_{\text{goal}}|| + \alpha_1 \sum_{i=1}^{n_a} \mathbb{1}_{\boldsymbol{g},i} ||p(l_i) - p_{\text{goal}}|| + \alpha_2 \sqrt{n_{\text{con}}} + \alpha_3 ||\dot{q}|| \tag{2}$$

The first term in Eq. 2 brings the keypoint $p(l_k)$ towards the goal p_{goal}. The second term provides a reward for moving any grippers which are grasping towards the goal. n_a is the number of arms, $\mathbb{1}_{\boldsymbol{g}}$ is a binary vector ($\mathbb{1}_{\boldsymbol{g},i} \in \mathbb{1}_{\boldsymbol{g}}$) indicating which grippers are grasping, and $l_i \in \boldsymbol{l}$ are the current grasp locations. This term is useful when the DOO is slack and the keypoint cannot be pulled directly (See Figure 4 C). The third term penalizes collision between the robot and environment, based on the number of contacts n_{con} reported by the dynamics. Finally, the fourth term penalizes high joint velocities to encourage smooth motion. The hyperparameters $\alpha_{1,2,3}$ were selected to prioritize collision avoidance first, then bringing the keypoint to the goal.

In *PlanGrasp* we sample n_x grasps (≈50) and choose the best one. Grasps are sampled by first choosing a strategy for each gripper. The possible strategies are `STAY`, `GRASP`, `MOVE`, or `RELEASE`. `STAY` means the gripper does not change its grasp, or remains not grasping. `GRASP` means the gripper is not grasping and should create a new grasp. `MOVE` means the gripper is currently grasping, but should move to a new grasp location. `RELEASE` means the gripper is currently grasping and should release. For the `GRASP` or `MOVE` strategies, we sample a location $l \in [0, 1]$. At least one gripper must be grasping. For each candidate grasp, we simulate release and grasp dynamics using MuJoCo. Modeling grasping using friction and caging is challenging, so we instead use equality constraints between the rope and the grippers that are grasping. MoveIt [33] is used to find collision-free paths to move the grippers to desired grasp locations. The result is a candidate state s and collision free trajectory for each candidate grasp. We choose the grasp with the lowest cost according to Eq. 3. With abuse of notation, we say the candidate state s, change in state Δs, and grasp state $\mathbb{1}_{\boldsymbol{g}}$ are derived from the candidate grasp locations $\boldsymbol{l}$.

$$C_{\text{grasp}}(\boldsymbol{l}) = \mathbb{1}_{\text{feasible}} + \mathbb{1}_{\mathcal{B}}(s) + \mathbb{1}_{\mathcal{G}_L}(s, \mathcal{G}_{L\,\text{goal}}) + \mathbb{1}_{\boldsymbol{g}} \cdot |\boldsymbol{l} - l_k| + \beta_1 \Delta s \tag{3}$$

The first term in Eq 3 assigns a large penalty (e.g. 100) if no collision-free path to the grasp was found. The next two terms assign a large penalty based on the $\mathcal{G}_L$-signature of candidate state, either for matching a blocklisted signature (second term) or for not matching the goal signature (third term). The fourth term encourages grasping near the keypoint, based on the geodesic distance for any grasping grippers. The final term penalizes the change in robot and DOO state. This results in shorter and faster grasps and is weighted by β_1 to be the least important term. The large penalties dominate the keypoint and state-change terms.

We use a blocklist of $\mathcal{G}_L$-signature's to avoid retrying topological configurations in which we have failed to reach the goal. Specifically, we blocklist the current $\mathcal{G}_L$-signature if the planner cannot find a grasp with lower geodesic cost (4th term in Eq (3)) than the current grasp (Alg 1 lines 8-12). In other words, we do not blocklist if it is possible to grasp closer to the keypoint. This avoids blocklisting the current $\mathcal{G}_L$-signature in the case that progress halts, not because of topological constraints, but because the grasp is too far away from the keypoint to control it. This is inspired by the idea of diminishing rigidity [34], which says that the control over a point on a deformable object decreases as the geodesic distance to the gripper increases. In the case of multiple grippers, the initial grasp locations $\boldsymbol{l}_0$ or new grasp locations $\boldsymbol{l}^*$ may be a list of locations, in which case we use the $\min$ when computing the geodesic distance.

V. APPLICATIONS

We now describe how the above framework can be applied or adapted to DOO manipulation in three different environments, Pulling, Untangling, and Threading. We use a horizon of $H = 15$ in MPPI.

A. Pulling Environment

The Pulling environment contains a large hose attached to a wall. The scene is depicted in Figure 4 C. The robot is initially not grasping the hose, and the head of the hose

Method	Success	Wall Time (m)	Sim Time (m)
$\mathcal{G}_L$-signature (ours)	**22/25**	**12 (5)**	1.4 (1.1)
Always Blocklist	**22/25**	14 (7)	**1.3 (1.0)**
No $\mathcal{G}_L$-signature	10/25	20 (10)	2.0 (1.3)
TAMP50	15/25	142 (116)	2.4 (1.7)
TAMP5	9/25	34 (22)	1.8 (1.2)

TABLE I: Results in the Untangle environment. Times in minutes are for the completion of the task, where Sim Time does not include planning time. Standard deviations are given in parentheses.

is out of the robot's reach. The goal region, shown as a purple sphere, is near the base of the robot on the floor. This environment requires regrasping to bring the head of the hose to the goal, and demonstrates the behavior of the general method in the case where there are no skeletons, and no changes in the $\mathcal{G}_L$-signature.

When applying Alg 1 in this environment, the robot initially chooses a grasp as far down the DOO as it can reach, due to the geodesic cost term in Equation 3. Then, the gripper pulls towards the goal due to the second term in the MPPI cost (2). This brings more of the DOO within reach. When the gripper reaches the goal, the cost cannot be decreased and the controller slows to a stop. At this point, trap detection triggers regrasp planning. Since the DOO is now closer, a plan is found that reaches closer to the tip ($l_k = 1$) than before. Because the grasp is closer to the tip, the current $\mathcal{G}_L$-signature is not blocklisted. This repeats until the grasp is close enough to the tip that it can be brought to the goal region. In the Pulling environment, our method succeeded in 25/25 trials, where each trial differs in the initial DOO configuration and the random seed used for sampling in planning.

B. Untangling Environment

The Untangle environment resembles a computer rack with a cable that needs to be plugged in. The scene is depicted in Figure 4 A. One end of the DOO is fixed to the environment (e.g. plugged in elsewhere), and the robot is initially grasping some other location on the DOO. The robot often needs to regrasp several times in order to reach the goal. Unlike in the Pulling environment, the $\mathcal{G}_L$-signature can take on many different values depending on the configuration of the DOO and the grasp configuration. This demonstrates the utility of the $\mathcal{G}_L$-signature in planning when there is no goal $\mathcal{G}_L$-signature.

We evaluate Alg 1 on this task, and compare to an ablation that omits the two terms using the $\mathcal{G}_L$-signature from Eq 3. We call this method *No $\mathcal{G}_L$-signature*. This often results in greedy re-grasping of the keypoint. We also evaluate a version called *Always Blocklist*, where we blocklist the current $\mathcal{G}_L$-signature every time a trap is detected. Finally, we compare our proposed method to a method inspired by task and motion planning (TAMP), where H additional steps of MPPI are simulated for each candidate grasp during planning and the final goal cost is used in place of cost terms relying on the $\mathcal{G}_L$-signature. We test two versions of this method with $H = 5$ and $H = 50$. Success rates and trial times are shown in Table I. Trials vary in the initial configuration of the robot, grasp location, DOO configuration, in the size of the computer rack, and in the location of the goal.

Algorithm 2 DOO Threading with the $\mathcal{G}_L$-signature

procedure THREADING($s, p_{\text{goal}}, n_x, \mathcal{G}_{L1}, \ldots, \mathcal{G}_{LN}$)
  $j = 1$ ▷ Threading subgoal index
  for $i < i^{\max}$ **do**
    if $j < N$ **then** ▷ threading subgoals
      $\dot{q}$ =MPPI($s, \mathcal{G}_{Lj}, C_{\text{goal}}$)
      $s = f(\dot{q})$ ▷ Execute and get state
      if disc penetrated **then**
        $\boldsymbol{l}^* = \mathit{PlanGrasp}(s, n_x, C_{\text{grasp}}, 1, \mathcal{B})$
        if $\mathcal{G}_L(\boldsymbol{l}^*) == \mathcal{G}_{Lj}$ **then**
          ExecuteGraspChange($\boldsymbol{l}^*$)
      if trapped **then**
        $\boldsymbol{l}^* = \mathit{PlanGrasp}(s, n_x, C_{\text{grasp}}, l - 0.05, \mathcal{B})$
        ExecuteGraspChange($\boldsymbol{l}^*$)
      if $\mathcal{G}_L(s) == \mathcal{G}_{Lj}$ **then**
        $j = j + 1$ ▷ next subgoal
    else ▷ final point reaching
      $\dot{q}$ =MPPI($s, p_{\text{goal}}, C_{\text{goal}}$)
      $s = f(\dot{q})$ ▷ Execute and get state
      if $p_{\text{goal}} - p(l_k) < d_{\text{goal}}$ **then**
        break

Methods using the $\mathcal{G}_L$-signature have the highest success rates and are faster than alternatives. *Always Blocklist* has an equivalent success rate as the full proposed method, but prematurely abandons grasps that would lead to reaching the goal. Our method and the *Always Blocklist* method each failed in 3 trials by trying too many unsuccessful grasps before $i^{\max}$ was reached. The *No $\mathcal{G}_L$-signature* ablation and both TAMP methods usually fail by greedily trying to grasp the keypoint. Without a very long horizon or the $\mathcal{G}_L$-signature, the planner often grasps with configurations that make reaching the goal impossible. The longer horizon used in $H = 50$ helps alleviate this issue but is insufficient in many cases while also causing a 10x increase in planning time.

C. Threading Environment

In the Threading environment, the objective is for the robot to thread the DOO through a series of fixtures in a specified order (e.g. "fixture 1, then fixture 2, then fixture 3"), after which it should bring the keypoint to a goal region. The threading is described by a series of goal signatures $\mathcal{G}_{L1}, \ldots, \mathcal{G}_{LN}$. This skill could be applied to installing cable harnesses in a car or electrical wiring in a building. One end of the DOO is fixed to the environment, and the robot is initially grasping some other location on the DOO. This environment is depicted in Figure 4 B.

The method is detailed in Alg 2, and key differences to Alg 1 are highlighted in green. In MPC, the $\mathcal{G}_L$-signature is used in the same way as in Alg 1, but the goal cost has been changed to match the new task. Additionally, we use the $\mathcal{G}_L$-signature to ensure certain goal signatures at each stage of threading. To reach a threading subgoal, we augment the

Method	Success	Wall Time (m)	Sim Time (m)
$\mathcal{G}_L$-signature	**42/50**	**8 (2)**	1.3 (0.4)
TAMP5	21/50	17 (3)	1.3 (0.6)
Wang et al. [15]	12/50	**8 (3)**	**1.0 (0.8)**

TABLE II: Results on the Threading task.

goal cost (Eq (2)) with the magnetic-field cost proposed in [15]. This uses the formula $\sum_{j=1}^{n_i} \Phi(\mathbf{s}_i^j, \mathbf{s}_i^{j'}, r)$ from Equation (1) for the direction of the magnetic field, but where r is the keypoint of the DOO. This causes the keypoint to follow virtual magnetic field lines through the fixture in the specified direction.

When a threading subgoal is reached, and the planner returns a grasp which does not match $\mathcal{G}_{L\text{goal}}$, we reject it and continue running MPPI to push the cable further through the fixture. This happens when there is no feasible grasp matching $\mathcal{G}_{L\text{goal}}$ due to obstacles or reachability issues. Furthermore, we also check $\mathcal{G}_{L\text{goal}}$ after executing the grasp to ensure that any deviations that occurred when executing the grasp plan do not change the $\mathcal{G}_L$-signature. To check when a threading subgoal is reached, we use the disc penetration check from [15]. The goal signatures $\mathcal{G}_{L1}, \ldots, \mathcal{G}_{LN}$ are used in the grasp planning (3rd term in Eq (3)), but the blocklist is not (2nd term). Grasp sampling is restricted to alternating single-gripper grasps, which speeds up grasp planning. The keypoint location for grasp planning is also restricted to speed up planning. It is chosen to be the tip ($l_k = 1$) when a threading subgoal is reached, and further down the DOO than the current grasp when stuck ($l_k = l - 0.05$).

We compare our proposed method to the TAMP5 method described previously. In this environment, the TAMP method often chooses grasps that correctly thread through fixtures 1 and 2, because those grasps allow immediate progress towards the next subgoal. However, it often grasps incorrectly on fixture 3, which requires the robot to first reach further around and results in less immediate progress towards the next subgoal. We also adapted the method in [15] from a single floating gripper to our dual arm robot. As in our method, we use alternating single-gripper grasps. Instead of the more general trap detection method we use, this baseline checks the distance between the gripper and the fixture being threaded. This baseline fails similarly to the TAMP5 method, but additionally fails when MPPI is trapped but is outside the distance-to-fixture threshold. Success rates and trial times are shown in Table II. Trials vary in the initial configuration of the robot, grasp location, DOO configuration, and in the positions of the fixtures. In the trials in which our method failed to complete the task, MPPI reached a joint configuration with one arm that prevented the other arm from grasping the DOO at or near the tip, as required by our method. This means the robot remained stuck until $i^{\max}$ was reached.

D. Real World Threading

We demonstrate a simplified version of the Threading task in the real world, as depicted in Figure 1. This shows the applicability of the proposed methods in the presence of significant calibration, perception, and dynamics modeling errors. We use CDCPD2 [35] to track the DOO and visual servoing from in-hand cameras to guide grasping. The environment geometry is specified manually, and the simulation dynamics were tuned to match the real world setup as closely as possible for the particular setup.

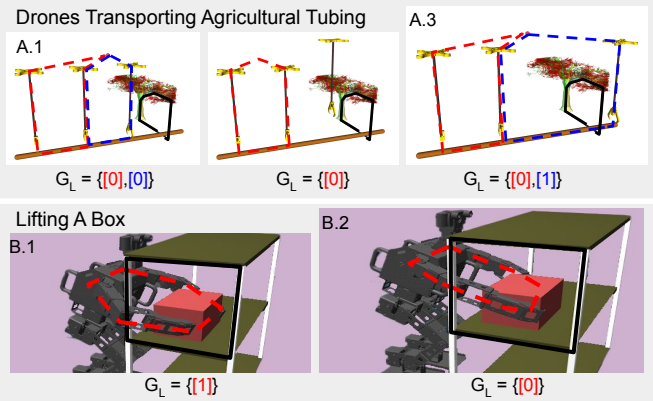

Fig. 5: Additional examples where the $\mathcal{G}_L$-signature may be useful. (A) Drones lifting a large object can be treated similarly to a multi-armed robot. (B) Dual-arm grasping of large rigid objects can result in distinct signatures.

VI. DISCUSSION

The planning methods described above are somewhat specific to their respective tasks, and the tasks represent a small subset of the domain of rope and cable manipulation. However, the $\mathcal{G}_L$-signature may be useful for other planning methods and other tasks, as shown in Figure 5. In both of these examples, we can compute the $\mathcal{G}_L$-signature and see that it categorizes different states in ways that may be useful for planning.

We use the $\mathcal{G}_L$-signature as part of the cost function in sampling-based MPC, but it could also be used as a constraint in RRT or A* planners (similar to [8]). Additionally, while the signature is composed of integers when all grasp and obstacle loops are closed, for incomplete loops they take on real values and have gradients which may be used in gradient-based planning methods. The $\mathcal{G}_L$-signature could also be used to constrain the topology of shape completion or scene reconstruction. For example, to ensure that when reconstructing the shape of a hook, it does not form a closed loop that would trap the robot arm or DOO.

VII. CONCLUSION

In this paper, we proposed the $\mathcal{G}_L$-signature which describes the topology of closed loops formed by grasping the DOO with respect to closed loops formed by stationary objects in the environment. Our $\mathcal{G}_L$-signature builds on the h-signature proposed in prior work on topological path planning. Furthermore, we describe an algorithm for manipulating DOOs that plans grasps based on the proposed $\mathcal{G}_L$-signature. In our experiments, we find that using the $\mathcal{G}_L$-signature improves task success and reduces planning times compared to a task and motion-planning method. Finally, we use the method to thread a cable and bring it to a goal region on a real robot.

REFERENCES

[1] Gabriel Arslan Waltersson, Rita Laezza, and Yiannis Karayiannidis. Planning and control for cable-routing with dual-arm robot. *ICRA*, 2022.

[2] Ashvin Nair, Dian Chen, Pulkit Agrawal, Phillip Isola, P. Abbeel, Jitendra Malik, and Sergey Levine. Combining self-supervised learning and imitation for vision-based rope manipulation. *ICRA*, 2017.

[3] Wilson Yan, Ashwin Vangipuram, P. Abbeel, and Lerrel Pinto. Learning predictive representations for deformable objects using contrastive estimation. In *Conference on Robot Learning*, 2020.

[4] Hidefumi Wakamatsu, Akira Tsumaya, Eiji Arai, and Shinichi Hirai. Manipulation planning for knotting/unknotting and tightly tying of deformable linear objects. *ICRA*, 2005.

[5] Mitul Saha and Pekka Isto. Manipulation planning for deformable linear objects. *IEEE Trans. Robotics*, 2007.

[6] Priya Sundaresan, Jennifer Grannen, Brijen Thananjeyan, Ashwin Balakrishna, Jeffrey Ichnowski, Ellen R. Novoseller, Minho Hwang, Michael Laskey, Joseph Gonzalez, and Ken Goldberg. Untangling dense non-planar knots by learning manipulation features and recovery policies. In *RSS*, 2021.

[7] Weifu Wang and Devin Balkcom. Knot grasping, folding, and regrasping. *IJRR*, 2018.

[8] Subhrajit Bhattacharya, Maxim Likhachev, and Vijay Kumar. Identification and representation of homotopy classes of trajectories for search-based path planning in 3d. In *RSS*, 2011.

[9] Subhrajit Bhattacharya, Maxim Likhachev, and Vijay R. Kumar. Topological constraints in search-based robot path planning. *Autonomous Robots*, 2012.

[10] Takeo Igarashi and Mike Stilman. Homotopic path planning on manifolds for cabled mobile robots. *WAFR*, 2010.

[11] *Path Deformation Roadmaps: Compact Graphs with Useful Cycles for Motion Planning*, 2008.

[12] Hidefumi Wakamatsu, Akira Tsumaya, Eiji Arai, and Shinichi Hirai. Manipulation planning for unraveling linear objects. *ICRA*, 2006.

[13] Jennifer Grannen, Priya Sundaresan, Brijen Thananjeyan, Jeffrey Ichnowski, Ashwin Balakrishna, Minho Hwang, Vainavi Viswanath, Michael Laskey, Joseph Gonzalez, and Ken Goldberg. Untangling dense knots by learning task-relevant keypoints. In *Conference on Robot Learning*, 2020.

[14] Priya Sundaresan, Jennifer Grannen, Brijen Thananjeyan, Ashwin Balakrishna, Michael Laskey, Kevin Stone, Joseph Gonzalez, and Ken Goldberg. Learning rope manipulation policies using dense object descriptors trained on synthetic depth data. *ICRA*, 2020.

[15] Weifu Wang, Dmitry Berenson, and Devin Balkcom. An online method for tight-tolerance insertion tasks for string and rope. *ICRA*, 2015.

[16] Daniel Sánchez, Weiwei Wan, and Kensuke Harada. Tethered tool manipulation planning with cable maneuvering. *IEEE Robotics and Automation Letters*, 2019.

[17] Peter Mitrano, Dale M^{c}Conachie, and Dmitry Berenson. Learning Where to Trust Unreliable Models in an Unstructured World for Deformable Object Manipulation. *Science Robotics*, 2021.

[18] Dale McConachie, Thomas Power, Peter Mitrano, and Dmitry Berenson. Learning When to Trust a Dynamics Model for Planning in Reduced State Spaces. *IEEE Robotics and Automation Letters*, 2020.

[19] Peter Mitrano and Dmitry Berenson. Data augmentation for manipulation. *RSS*, 2022.

[20] Peter Mitrano, Alex LaGrassa, Oliver Kroemer, and Dmitry Berenson. Focused adaptation of dynamics models for deformable object manipulation. *ICRA*, 2023.

[21] Jeffrey Mahler, Florian T. Pokorny, Brian Hou, Melrose Roderick, Michael Laskey, Mathieu Aubry, Kai Kohlhoff, Torsten Kröger, James J. Kuffner, and Ken Goldberg. Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards. In *ICRA*, 2016.

[22] Jeffrey Mahler, Jacky Liang, Sherdil Niyaz, Michael Laskey, Richard Doan, Xinyu Liu, Juan Aparicio Ojea, and Ken Goldberg. Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics. In *Robotics: Science and Systems*, 2017.

[23] Ryan Hoque, Daniel Seita, Ashwin Balakrishna, Aditya Ganapathi, Ajay Kumar Tanwani, Nawid Jamali, Katsu Yamane, Soshi Iba, and Ken Goldberg. Visuospatial foresight for multi-step, multi-task fabric manipulation. *RSS*, 2020.

[24] Xingyu Lin, Yufei Wang, Zixuan Huang, and David Held. Learning visible connectivity dynamics for cloth smoothing. *Conference on Robot Learning*, 2021.

[25] Yilin Wu, Wilson Yan, Thanard Kurutach, Lerrel Pinto, and P. Abbeel. Learning to manipulate deformable objects without demonstrations. *ArXiv*, abs/1910.13439, 2019.

[26] Fan Zhang and Y. Demiris. Learning garment manipulation policies toward robot-assisted dressing. *Science Robotics*, 2022.

[27] Thierry Siméon, Jean-Paul Laumond, Juan Cortés, and Anis Sahbani. Manipulation planning with probabilistic roadmaps. *IJRR*, 2004.

[28] Eric W Weisstein. Fan graph. *MathWorld – Wofram Web Resource*.

[29] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. *Python in Science Conference*, 2008.

[30] Grady Williams, Paul Drews, Brian Goldfain, James M. Rehg, and Evangelos A. Theodorou. Aggressive driving with model predictive path integral control. *ICRA*, 2016.

[31] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *IROS*, 2012.

[32] Sheng Zhong, Zhenyuan Zhang, Nima Fazeli, and Dmitry Berenson. TAMPC: A controller for escaping traps in novel environments. *RA-L*, 2021.

[33] David Coleman, Ioan A. Şucan, Sachin Chitta, and Nikolaus Correll. Reducing the barrier to entry of complex robotic software: a moveit! case study. *Journal of Software Engineering for Robotics*, 2014.

[34] Dmitry Berenson. Manipulation of deformable objects without modeling and simulating deformation. In *IROS*, 2013.

[35] Yixuan Wang, Dale McConachie, and Dmitry Berenson. Tracking Partially-Occluded Deformable Objects while Enforcing Geometric Constraints. *ICRA*, 2021.