Learning Robot Policies for Untangling Dense Knots in Linear Deformable Structures

Jennifer Grannen*¹, Priya Sundaresan*¹, Brijen Thananjeyan¹, Jeff Ichnowski¹, Ashwin Balakrishna¹, Vainavi Viswanath¹, Michael Laskey², Joseph E. Gonzalez¹, Ken Goldberg¹

¹Department of Electrical Engineering and Computer Sciences
University of California Berkeley United States
²Toyota Research Institute (TRI)
jenngrannen@berkeley.edu, priyasundaresan@berkeley.edu
* equal contribution

Abstract: Untangling linear deformable objects (LDOs) such as ropes, wires, and cables is a challenging robot task due to the vast set of possible arrangements, visual homogeneity, self-occlusions, and complex dynamics. Prior geometric methods have focused on loose knot configurations using full rope state estimation before planning, while prior data-driven approaches require substantial environment interaction. We consider dense (tight) knots that lack space between self-intersections and present an iterative algorithm that leverages geometric structure in configurations to untie knots. We then practically instantiate this algorithm with HULK: Hierarchical Untangling from Learned Keypoints, which combines learning-based perception with a bilateral geometric planner to perform untangling without performing full state estimation. To evaluate the learned policy, we develop a novel simulator to simulate rope with figure-eight and overhand knots and with varying appearances. We present experiments comparing two variants of HULK to three baselines and find that HULK is able to untangle LDOs consisting of tight figure-eight and overhand knots. We find that HULK achieves higher success rates and fewer empirical actions than analytical baselines while generalizing to varied textures and appearances. HULK is able to successfully untangle an LDO from a dense initial configuration containing only up to two overhand and figure-eight knots in 97.9% of 378 simulated experiments with an average of 12.1 actions per trial, suggesting that the policy can learn the task of untangling effectively from an algorithmic supervisor.

Keywords: Deformable Manipulation, Computer Vision

1 Introduction

Untangling linear deformable objects (LDOs) has a wide range of applications in surgery [1, 2, 3], manufacturing [4], and households [5]; however, this task has two challenges: (1) state estimation and (2) manipulation. State estimation is complicated by the high-dimensional configuration space, visual homogeneity of the LDO, and self-occlusions present in knots. Manipulation is complicated by friction and tension in knots, stiffness, and the need for bilateral motions to achieve loosening and manage slack [6, 7, 8, 9]. Prior work on untangling explored both classical [8, 6] and learning-based perception methods [7] to estimate the state of loosely knotted LDOs, and used geometric algorithms or learning from demonstrations to perform manipulation [8, 6, 7, 10]. However, to infer an LDO’s state, these methods often rely on accurate segmentation, which is challenging in knotted configurations that lack space between adjacent LDO segment crossings [11]. To address these challenges, we present a geometric algorithm and a keypoints-based implementation based on geometric planning and advances in deep learning for perception.

The algorithm, Basic Reduction of Under-Crossing Entanglements (BRUCE), sequentially undoes one crossing at a time until the LDO has no self-intersections using two manipulation primitives from knot theory [8]: (1) straightening actions that pull the endpoints of the LDO apart to pull the LDO
taut, disambiguate the configuration, and possibly undo self-crossings and (2) loosening actions that pull slack out from under a crossing while holding an anchor point in place to undo crossings.

We implement BRUCE by incorporating deep-learning-based sensing to create HULK: Hierarchical Untangling from Learned Keypoints. HULK generates a perception-driven policy for untangling densely knotted LDOs. We build on recent work in keypoint detection as in Papandreou et al. [12] by learning to predict only keypoints on the LDO that are directly relevant to the task. While prior work in deformable object tracking estimated the full state of the LDO (e.g., through dense object descriptors [7, 13, 14]), full state estimation is often unnecessary to perform many downstream tasks such as untangling. This motivates learning only features of the LDO that are relevant to the task at hand. HULK learns keypoints that provide task-specific contextual information, which can enable fine-grained tracking and manipulation of crossings and endpoints without full state estimation.

This paper makes the following contributions: (1) a novel simulator in Blender for modelling the dynamics of linear deformable objects; (2) HULK (Hierarchical Untangling from Learned Keypoints): a novel framework for untangling densely knotted LDOs in simulation using a learned perception system which predicts task-relevant keypoints and exploits geometry (3) extensive simulation experiments comparing HULK against several baseline policies with alternative perception pipelines or access to ground truth. These experiments suggest that HULK can achieve untangling with empirically fewer or comparable actions to baselines with more state information, and also suggests that HULK can generalize to various configurations and visual appearance.

2 Related Work

Figure 1: HULK is an algorithm for untangling linear deformable objects (LDOs) trained in simulation. HULK’s learned system identifies keypoints to remove figure-eight and overhand knots from an LDO, and is trained to be robust to several variations (bottom row). The top two rows show a full sequence of untangling actions on an LDO with a figure-eight knot and an overhand knot. Reidemeister moves straighten the rope (blue pick-place actions), and Node Deletion moves remove crossings (red pin point and green pull action).

Manipulation of LDOs using analytic [8, 6, 15, 16], learned [17, 9], and hybrid [7, 18] approaches has received considerable interest in recent years, but to the best of our knowledge the ability to handle dense knots has not been addressed. Effective LDO manipulation also often requires some degree of state estimation from perception, such as through nonrigid registration methods [19] or more recent deep-learning approaches for global object correspondence [20, 13, 14, 7, 21]. We extend work from Lui and Saxena [8], which achieves 76.9% success on physical experiments of untangling loosely knotted LDOs. In their pioneering 2013 paper, the authors tackle state estimation by representing an LDO as a graph and approximating it using RGB-D perception and feature engineering [8]. They plan untangling actions by manipulating nodes and edges to minimize crossings in the graph, taking into account heuristics such as measurements of slack and empty space. We build on their work by using a geometrically motivated algorithm that circumvents full state estimation coupled with a learning-based policy for robustness and generalization to densely knotted LDOs.

Several works have also demonstrated robot knot tying and LDO arrangement into desired shapes given human demonstrations. Sundaresan et al. [7] apply a geometric approach to both tasks by learning a dense object descriptor mapping [13, 14] in simulation, enabling deformable correspondence and tracking. This method provides a similar ordering of pixels to a graph structure. However, when LDO configurations are complex, robust state estimation becomes challenging, largely due to the lack of adequate segmentation of the LDO and its overlapping segments (Fig. 4 B). In contrast, HULK focuses only on semantically meaningful keypoints, as opposed to global LDO correspondence, to allow for fine visuomotor control required for untangling. Other methods present algorithms for knot tying and LDO manipulation but do not leverage geometric structure [9, 6, 2], whereas we reason
about the geometry of intersections to infer appropriate untangling actions. In contrast to approaches which attempt to learn manipulation policies end-to-end, we also build on recent approaches to LDO manipulation that decouple perception from control \[7, 18, 17\]. While these approaches are sample efficient and modular, they attempt to learn intermediate or explicit state representations unlike HULK, which estimates task specific keypoints.

Several researchers have focused on a larger class of problems in 1D and 2D deformable objects, including fabrics and clothing. Soft object manipulation using classical perception methods \[8, 22, 23, 24\] has proven effective, but these methods require hand-engineered features that are often specific to the experimental setup and can fail to handle highly deformed configurations. To mitigate the perception and modelling challenges associated with deformable objects, several works have employed deep learning-based methods including reinforcement learning \[25, 26, 27\], IL techniques \[28, 29, 30\], video prediction models \[31, 32\], and other approaches \[7, 33, 9, 6, 15\]. However, accurate reward engineering and loss function specification remains challenging in these methods, partially due to the infinite-dimensional state spaces of deformable objects. End-to-end learning also can result in black-box policies that abstract away visual reasoning and lack interpretability. Instead, we separate visual reasoning from manipulation to leverage task-specific geometry and generate visually interpretable manipulation plans. Similar strategies decoupling perception and manipulation have proven effective in a broad range of robotic manipulation tasks \[7, 33, 34, 35, 36\].

3 Problem Statement

Given an RGB image of an LDO in a semi-planar densely knotted initial configuration, the objective is to manipulate it with a sequence of actions to a fully untangled state with no crossings. We formulate this problem using a undirected graph abstraction \(G = (V,E)\) to model an LDO’s configuration \[8\]; HULK, however, only takes an image as input and does not attempt to explicitly reconstruct \(G\) or estimate its parameters.

The graph, illustrated in Fig. 2, models an LDO configuration. Each vertex \(v \in V\) is an intersection (node) or endpoint; intermediate points between nodes and endpoints are not included in the graph since they do not participate in crossings. Each edge \(e \in E\) is defined as a 2-vertex tuple \(e = (u, v), u \in V, v \in V\). It is possible to have multiple edges between vertices such as between nodes 2 and 3 in Fig. 2 \(A\). We denote the leftmost degree-one vertex \(v_1\), and the rightmost degree-one vertex (endpoint) \(v_2\), breaking ties arbitrarily. For every vertex \(v\), we annotate the incident edges \(e = (v, v') \in E\) with + or − as follows (Fig. 2).

\[
X(v, e) = \begin{cases} 
+ & \text{if vertex } v \text{ is an endpoint or if } e \text{ crosses above the other edge at vertex } v \\
- & \text{otherwise}
\end{cases}
\]  

(1)

The graphical representation also provides a termination condition for the task: \(|V| = 2\). When \(|V| = 2\) the LDO is in a state with two endpoints exposed and no intersections.

To quantify the tightness of knotted configurations, we define a looseness measure \(L(E) \in \mathbb{R}\). We consider point-pairs \((p, p')\) sampled from a subspace \(H(u, v) \subset \mathbb{R}^3\) encompassing the physical length and radial thickness of 2 LDO segments joining 2 vertices \(u\) and \(v\), as shown between vertices 2 and 3 in Fig. 2 \(B\). \(L(E)\) is proportional to the maximum Euclidean distance taken over all point pairs sampled from \(H(u, v)\), minimized over all vertex pairs \((u, v)\) that share 2 edges. For convenience, we denote the 2 edges between \((u, v)\) as \(\{e_+, e_-\}\) where \(e_+\) corresponds to the edge with annotation \(X(u, e_+ = +)\) and the opposite notation applies for \(e_-\). In Fig. 2 \(A\) between nodes 2 and 3, \(e_+\) represents the lower edge at node 2, and \(e_-\) corresponds to the upper edge. \(L(E)\) quantifies the minimum amount of empty space between adjacent LDO segments, shown in the purple region with the annotated yellow line segment in Fig. 2 \(A\) and Fig. 2 \(B\).

\[
L(E) = \min_{\{e_+, e_-\} \in E} \max_{p \in e_+, p' \in e_-} \frac{\|p - p'\|_2}{\text{LDO radial thickness}}
\]  

(2)

We define a dense configuration to be one in which \(L(E) = 0\) and a loose configuration has \(L(E) > 0\). In the annotated overhand knot in Fig. 2, the maximum distance between two edges is minimized between vertices 2 and 3 and is nonzero. A non-zero \(L(E)\) occurs when the largest point pair \((p, p')\) distance between adjacent edges is non-zero, corresponding to a loose configuration. When \(L(E) = 0\), as in the completed overhand knot in Fig. 2, at least two edges of the LDO are directly adjacent, and segmentation for state estimation cannot reliably isolate overlapping edges.
We consider the task of untangling LDOs with dense overhand and Fig.-eight knots and varied textures (smooth and braid). Each type of knot is parameterized by the locations of nodes (intersections) and endpoints and can be represented conveniently as a linear graph as in Lui and Saxena [8] annotated with (+, -) according to Section 3.

At each time \( t \), we consider two simultaneously executed actions, one for each arm. We use one pulling and one holding action, both with 4 variables in the global workspace coordinate frame:

\[ a_{t, l} = (x_{t, l}, y_{t, l}, \Delta x_{t, l}, \Delta y_{t, l}) \quad | \quad a_{t, r} = (x_{t, r}, y_{t, r}, \Delta x_{t, r}, \Delta y_{t, r}) \]

The right arm takes a pull action \( a_{t, r} \) by grasping the LDO at \( (x_{t, r}, y_{t, r}) \), lifting by a fixed offset, and travelling by \( (\Delta x_{t, r}, \Delta y_{t, r}) \) before releasing. A hold action is denoted with the special case of \( (x_{t, r}, y_{t, r}) = (0, 0) \). We use the same notation for the left arm actions \( a_{t, l} \).

**Assumptions**  We make the following assumptions on the starting configuration: (1) semi-planar, in which each vertex has at most two LDO segments (four edges), (2) only overhand or figure-eight knots or loops without knots, and (3) endpoint vertices are visible. With the semi-planar assumption, every vertex has degree 1 or 4, \( |V| = N + 2 \), and \( |E| = 2N + 1 \), where \( N \) is the number of nodes.

### 4 Geometric Algorithm

We first introduce BRUCE: Basic Reduction of Under-Crossing Entanglements, an algorithm for untangling an LDO from a semi-planar starting configuration, defined over the graphical model from Section 3. HULK is a practical instantiation of BRUCE that uses learned perception based on object detection and localized keypoint regression discussed in Section 5 to bypass full graph estimation in favor of direct prediction of task-relevant features.

As in Lui and Saxena [8], we use two types of manipulation primitives: Reidemeister moves and Node Deletion moves. Reidemeister moves remove occlusions that are not part of a knot by pulling each end of the LDO to opposite sides of the workspace. Node Deletion moves remove a node in the graphical abstraction of the LDO by pulling the endpoint corresponding to the incident edge labelled − from one side of the crossing to the other while holding down the other incident edge labelled + to prevent the rest of the configuration from shifting. This action removes one node while leaving the other nodes in the LDO’s configuration unchanged, reducing \( |V| \) by one and \( |E| \) by two.

BRUCE iteratively undoes crossings from the right endpoint until no self occlusions remain. It starts by performing a Reidemeister move to remove crossings separate from the relevant knots and disambiguate the configuration. Next, it identifies the next under-crossing, \( c \), that must be undone by selecting the first vertex adjacent to two successive − annotated edges. It then performs one Node Deletion move on \( c \) followed by a Reidemeister move. It repeats these two steps until there are no occlusions left in the LDO, corresponding to a graph representation with \( |V| = 2 \) for the endpoints alone. This procedure is illustrated in Fig. 3.

If we assume each operation is executed correctly, BRUCE is guaranteed to untangle any semi-planar configuration. Each iteration of BRUCE performs Node Deletion and Reidemeister operations that monotonically reduce the number of crossings until no crossings remain.

### 5 Learning-Based Implementation

HULK learns task-relevant features of an LDO from synthetic data to practically instantiate BRUCE. Here, we discuss the LDO simulation environment and training procedures implemented in HULK.

#### 5.1 Simulation

We propose an implementation that learns from training data generated in simulation. The simulation exposes the ground-truth state of an LDO which is not readily available outside of simulation, allowing...
We partition the algorithm into two subproblems: (1) planning actions to execute the Reidemeister Move on the first under-crossing relative to the right endpoint \( v_r \). While holding the LDO in place, a series of pull actions are performed in a Node Deletion move. Once the right endpoint is revealed, a final Reidemeister move is taken to restore the LDO to a straightened configuration. For the multi-knot case, the process repeats until a final Reidemeister move results in no remaining knots.

for self-supervised collection of annotations and scalable dataset generation. We use Blender 2.8 [37], a graphics and animation suite that supports physics simulations, to create a realistic LDO simulation environment. Compared to other deformable simulators (e.g., Mujoco [38] and PyBullet [39]), Blender provides greater flexibility over the appearance of the LDO for domain randomization.

We construct a mass-spring model of an LDO consisting of 50 rigid-body cylindrical meshes linked by springs following prior work [29, 40]. The simulator supports two textures for the LDO model, (1) a smooth texture consistent with hoses, cables, and tubing; and (2) a braided texture modelled after twine and nylon rope (Fig. 1). We hand-code one knot-tying trajectory per knot type, and add random noise to each trajectory to generate initially perturbed dense configurations.

To generate synthetic training data, we produce a variety of dense initial configurations and construct their corresponding graph representation. We use the ordered set of cylinder positions, queried from Blender’s Python API, and ray tracing to detect crossings and infer the graph. Given the reconstructed graphs, we execute BRUCE and record overhead RGB renderings and ground-truth annotations.

5.2 HULK Implementation

HULK uses learning to model only the portion of the graph that matters to the task at any point in time. We partition the algorithm into two subproblems: (1) planning actions to execute the Reidemeister and Node Deletion moves and (2) perceiving the necessary geometric features for these actions. We introduce two versions of generic HULK that differ only in perception: (1) HULK-G, which operates exclusively on the global image of an LDO, and (2) HULK-L, which operates on local knot crops of an LDO in addition to the global image. Both variants of HULK employ a hierarchical manipulation strategy to sequentially undo crossings one by one.

5.2.1 Manipulation

HULK is implemented using a termination condition, Node Deletion moves, and Reidemeister moves planned directly from bounding boxes around each knot in the configuration and predicted keypoints \( \hat{p}_r, \hat{p}_l, \hat{p}_{pull}, \hat{p}_{hold} \). The keypoints, \( p_l, p_r, p_{pull}, \) and \( p_{hold} \), indicate the pixel locations of the left and right LDO endpoints \( v_l \) and \( v_r \), and the pull and hold grasps for the next planned Node Deletion move on the first under-crossing relative to \( v_r \), denoted \( c \) (Fig. 3).

In a Node Deletion move (Alg. 2, Ln. 5) at time \( t \), the left arm grasps at \( \hat{p}_{pull} \) and pulls in the direction of the action vector, \( \hat{p}_{pull} - \hat{p}_{hold} \), and the right arm grasps at \( \hat{p}_{hold} \) to hold the LDO in place:

\[
\mathbf{a}_{t,l} = (\hat{p}_{x,pull}, \hat{p}_{y,pull}, \hat{p}_{x,pull} - \hat{p}_{x,hold}, \hat{p}_{y,pull} - \hat{p}_{y,hold}) \quad \text{and} \quad \mathbf{a}_{t,r} = (\hat{p}_{x,hold}, \hat{p}_{y,hold}, 0, 0)
\]

In a Reidemeister Move (Alg. 2, Lines 2 and 6), two consecutive actions pull \( \hat{p}_l \) to a predefined point \( w_l \) at one end of the workspace, and \( \hat{p}_r \) to a predefined point \( w_r \) at the opposite end.

\[
\mathbf{a}_{t,l} = (\hat{p}_{x,l}, \hat{p}_{y,l}, w_{x,l} - \hat{p}_{x,l}, w_{y,l} - \hat{p}_{y,l}) \quad \text{and} \quad \mathbf{a}_{t,r} = (\hat{p}_{x,r}, \hat{p}_{y,r}, w_{x,r} - \hat{p}_{x,r}, w_{y,r} - \hat{p}_{y,r})
\]

The untangling termination condition for BRUCE (Alg. 1), \(| V | > 2\), refers to a configuration free of intersections. For practical implementation, HULK slightly relaxes this constraint to allow intersections that do not form a knot, as taking a Reidemeister move (Alg. 2, Ln. 6) will remove any crossings that are not part of a knot. We also want to immediately detect when an endpoint is freed from an under-crossing. However, the bounding box for a knot loosened to this extent is undefined.
Thus, we define a condition to approximate when an action pulls the desired endpoint beyond \( \hat{p}_{\text{hold}} \), such that the inner product between the Node Deletion action vector and the vector between \( \hat{p}_r \) and \( \hat{p}_{\text{hold}} \) is above a threshold \( \lambda \). In experiments, we set \( \lambda \) to 0.7 to favor false negatives over positives, as early termination is a greater risk to untangling than late termination. The final termination condition is summarized by Eq. 3, where we additionally impose a hard limit \( T = 30 \) on the number of actions:

\[
g(I) = \emptyset \quad \text{OR} \quad \left( \langle \hat{p}_r - \hat{p}_{\text{hold}}, \hat{p}_{\text{hold}} - \hat{p}_{\text{pull}} \rangle > \lambda \right) \quad \text{OR} \quad t > T.
\]

A comparison of BRUCE and HULK is outlined here, noting that the perception components required to implement HULK are discussed in detail below:

### Algorithm 1 BRUCE

1. **Require**: Graph \( G = (V, E) \) of LDO
2. Reidemeister move with \( v_r, v_l \)
3. while \(|V| > 2\) do
4. Find \( c \) by traversal (\( v_r \rightarrow v_l \))
5. Node Deletion move on \( c \)
6. Reidemeister move with \( v_r, v_l \)
7. return DONE

### Algorithm 2 HULK

1. **Require**: RGB image of LDO
2. Initial Reidemeister move with \( \hat{p}_r, \hat{p}_l \)
3. while NOT Eq. 3 do
4. Directly predict \( \hat{p}_{\text{pull}}, \hat{p}_{\text{hold}} \)
5. Node Deletion move with \( \hat{p}_{\text{pull}}, \hat{p}_{\text{hold}} \)
6. Reidemeister move with \( \hat{p}_r, \hat{p}_l \)
7. return DONE

#### 5.2.2 Perception

HULK employs perception-based learning to robustly infer only the task-relevant components of the graph of a particular configuration as defined in Section 5.2.1. HULK-G infers all keypoints \( p_l, p_r, p_{\text{pull}}, \) and \( p_{\text{hold}} \) from a full-resolution RGB image of an LDO. HULK-L also infers \( p_l, p_r \) globally, but additionally uses local information to estimate \( p_{\text{pull}} \) and \( p_{\text{hold}} \) directly from a bounding box crop of the right-most knot.

**Knot Detection**: HULK uses a bounding box knot-detector to instantiate a termination condition for the task. HULK-L additionally uses this module to predict \( \hat{p}_{\text{pull}}, \) and \( p_{\text{hold}} \) local to knot crops instead of the global image as in HULK-G. We learn a function \( g : \mathbb{R}^{640 \times 480 \times 3} \rightarrow \mathbb{R}^{4 \times N} \) that maps a full-resolution RGB image to bounding boxes \( (x_{\min}, y_{\min}, x_{\max}, y_{\max})_i \) \( i = 1, \ldots, N \) for \( N \) knots contained in the image using a Mask-RCNN framework as in He et al. [41] (Fig. 4).

**Keypoint Regression**: HULK-L and HULK-G implement separate keypoint regression modules. The distinction is that HULK-L attempts to take advantage of local information both in its style of learning and by explicitly using predicted knot bounding boxes. HULK-G globally regresses all 4 keypoints by learning a mapping \( f_{\text{all}} : \mathbb{R}^{640 \times 480 \times 3} \rightarrow \mathbb{R}^{640 \times 480 \times 4} \) where each of the four channel outputs are a 2D Gaussian centered at \( p_l, p_r, p_{\text{pull}}, \) and \( p_{\text{hold}} \) (Fig. 4). HULK-L, based on on Papandreou et al. [12], is a two-stage coarse-to-fine approach for inferring the locations of desired points. HULK-G learns two mappings \( f_{\text{reid}} : \mathbb{R}^{640 \times 480 \times 3} \rightarrow \mathbb{R}^{640 \times 480 \times (3 \times 2)} \) for inferring \( p_r, p_l \) from the global image and \( f_{\text{node}} : \mathbb{R}^{80 \times 60 \times 3} \rightarrow \mathbb{R}^{80 \times 60 \times (3 \times 2)} \) for inferring \( p_{\text{pull}}, p_{\text{hold}} \) from a local knot crop of size \((80,60)\) (Fig. 4 D). Each network outputs 3 channels for each of the 2 keypoints. One channel is a predicted binary heatmap classifying each pixel as 1 if it lies within a radius \( R \) of a keypoint and 0 otherwise. This is intended to narrow the scope of prediction to a highly localized area around the desired point as shown in the white discs in Fig. 4 D. The remaining two channels separately regress the \( x \) and \( y \) offsets for each pixel classified as 1 relative to the point of interest. The final predicted keypoint is a pixel classified as 1 in the heatmap with the minimum predicted offset to the desired keypoint, shown in blue in the Local Offset subfigure of Fig. 4.

Provided a global image of the LDO, \( I \in \mathbb{R}^{640 \times 480 \times 3} \), HULK-G computes \( \hat{p}_l, \hat{p}_r, \hat{p}_{\text{pull}}, \hat{p}_{\text{hold}} = f_{\text{all}}[I] \). HULK-L computes \( \hat{p}_l, \hat{p}_r = f_{\text{reid}}[I] \), and predicts the rightmost bounding box \( b \) from \( g(I) \) (Fig. 4), consistent with the assumption that we always untangle by finding the first under-crossing relative to the right endpoint. Finally, HULK-L finds \( \hat{p}_{\text{pull}}, \hat{p}_{\text{hold}} = f_{\text{node}}[I[b']] \) locally within \( b' \), where \( b' \) is \( b \) resized to aspect ratio \((80,60)\).

### 6 Experiments

We experimentally evaluate HULK on the task of untangling LDOs in simulation to test its efficiency over methods which do not leverage geometric structure. We compare HULK-L and HULK-G
Figure 4: Perception Overview: We compare two implementations of HULK's perception (C, D) qualitatively against full state estimation approaches on an RGB image of an LDO in a highly deformed configuration (A). We train a dense object descriptor network [7, 13, 14] that learns a dense mapping to a unit-normalized 3D descriptor space visualized in RGB (B). The mapping fails to discriminate between overlapping segments, shown by the lack of color distinction around the knot, preventing reliable inference of task-relevant keypoints. In HULK-G (C), we train a network to output 4 Gaussians centered around $p_l, p_r, p_{pull}, p_{hold}$ (left to right). HULK-L (D) fuses global and local information from bounding box crops of knots to produce localized keypoint predictions.

Overview of Policies: All policies predict bounding boxes from RGB image inputs and use Eq. 3 as a termination condition. Each policy varies in its estimation of $p_l, p_r, p_{pull}, p_{hold}$ which are used to implement Reidemeister and Node Deletion moves. The Oracle baseline takes RGB images as input and implements BRUCE given full access to the ground-truth 3D LDO state in simulation. The Depth baseline requires RGB-D perception and makes the approximation that $p_{hold}$ is the highest depth pixel within $b$. This policy naively pulls slack 15 pixels to the left since we assume to always untangle relative to the right endpoint. The Random baseline requires both RGB and the LDO binary segmentation mask, and attempts to untangle by taking arbitrary actions sampled on the segmentation mask of the LDO within the bounding box of a predicted knot. HULK-G uses global keypoint regression as in Fig. 4 to detect endpoints and pull/hold locations to perform untangling from input RGB images. Lastly, HULK-L combines bounding box knot detection, global endpoint regression, and local pull/hold detection to perform untangling with only access to overhead RGB images.

6.1 Simulation Experiments

All manipulation experiments are performed in the simulated environment in Blender 2.8 [37] from Section 5. We assume access to observations from a synthetic overhead RGB camera and a mapping from pixel space to 3D world coordinates via projection. We further assume that the bilateral manipulation required for Reidemeister and Node Deletion moves can be performed without error. In the starting configuration, we assume that the endpoints are unoccluded to plan the initial Reidemeister move.

In HULK-L, for each of the three types of textures considered (capsule, smooth, and braid as in Fig. 1), we learn a separate bounding box knot-detection model, endpoint keypoint regression model, and local pull/hold keypoint regression model using the procedure described in Section 5. In HULK-G, we learn a separate bounding box knot-detection model and global keypoint regression model for endpoints and pull/hold locations for each of the three textures. We train each network on 3,500 rendered synthetic images of the appropriately textured LDO in randomized initial knotted configurations. We consider all combinations of manipulation policy, starting knot configuration, and texture. For each of these combinations, we run 21 trials strictly in simulation. Due to lab closures necessitated by COVID-19, physical experiments of HULK could not be safely performed before submission. However, since HULK is based on RGB image observations in a similar simulator and with similar domain-randomization techniques as used in work with successful sim2real [7, 33, 29], we expect that it can be extended to a real world setup in immediate follow-up work.

One property of HULK is that failure modes arise in separate components of the perception method allowing for more modular improvement approaches. In the bounding box detection module, false negatives can lead to premature termination of untangling. In HULK-L, since the pull/hold keypoint regression module is conditioned on an accurately predicted bounding box, erroneously predicted
Figure 5: Untangling Simulator Experiment Results: We report the untangling success rate and efficiency over 21 trials for each texture, initial configuration, and policy. We find that the bounding box knot detector performs best with the smooth texture relative to the braid and capsule textures because the knot geometry is easiest to perceive in the smoother texture. Additionally, both the global and local keypoint detectors perform worse on the smooth texture due to the lack of features along the LDO to disambiguate crossings. Bounding box dimensions or false positives can result in a crop that either does not include the relevant under-crossing or captures too large of a crop. Failures in the keypoint module typically occur in predicting a pull/hold pair rather than endpoints, when the network identifies keypoints local to a crossing that is not the next immediate under-crossing. Other failure modes include premature or late termination caused by a mispredicted action that erroneously triggers the termination condition.

We evaluate the performance of HULK relative to the other baselines by considering the success rate of untangling as a function of the number of actions taken (Fig. 5). We benchmark the HULK-L, HULK-G, Random, and Depth policies by comparing their performance relative to Oracle. HULK-L and HULK-G outperform the Random baseline across all textures and initial knot configurations with empirically higher efficiency and higher success, as shown in the steeper convergence of both HULK implementations compared to Random in all plots in Fig. 5. We also find that HULK-G and HULK-L match the performance of the Depth baseline in several trials. The comparable performance between HULK and Depth is notable since both HULK-L and HULK-G do not have access to depth observations, and the Depth baseline also has access to ground-truth information for the endpoints which is critical to preventing premature termination. The Depth baseline also naively pulls directly left in all cases, which is not camera pose agnostic, and depth provides a strong noise-free signal in simulation. We note HULK-L outperforms HULK-G in the 2 out of 3 multiple knot experiments as its hierarchical perception encourages the prediction of \( \hat{p} \text{pull} \) and \( \hat{p} \text{hold} \) around the correct crossing.

We also report that the duration of experimental trials is dominated by rendering times in Blender, suggesting that HULK could be executed at high frequency in physical trials. Bounding box inference takes approximately 300 ms and keypoint inference takes 2 ms.

7 Conclusion

We present HULK, a perception-based system learned in simulation from an algorithmic supervisor (BRUCE). HULK untangles dense overhand and figure-eight knots in linear deformable objects (LDOs) from RGB observations. HULK (1) exploits geometry at local and global scales and (2) learns to model only task-specific features, instead of performing full state estimation, to enable fine-grained manipulation. We present two variants of HULK, HULK-G and HULK-L, which learn task-specific bounding boxes and keypoints. These outputs are used to geometrically plan straightening and loosening actions. We evaluate the effectiveness of HULK against several baselines in simulation. Experiments suggest that HULK can more successfully and efficiently untangle LDOs consisting of varied textures and initial configurations compared to analytical approaches. In future work we will implement HULK on a physical system with the ABB YuMi robot which is capable of performing the required bilateral manipulation primitives. Additionally, we will explore extending HULK to a more general class of knots and initial configurations, and apply local and global reasoning and keypoint perception 2D and 3D deformable manipulation.
References


