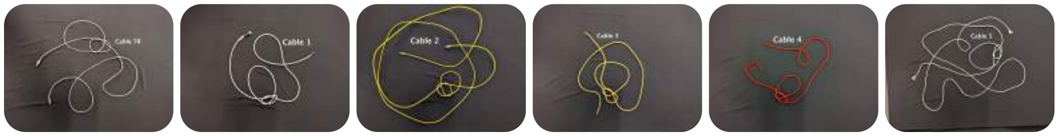


Figure 5: **Cables for Tracing Unseen Cables Experiment**

Table 6: Tracing on Unseen Cables Results

Cable Reference	TR	1	2	3	4	5	**Avg.**
Tracing Success Rate	6/8	7/8	8/8	7/8	6/8	6/8	**40/48=83%**
Failures	(I) 2	(I) 1		(II) 1	(I) 1, (III) 1	(II) 1, (III) 1	

7 Appendix

7.1 Experiments Failure Mode Analysis

7.1.1 Using LTODO for Tracing Cables Unseen During Training

(1) Retraces previously traced cable (went in a loop).

(2) Missteps onto a parallel cable.

(3) Skips a loop.

Figure 5 shows the cables tested on. The most common failure mode is (I), retracing previously traced cable. This is commonly observed in cases with near parallel segments or in dense loop areas within a knot.

7.1.2 Using LTODO for Cable Inspection in Multi-Cable Settings

(I) Misstep in the trace, i.e. the trace did not reach any adapter.

(II) The trace reaches the wrong adapter.

(III) The trace reaches the correct adapter but is an incorrect trace.

The most common failure mode for the learned tracer, especially in Tier A3, is (I). One reason for such failures is the presence of multiple twists along the cable path (particularly in Tier A3 setups, which contain more complex inter-cable knot configurations). The tracer is also prone to deviating from the correct path on encountering parallel cable segments. In Tier A2, we observe two instances of failure mode (III), where the trace was almost entirely correct in that it reached the correct adapter but skipped a section of the cable.

The most common failure modes across all tiers for the analytic tracer are (II) and (III). The analytic tracer particularly struggles in regions of close parallel cable segments and twists. As a result of the scoring metric, 87 of the 90 paths that we test reach an adapter; however, 45/90 paths did not reach the correct adapter. Even for traces that reach the correct adapter, the trace is incorrect, jumping to other cables and skipping sections of the true cable path.

7.1.3 Using LTODO for Physical Robot Knot Tying from Demonstrations

(1) Trace missteps onto a parallel cable.

(2) Cable shifted during manipulation, not resulting in a knot at the end.

Table 7: Multi-Cable Tracing Results

	Analytic	Learned
Tier A1	3/30	**27/30**
Tier A2	2/30	**23/30**
Tier A3	1/30	**23/30**
Failures	(I) 3, (II) 45, (III) 36	(I) 14, (II) 1, (III) 2

Table 8: Learning From Demos

	Succ. Rate	Failures
Tier B1	5/5	-
Tier B2	4/5	(1) 1
Tier B3	4/5	(1) 1, (2) 1

Table 9: LTODO Experiments

	SGTM 2.0	LTODO (-LT)	LTODO (-CC)	LTODO
Tier C1	2/30	14/30	20/30	**24/30**
Tier C2	**28/30**	8/30	21/30	26/30
Tier C3	12/30	14/30	0/30	**19/30**
Failures	(A) 30, (B) 18	(D) 11, (F) 7 (G) 24, (H) 11	(B) 38, (C) 5, (E) 6	(B) 11, (D) 8 (F) 1

Table 10: LTODO and Physical Robot Experiments (90 total trials)

	Tier D1		**Tier D2**		**Tier D3**	
	SGTM 2.0	LTODO	SGTM 2.0	LTODO	SGTM 2.0	LTODO
Knot 1 Succ.	11/15	**12/15**	6/15	**11/15**	9/15	**14/15**
Knot 2 Succ.	-	-	-	-	2/15	**6/15**
Verif. Rate	**11/11**	8/12	**6/6**	6/11	**1/2**	2/6
Knot 1 Time (min)	1.1±0.1	2.1±0.3	3.5±0.7	3.9±1.1	1.8±0.4	2.0±0.4
Knot 2 Time (min)	-	-	-	-	3.1±1.2	7.5±1.6
Verif. Time (min)	5.7±0.9	6.1±1.4	6.4±1.8	10.1±0.7	5.4	9.6±1.5
Failures	(7) 4	(1) 2, (2) 1 (1) 2, (2) 1	(1) 3, (5) 6 (1) 3, (5) 6	(2) 2, (4) 1 (5) 1	(1) 3, (2) 3, (5) 3 (6) 2, (7) 2	(1) 2, (2) 3 (3) 1, (6) 3

Failure mode (1) occurs when the distractor cable creates near parallel sections to the cable of interest for knot tying, causing the trace to misstep. Failure mode (2) occurs when the manipulation sometimes slightly perturbs the rest of the cable's position while moving one point of the cable, causing the end configuration to not be a knot, as intended.

7.1.4 Using LTODO for Knot Detection

(A) The system fails to detect a knot that is present—a false negative.

(B) The system detects a knot where there is no knot present—a false positive.

(C) The tracer retraces previously traced regions of cable.

(D) The crossing classification and correction schemes fail to infer the correct cable topology.

(E) The knot detection algorithm does not fully isolate the knot, also getting surrounding trivial loops.

(F) The trace skips a section of the true cable path.

(G) The trace is incorrect in regions containing a series of close parallel crossings.

(H) The tracer takes an incorrect turn, jumping to another cable segment.

For SGTM 2.0, the most common failure modes are (A) and (B), where it misses knots or incorrectly identifies knots when they are out of distribution. For LTODO (-LT), the most common failure modes are (F), (G), and (H). All 3 failures are trace-related and result in knots going undetected or being incorrectly detected. For LTODO (-CC), the most common failure modes are (B) and (E). This is because LTODO (-CC) is unable to distinguish between trivial loops and knots without the crossing cancellation scheme. By the same token, LTODO (-CC) is also unable to fully isolate a knot from surrounding trivial loops. For LTODO, the most common failure mode is (B). However, this is a derivative of failure mode (D), which is present in LTODO (-LT), LTODO (-CC), and LTODO. Crossing classification is a common failure mode across all systems and is a bottleneck for accurate knot detection. In line with this observation, we hope to dig deeper into accurate crossing classification in future work.

7.1.5 Using LTODO for Physical Robot Untangling

(1) Incorrect actions create a complex knot.

(2) The system misses a grasp on tight knots.
(3) The cable falls off the workspace.
(4) The cable drapes on the robot, creating an irrecoverable configuration.
(5) False termination.
(6) Manipulation failure.
(7) Timeout.

The main failure modes in LTODO are (1), (2), and (6). Due to incorrect cable topology estimates, failure mode (1) occurs: a bad action causes the cable to fall into complex, irrecoverable states. Additionally, due to the limitations of the cage-pinch dilation and endpoint separation moves, knots sometimes get tighter during the process of untangling. While the perception system is still able to perceive the knot and select correct grasp points, the robot grippers bump the tight knot, moving the entire knot and causing missed grasps (2). Lastly, we experience manipulation failures while attempting some grasps as the YuMi has a conservative controller (6). We hope to resolve these hardware issues in future work.

The main failure modes in SGTM 2.0 are (5) and (7). Perception experiments indicate that SGTM 2.0 has both false positives and false negatives for cable configurations that are out of distribution. (5) occurs when out-of-distribution knots go undetected. (7) occurs when trivial loops are identified as knots, preventing the algorithm from terminating.

7.2 Details on LTODO Methods

7.2.1 Over/Undercrossing Predictor

Model Architecture and Inference: The binary classification threshold of 0.275 is determined by testing accuracy on a held-out validation set of 75 images on threshold values in the range $[0.05, 0.95]$ at intervals of 0.05. Scores < 0.275 indicate undercrossing predictions and scores ≥ 0.275 indicate overcrossing predictions. We output the raw prediction score and a scaled confidence value (0.5 to 1) indicating the classifier's probability.

7.3 Details on Robot Untangling using LTODO

7.3.1 Knot Definition

Consider a pair of points p_1 and p_2 on the cable path at time t with $(p_1, p_2 \in \mathcal{C}_t)$. Knot theory strictly operates with closed loops, so to form a loop with the current setup, we construct an imaginary cable segment with no crossings joining p_1 to p_2 [44]. This imaginary cable segment passes above the manipulation surface to complete the loop between p_1 and p_2 ("$p_1 \rightarrow p_2$ loop"). A knot exists between p_1 and p_2 at time t if no combination of Reidemeister moves I, II (both shown in Figure 6), and III can simplify the $p_1 \rightarrow p_2$ loop to an unknot, i.e. a crossing-free loop. In this paper, we aim to untangle semi-planar knots. For convenience, we define an indicator function $k(s) : [0, 1] \rightarrow \{0, 1\}$ which is 1 if the point $\theta(s)$ lies between any such points p_1 and p_2, and 0 otherwise.

Based on the above knot definition, this objective is to remove all knots, such that $\int k(s)_0^1 = 0$. In other words, the cable, if treated as a closed loop from the endpoints, can be deformed into an unknot. We measure the success rate of the system at removing knots, as well as the time taken to remove these knots.

7.3.2 State Definition

We construct line segments between consecutive points on the trace outputted by the learned cable tracer (Section 4.1). Crossings are located at the points of intersection of these line segments. We use the crossing classifier (Section 4.2) to estimate whether these crossings are over/undercrossings. We also implement probabilistic crossing correction with the aim of rectifying classification errors, as we describe in Section 4.2.2.

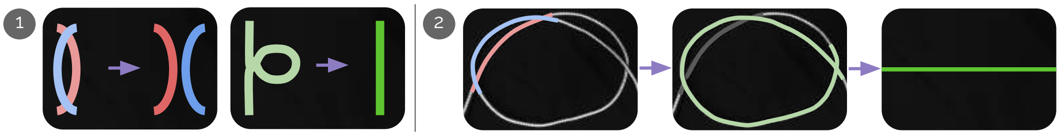


Figure 6: **Reidemeister Moves and Crossing Cancellation**: Left of part 1 depicts Reidemeister Move II. Right of part 1 depicts Reidemeister Move I. Part 2 shows that by algorithmically applying Reidemeister Moves II and I, we can cancel trivial loops, even if they visually appear as knots.

We denote the sequence of corrected crossings, in the order that they are encountered in the trace, by $\mathcal{X} = (c_1, ..., c_n)$, where n is the total number of crossings and $c_1, ..., c_n$ represent the crossings along the trace.

7.3.3 Crossing Cancellation

Crossing cancellation allows for the simplification of cable structure by removing non-essential crossings, shown in Figure 6. It allows the system to filter out some trivial configurations as Reidemeister moves maintain knot equivalence [44]. We cancel all pairs of consecutive crossings (c_i, c_{i+1}) in $\mathcal{X}$ for some j) that meet any of the following conditions:

- *Reidemeister I:* c_i and c_{i+1} are at the same location, or
- *Reidemeister II:* c_i and c_{i+1} are at the same set of locations as c_j and c_{j+1} ($c_j, c_{j+1} \in \mathcal{X}$). Additionally, c_i and c_{i+1} are either both overcrossings or both undercrossings. We also cancel (c_j, c_{j+1}) in this case.

We algorithmically perform alternating Reidemeister moves I and II as described. We iteratively apply this step on the subsequence obtained until there are no such pairs left. We denote the final subsequence, where no more crossings can be canceled, by $\mathcal{X}'$.

7.3.4 Knot Detection

We say that a subsequence of $\mathcal{X}'$, $\mathcal{K}_{ij} = (c_i, ..., c_j)$, defines a potential knot if:

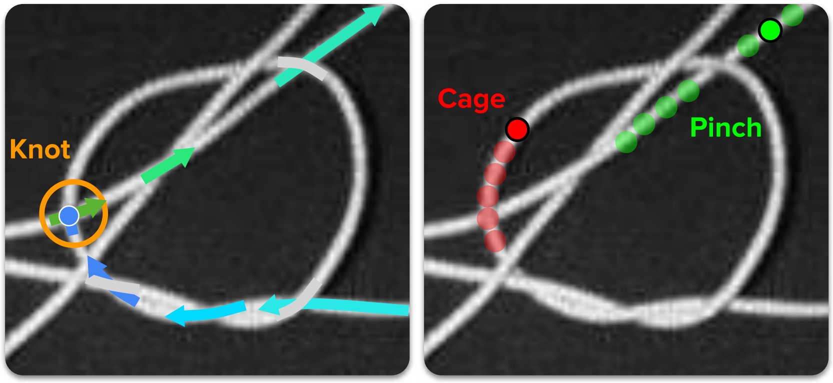
- c_i is an undercrossing, and
- c_j is an overcrossing at the same location, and
- at least one intermediate crossing, i.e. crossing in $\mathcal{X}'$ that is not c_i or c_j, is an overcrossing.

The first invariant is a result of the fact that all overcrossings preceding the first undercrossing (as seen from an endpoint) are removable. We can derive this by connecting both endpoints from above via an imaginary cable (as in Section 7.3.1): all such overcrossings can be removed by manipulating the loop formed. The second invariant results from the fact that a cable cannot be knotted without a closed loop of crossings. The third and final invariant can be obtained by noting that a configuration where all intermediate crossings are undercrossings reduces to the unknot via the application of the 3 Reidemeister moves. Therefore, for a knot to exist, it must have at least one intermediate overcrossing.

Notably, these conditions are necessary, but not sufficient, to identify knots. However, they improve the likelihood of bypassing trivial configurations and detecting knots. This increases the system's efficiency by enabling it to focus its actions on potential knots.

7.3.5 Algorithmic Cage-Pinch Point Detection

As per the definition introduced in Section 7.3.4, given knot $\mathcal{K}_{ij} = (c_i, ..., c_j)$, c_i and c_j define the segments that encompass the knot where c_i is an undercrossing and c_j is an overcrossing for the same crossing. The pinch point is located on the overcrossing cable segment, intended to increase space for the section of cable and endpoint being pulled through. The cage point is located on the



Knot Detection & Topology   Cage-Pinch Points

Figure 7: **Knot Detection and Cage Pinch Point Selection**: The left image shows using crossing cancellation rules from knot theory, the knot detection algorithm analytically determines where the knot begins in the cable. The right image shows the survey process for selecting the cap pinch points.

undercrossing cable segment. To determine the pinch point, we search from crossing c_{u1} to crossing c_{u2}. c_{u1} is the previous undercrossing in the knot closest in the trace to j. $u2 > j$ and c_{u2} is the next undercrossing after the knot. We search in this region and select the most graspable region to pinch at, where graspability (G) is defined by the number of pixels that correspond to a cable within a given crop and a requirement of sufficient distance from all crossings c_i. To determine the cage point, we search from crossing c_i to c_k where $i < k < j$ and c_k is the next undercrossing in the knot closest in the trace to c_i. We similarly select the most graspable point. If no points in the search space for either the cage or pinch point are graspable, meaning $G < \mathcal{T}$ where $\mathcal{T}$ is an experimentally derived threshold value, we continue to step along the trace from c_{u2} for pinch and from c_k for cage until $G \geq \mathcal{T}$. This search process is shown in Figure 7.

7.3.6 Manipulation Primitives

We use the same primitives as in SGTM 2.0 (Sliding and Grasping for Tangle Manipulation 2.0) [6] to implement LTODO as shown in Figure 8 for untangling long cables. We add a *perturbation* move.

Cage-Pinch Dilation: We use cage-pinch grippers as in Viswanath et al. [5]. We have one gripper cage grasp the cable, allowing the cable to slide between the gripper fingers but not slip out. The other gripper pinch grasps the cable, holding the cable firmly in place. This is crucial for preventing knots in series from colliding and tightening during untangling. The *partial* version of this move introduced by Shivakumar et al. [6] separates the grippers to a small, fixed distance of 5 cm.

Reveal Moves: First, we detect endpoints using a Mask R-CNN object detection model. If both endpoints are visible, the robot performs an *Endpoint Separation Move* by grasping at the two endpoints and then pulling them apart and upwards, away from the workspace, allowing gravity to help remove loops before placing the cable back on the workspace. If both endpoints are not visible, the robot performs an *Exposure Move*. This is when it pulls in cable segments exiting the workspace. Building on prior work, we add a focus on where this move is applied. While tracing, if we detect the trace hits the edge, we perform an exposure move at the point where the trace exits the image.

Perturbation Move: If an endpoint or the cable segment near an endpoint has distracting cable segments nearby, making it difficult for the analytic tracer to trace, we perturb it by grasping it and translating in the x-y plane by uniformly random displacement in a 10cm $\times$ 10cm square in order to separate it from slack.

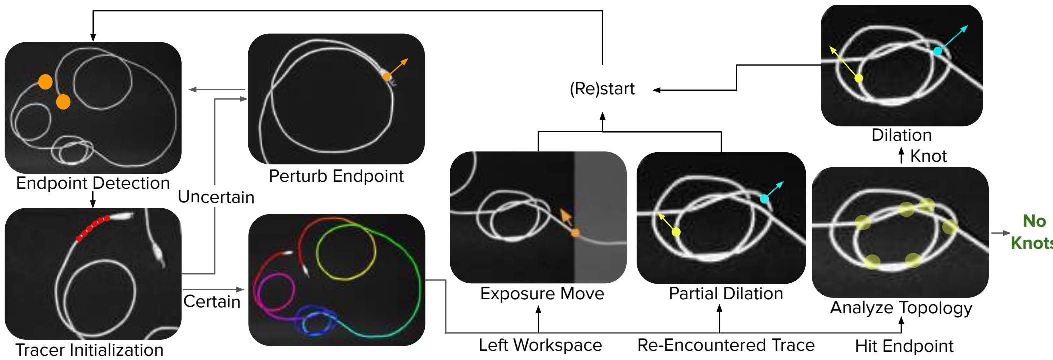


Figure 8: **Untangling Algorithm with LTODO**: We first detect the endpoints and initialize the tracer with start points. If we are not able to obtain start points, we perturb the endpoint and try again. Next, we trace. While tracing, if the cable exits the workspace, we pull the cable towards the center of the workspace. If the tracer gets confused and begins retracing a knot region, we perform a partial cage-pinch dilation that will loosen the knot, intended to make the configuration easier to trace on the next iteration. If the trace is able to successfully complete, we analyze the topology. If there are no knots, we are done. If there are knots, we perform a cage-pinch dilation and return to the first step.

7.3.7 Cable Untangling System

Combining LTODO and the manipulation primitives from Section 7.3.6, the cable untangling algorithm works as follows: First, detect endpoints and initialize the learned tracer with 6 steps of the analytic tracer. If LTODO is unable to get these initialization points, perturb the endpoint from which we are tracing and return to the endpoint detect step. Otherwise, during tracing, if the cable leaves the workspace, perform an exposure move. If the trace fails and begins retracing itself, which can happen in denser knots, perform a partial cage-pinch dilation as in [6]. If the trace completes and reaches the other endpoint, analyze the topology. If knots are present, determine the cage-pinch points for it, apply a cage-pinch dilation move to them, and repeat the pipeline. If no knots are present, the cable is considered to be untangled. The entire system is depicted in Figure 8.

7.4 Details on Learning from Demos with LTODO

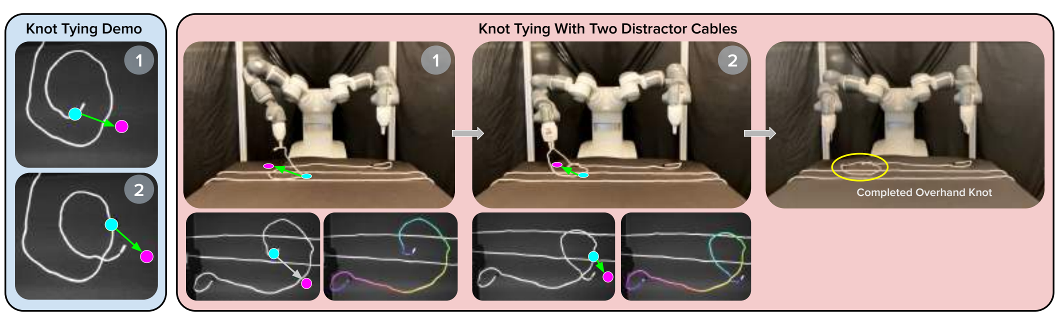


Figure 9: **Using LTODO for Learning from Demos:** The left (blue panel) displays the single human demonstration, indicating the pick and place points for tying an overhand knot. The right (pink panel) shows this demonstration successfully applied to the cable in a different configuration with 2 other distractor cables in the scene. The first step of the demonstration is achieved through an arc length relative action while the second step is achieved through a crossing relative action.

When performing state-based imitation, each of the pick and place points p_i from the demonstration is parameterized in the following way: 1) find the point along the trace, T, closest to the chosen point $\hat{p_i}$ with index j in T, 2) find the displacement $d_i = p_i - \hat{p_i}$ in the local trace-aligned coordinate system of $\hat{p_i}$, 3) in memory, for point p_i, store d_i, arc length of $\hat{p_i}$ ($\sum_{x=1}^{j} T_x - T_{x-1}$), and the index value of the crossing in the list of crossings just before $\hat{p_i}$.

When rolling out a policy using this demonstration, there are two ways to do so: 1) relative to the arc length along the cable, or 2) relative to the fraction of the arc length between the 2 crossing indices. The way to do so is to find the point on the cable with the same arc length as $\hat{p}_i$ from the demo or the fractional arc length between the same 2 crossing indices, depending on the type of demonstration. Then, apply d_i in the correct trace-aligned coordinate system. An example demonstration is shown in Figure 9.

Additional conceivable ways, not explored in this work, include relative to the location of a knot along the cable or others.