Generalizing Perception Systems Applied to Untangling Long Cables

Vainavi Viswanath

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2023-95
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2023/EECS-2023-95.html

May 11, 2023

Acknowledgement

Thank you to Professor Ken Goldberg for the opportunities and mentorship over the past 3 years.

To Kaushik Shivakumar for the collaboration and making research fun.

To all my mentors for their invaluable advice: Jennifer Grannen, Priya Sundaresan, Justin Kerr, Brijen Thananjeyan, Ashwin Balakrishna, Ellen Novoseller, and Jeff Ichnowski.

To all my collaborators: Mallika Parulekar, Jainil Ajmera, and Anrui Gu, Vincent Schorp, and Satvik Sharma.

To all my friends at Berkeley for all the laughs, advice, and sweet memories.

To my parents for their unwavering love and support.

Generalizing Perception Systems Applied to Untangling Long Cables

by

Vainavi Viswanath

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Ken Goldberg, Chair
Professor Joseph E. Gonzalez

Spring 2023

Generalizing Perception Systems Applied to Untangling Long Cables

by Vainavi Viswanath

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

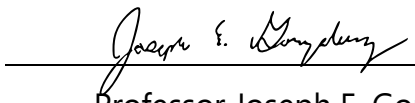
Committee:

Professor Ken Goldberg
Research Advisor

10 May 2023

(Date)

* * * * * * *



Professor Joseph E. Gonzalez
Second Reader

5/11/2023

(Date)

Generalizing Perception Systems Applied to Untangling Long Cables

Abstract

Generalizing Perception Systems Applied to Untangling Long Cables

by

Vainavi Viswanath

Master of Science in Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Ken Goldberg, Chair

Long cables are widely found in domestic and industrial settings [30, 39, 51]. These linear deformable objects form knots that hinder functionality and pose safety risks. Untangling them is crucial for restoring proper cable function and enabling downstream tasks like surgical suturing and automotive wire harness assembly. However, untangling long cables is challenging due to their infinite-dimensional state-space, self-occlusion, and knot formation [12, 28, 45, 46, 52]. Longer cables are even more prone to knotting, and as one segment is untangled, the remaining segments, known as "slack," can create new knots.

To address these challenges, our research focuses on developing RGB perception and motion primitives, along with specialized gripper jaws, for efficient untangling of long cables. Our algorithm utilizes these advancements to iteratively untangle cables, achieving success rates of 67% for isolated knots and 50% for complex configurations.

In the project's initial phase, we identified that failure primarily stems from drastic actions in uncertain states, resulting in irrecoverable cable states. We introduce novel metrics and cable-interacting actions that reduce perception uncertainty before untangling maneuvers. This integrated system accommodates variations in cable states, achieving an 83% success rate in untangling cables with one or two knots and a 70% termination detection success across these configurations. Our method demonstrates a 43% improvement in untangling accuracy and completes the task three times faster compared to the previous system.

Despite these improvements, the system still fails to untangle cables in out of distribution states. We develop a generalized perception approach using a novel cable state estimator, including a learning-based iterative tracer, crossing classifier, and crossing correction algorithm for semi-planar knots. Implementing this system leads to a 13% improvement in untangling success for complex knots. Additionally, our method demonstrates versatility by achieving an 81% success rate in multi-cable tracing. Furthermore, it handles variations in cable appearances, tracing cables not in the training dataset with an 85% success.

To my parents, for their undying and unconditional love and support.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to express my sincere gratitude to Professor Ken Goldberg for welcoming me into AUTOLab when I was a naive freshman at UC Berkeley. Professor Goldberg has been instrumental in providing me with countless opportunities, unwavering encouragement, and steadfast support, allowing me to grow as a researcher. Though I have much more to discover and learn, I will forever cherish Professor Goldberg's guidance in my early steps within the realm of robotics research. His boundless creativity and unwavering passion for robotics serve as an ongoing inspiration to me.

I would also like to thank my first mentors, Jennifer Grannen and Priya Sundaresan. Despite my limited research experience, Jennifer and Priya graciously welcomed me onto their project, and their guidance played a pivotal role in shaping me into a stronger researcher.

I am immensely grateful to all my advisors, Justin Kerr, Brijen Thananjeyan, Ashwin Balakrishna, Ellen Novoseller, and Jeff Ichnowski, for their invaluable insights, meaningful discussions, and technical expertise.

Collaborating with Kaushik Shivakumar on several papers has been both enjoyable and enlightening. I have learned so much from him and I am continually inspired by his unwavering dedication to embracing new challenges. Thank you, Kaushik, for being an exceptional collaborator and an even greater friend.

I would also like to express my appreciation to all my collaborators: Mallika Parulekar, Jainil Ajmera, Anrui Gu, Vincent Schorp, and Satvik Sharma. Working with each of you has taught me a lot and it has been a truly wonderful experience.

Thank you to my friends at AUTOLab for the spontaneous and engaging discussions, spirited frisbee matches as part of the lab IM team, and enjoyable movie nights. Your camaraderie and shared experiences have made my time in the lab truly memorable.

Lastly, I extend my heartfelt gratitude to my parents for their unwavering support and constant encouragement throughout my academic journey. They have been my pillars of strength, uplifting me during moments of disappointment and celebrating my victories with unwavering enthusiasm.

Chapter 1

Introduction

Long cables are commonplace in household and industrial settings, from wires in homes to cables in manufacturing plants to ropes in sailing [30, 39, 51, 60]. Cables—defined here as single-dimensional deformable objects—often tangle and form knots, which can be unsightly, unsafe, and reduce utility. However, autonomously manipulating cables to untangle them is challenging due to their infinite-dimensional configuration spaces and tendency to form self-occlusions and knots [12, 28, 45, 46, 52]. These challenges grow with increasing cable length as longer cables allow more knots to form. Additionally, as robots untangle one segment of a cable, the other cable segments, the *slack*, can form new knots, occlude visibility, or impede grasping. The stiffness of cables further adds to the slack management challenge, because untangled slack may appear to hold a knot shape even without a knot present. Robustly manipulating long cables requires policies that perceive cable configurations and effectively manage slack during manipulation.

Prior work in robot cable untangling considers short cables and studies how to untangle dense knots, which have no open space between adjacent cable segments [12, 46, 52], or considers longer cables, estimated to be no longer than one meter in length, whose paths can be clearly traced using analytic methods [28].

This thesis studies 3 levels of projects working towards generalizing perception system applied to untangling long cables. First, this work explores untangling overhand and figure 8 knots in cables up to 3 m long, using active slack management during manipulation to prevent the formation of new knots and crossings. This is covered in Chapter 3 and discusses [53] which was accepted into RSS 2022 and won the Best Systems Paper Award.

Because the first approach lacks uncertainty awareness and takes actions that are often overly aggressive or conservative, the next chapter in this thesis focuses on quantifying uncertainty to enable applying interactive perception [4], which involves physically manipulating objects in a scene to better understand it, allowing the system to accommodate variations in cable state. This is covered in Chapter 4, which discusses [44], accepted in ICRA 2023.

These prior two approaches are still limited as they only address 2 categories of knots: overhand and figure 8 knots. Additionally, the system still failed to untangle cables within the time limit when out of distribution. To generalize untangling, this thesis expands to

perform full cable state estimation for semi-planar cable configurations, i.e. each crossing consists of at most 2 cable segments when viewed from above. This allows us to consider semi-planar knots. The single-cable semi-planar knots specifically tested in this final chapter of the thesis are overhand, figure 8, overhand honda, bowline, linked overhand, and figure 8 honda knots. The double-cable semi-planar knots considered in this work are carrick bend, sheet bend, and square knots. Besides untangling, this work highlights a high-accuracy state estimation technique for 1D deformable objects that can be applied to downstream tasks. This is covered in Chapter 5 and discusses [54].
This thesis makes the following contributions:

1. A novel untangling algorithm, Sliding and Grasping for Tangle Manipulation (SGTM) 1.0, to untangle cables of up to 3 meters in length from starting configurations that contain up to 2 overhand and figure 8 knots. The algorithm consists of three new bilateral manipulation primitives for cable untangling and disambiguating made possible through cage-pinch grippers introduced in this work: shaking, tracing, and dual-cage separation. (Chapter 3)

2. Novel perception-based metrics to estimate uncertainty in cable configurations, including tracing, network, and observational uncertainties. These are coupled with new primitives, including interactive perception actions, for cable slack management, untangling, and termination. This system allows for robustness to variations in cable state. The full algorithm is referred to as Sliding and Grasping for Tangle Manipulation (SGTM) 2.0. (Chapter 4)

3. Tracing to Untangle Semi-planar Knots (TUSK): A novel cable state estimator consisting of a learning-based iterative tracer and a crossing classifier with a crossing correction algorithm. To the best of our knowledge, this is the first work to conduct full state estimation on long, dense cables. This work also contributes untangling specific contributions: an analytic knot detection algorithm and analytic untangling point selection algorithm given the cable state estimate. (Chapter 5)

Chapter 2

Related Work

2.1 Deformable Object Manipulation

Robot manipulation of deformable objects, such as cables (1D), fabric (2D), and bags (3D), is difficult because they have a near-infinite state space, can form self-occlusions, and are difficult to model. This study focuses on the problem of untangling knots in long cables. Recently, there has been progress in deformable manipulation, including algorithms for untangling cables [12, 28, 46, 52], smoothing and folding fabric [10, 14, 15, 21, 41, 49, 56], and placing objects into bags [5, 42].

Methods for intelligently and autonomously manipulating deformable objects lie on a spectrum ranging from completely model-free, directly perception-driven approaches to those that directly estimate the state of the object of interest and then perform planning on it.

Examples of model-based methods for deformable objects are dense descriptors [9], which have been applied to cable knot tying [45] and fabric smoothing [10], as well as visual dynamics models for non-knotted cables [55, 61] and fabric [14, 24, 61]. Model-free approaches include reinforcement or self-supervised learning for fabric smoothing and folding [1, 22, 29, 58] and straightening curved ropes [58], or directly imitating human actions [41].

2.2 Interactive and Active Perception

In 1984, Goldberg and Bajcsy [11] explored active perception of shape using a robot to actively move a touch sensor to trace object contours. In 1988, Bajcsy [2] defined *active perception* as a search of models and control strategies for perception. Strategies vary according to the sensor and the task goal, including controlling camera parameters [3] and moving a tactile sensor according to haptic input [11]. Recently, Bohg et al. [4] explore the differences between *active* and *interactive* perception, the latter of which specifically exploits environment interactions to simplify and enhance perception to achieve a better understanding of the scene [4, 33].

Within robotic manipulation, several works have focused on improving understanding of the environment through scene interaction. Tsikos and Bajcsy [50] propose interacting with random heaps of unknown objects through pick and push actions for scene segmentation. Danielczuk et al. [8] present the mechanical search problem, where a robot locates and retrieves an occluded target object from a cluttered bin through a series of targeted parallel jaw grasps, suction grasps, and pushes. Novkovic et al. [33] propose a combination of camera motions with environment interactions to find a target cube hidden in a pile of cubes.

Interactive perception has also been applied to deformable manipulation. Willimon et al. [57] interact with a pile of laundry to isolate and identify individual clothing items. In our work, the robot interacts with the cable to reveal more information about the cable state.

2.3 Cable Untangling

Pioneering research in untangling, such as that conducted by Lui and Saxena [27], relies on decomposing point clouds of rope into segments which are refined into a graphical representation of the cable's structure based on priors on cable behavior such as bending radius. Other methods learn visual models for cable manipulation over which to plan [32], investigate iteratively refining dynamic actions [6], or use approximate state dynamics along with a learned error function [31]. Fusing point clouds across time has shown success in tracking segments of cable provided they are not tangled on themselves [40, 47].

Studies on dense knots, such as those by Grannen et al. [12] and Sundaresan et al. [46], employ learning-based keypoint detection to parameterize action primitives for untangling isolated knots.

Prior cable state estimation work includes that led by Huang et al. [16], which estimates the topological state of multiple ropes against varying backgrounds and uses primitives to untangle rope configurations consisting primarily of loops with 2 to 4 crossings. Additional linear deformable tracing work includes that of Keipour et al. [19], which models cables as chains of jointed cylindrical bodies, then uses the model for routing tasks. The works of Jackson et al. [18] and Padoy et al. [34] trace surgical strings in stereo or mono images by optimizing a continuous spline representation. In contrast, this work primarily focuses on much longer cables with a greater variety of configurations, for which analytical methods struggle to differentiate nearby, twisted cables. Some prior work approaches this problem [35] but does not fully estimate cable state, only identifying crossings.

Chapter 3

SGTM 1.0

3.1 Problem Statement

The overarching goal across Chapters 3, 4, and 5 is untangling long cables with dense and loose knot configurations. In this section we will formalize the problem statement.

Workspace Definition and Assumptions

We define an (x, y, z) Cartesian coordinate frame containing a bilateral robot and a flat manipulation surface that is parallel to the xy-plane. We assume an overhead RGB-D camera facing the manipulation surface from above and the rigid transformations between the camera, robot, and workspace coordinate frames are known. Because the RGBD camera uses structured light, we assume the workspace must be static during image capture.

At each iteration of a rollout ($t \in 0, 1, ...N$), the manipulation workspace contains an incompressible cable $\mathcal{C}$ with cross-section radius r and length l, which traces a continuous volume $c_t(s) : [0, 1] \rightarrow (x, y, z)$ in the workspace. The parameter $s \in [0, 1]$ is used to index into the cable path (with length normalized to 1), with $c_t(s)$ corresponding to the center of the circular cable cross-section with distance sl from the endpoint defined as the first endpoint. The circular slice lies in the plane orthogonal to the cable's local direction, $\nabla_s c_t(s)$.

We assume that the cable can be segmented from the manipulation surface using color thresholding and that in the starting configuration c_0, the entire cable is contained in the workspace. The initial cable configuration can form self-crossings, which could include knots or loops. A cable segment contains at least one *knot* if pulling it taut by pulling the two endpoints in opposite directions does not result in a straight cable, i.e. one with no crossings. We assume that any knots in the initial configuration are either overhand or figure-eight knots (Fig. 3.1), and that sufficient space within each knot exists to fit both robot jaws inside. We assume that individual knots are distinct, such that no knots are embedded within other knots. We classify the initial cable configurations we consider into tiers of difficulty in Section 3.3.

Figure 3.1: **Knots that SGTM 1.0 Untangles**: Left: an overhand knot. Right: a figure 8 knot.

Knot Definition

Let $a, b \in [0, 1]$ be two indices on the cable's path, where $a < b$. We say that a knot exists at time t between indices a and b if grasping and pulling apart the points $c_t(a)$ and $c_t(b)$ does not straighten the cable segment (i.e., by removing all crossings) between a and b. We let $k(a, b; c_t) = 1$ if a knot exists between a and b on cable c_t, otherwise $k(a, b; c_t) = 0$. For a particular cable configuration c_t, we can represent its knot structure by the minimal set of intervals $\mathcal{K}(c_t) = \{(a_1, b_1), \ldots, (a_n, b_n)\}$ containing a knot:

$$
\begin{aligned}
\mathcal{K}(c_t) = \text{argmin}_{\mathcal{K}(c_t) \in \bigcup_{k=0}^{\infty}([0,1]\times[0,1])^k} & \sum_{(a_i,b_i)\in\mathcal{K}(c_t)} |b_i - a_i| \\
\text{s.t.} \quad & k(a_i, b_i; c_t) = 1, \ \forall i \in \{1, \ldots, |\mathcal{K}(c_t)|\}, \\
& a_i < b_i \ \forall \ i; \ \ b_i \leq a_{i+1} \ \forall \ i, \\
& k(\tilde{a}, \tilde{b}; c_t) = 0 \ \forall \ \tilde{a}, \tilde{b} \\
& \quad \text{s.t.} \quad b_i \leq \tilde{a} \leq \tilde{b} \leq a_{i+1}, \ b_0 := 0, a_{|\mathcal{K}(c_t)|+1} := 1.
\end{aligned}
$$

This definition finds the smallest interval surrounding each distinct knot in the cable. Specifically, it enforces that a knot exists between each (a_i, b_i) pair and also that no knots are excluded, meaning $k(\tilde{a}, \tilde{b}; c_t) = 0$ between successive knots. For a given cable configuration c_t, the number of distinct knots is the number of intervals in $\mathcal{K}(c_t)$. This notation is illustrated in Figure 3.2 for a single figure-eight knot on a cable.

Task Objective

The goal of the cable untangling problem is to manipulate the cable into a configuration c_t such that $\mathcal{K}(c_t) = \emptyset$, indicating that no knots exist in the cable. We call such a configuration *untangled*.

Algorithm Inputs and Outputs

The algorithm is provided with overhead RGBD images as input at each iteration and it outputs either an open-loop trajectory to execute with the bilateral robot or a termina-

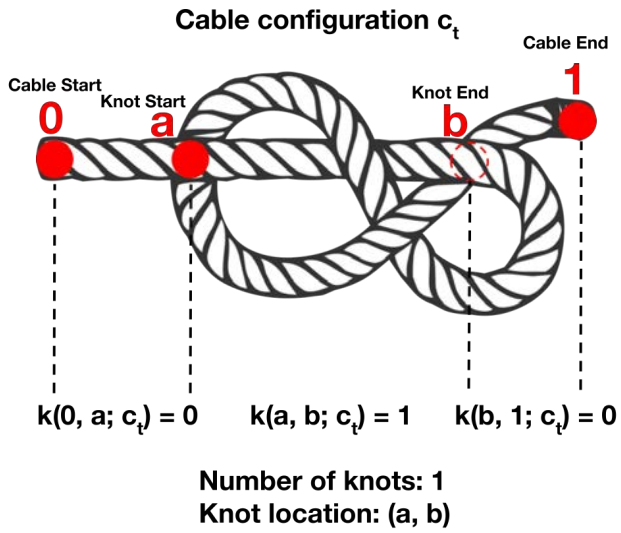


Figure 3.2: **Knot Definition on Figure-Eight Knot**: for a cable configuration at time t, c_t, a knot exists between two points (a, b) if pulling apart at those points will not result in a straight cable. In this example, no knots exist in $(0, a)$ and $(b, 1)$; thus, $k(0, a; c_t) = 0$ and $k(b, 1; c_t) = 0$. A figure-eight knot exists between indices a and b, and so $k(a, b; c_t) = 1$. Observe that (a, b) is the smallest interval that contains the knot, such that $\mathcal{K}(c_t) = \{(a, b)\}$.

tion signal. To output a termination signal, the robot must detect that it has successfully untangled the cable.

Performance Metrics

We record the success rate across three difficulty tiers of problem instances, as described in Section 3.3. We also report the time taken for the algorithm to terminate in each trial, as well as the time taken to first reach an untangled state, which may differ from the total time if the algorithm takes additional time to identify that it has fully untangled the cable.

3.2 Methods

SGTM 1.0 (Figure 5.5) proceeds in iterations consisting of two phases. In the first phase, SGTM 1.0 actively manipulates the cable to identify knots by pulling apart the endpoints, shaking the cable, and tracing along the cable physically. In the second phase, it identifies exposed knots and undoes them using a novel bilateral manipulation primitive that cages the knot and slides it apart. In this section we will describe *1)* the novel gripper jaw design, which allows multiple grasping modes with a compact form factor and no additional moving parts; *2)* the manipulation primitives used as sub-components in SGTM 1.0; *3)* the perception systems used, and *4)* how SGTM 1.0 uses these components to untangle long cables.

Figure 3.3: **Pinch-Cage (PC) Jaws**. Left: a rendering of the design of the pinch-cage gripper jaws, which attach to the YuMi's standard white gripper. Center: the cage grasp, in which the perpendicular "foot" segment of the attachment enables the cable to slide freely. Right: the pinch grasp, in which the cable is held tightly, preventing slack from slipping through.

Pinch-Cage (PC) Grippers

Long cables can limit the efficacy of pinch grasps, in which both grippers firmly grasp the cable, because such grasps are unable to efficiently manage slack between grippers. Prior work demonstrates the promise of sliding grippers along cables [43], and to this end we present a passive mechanical design that facilitates pinch-pinch, cage-cage, and cage-pinch grasps. The design, shown in Fig. 3.3, attaches to the YuMi robot's standard parallel jaw grippers. Each jaw includes a perpendicular "foot" segment that facilitates both **caging grasps**, where the jaws open partially and the feet prevent the cable from slipping out, and **pinching grasps**, where the jaws firmly grasp the cable. Caging grasps enable the cable to freely slide inside the gripper, while pinch grasps impart more force and prevent slack from slipping through. These two modes of cable grasping enable a variety of manipulation primitives that are particularly suitable for long cables, without the need for sensing or closed-loop control.

Manipulation Primitives

SGTM 1.0 employs 5 cable manipulation primitives which are facilitated by the PC jaws and use an analytic grasp planner which selects collision-free gripper poses such that the grippers avoid each other as well as neighboring cable segments.

Reidemeister move

To uncover the cable's underlying knot structure $\mathcal{K}(c_t)$, the robot spreads it apart by pinching both of the endpoints, $c_t(0)$ and $c_t(1)$, and pulling them 1.2 meters apart towards opposite ends of the workspace. This was determined empirically to remove occlusions while ensuring no knots are tightened. This is an example of a Reidemeister move in knot theory, and in this case it serves to spread out slack to reveal knots.

Cable Shaking

The Reidemeister move requires both cable endpoints to be visible and graspable. When this condition does not hold, we leverage dynamic shaking actions, a popular manipulation primitive in deformable manipulation [13, 62]. In this paper, the robot performs shaking actions to uncover occluded endpoints and loosen knot structure. If no endpoints are identifiable, the robot computes a random point on the cable's mask in the overhead RGBD image. Using the pointcloud from the RGBD image, the robot identifies and pinches the 3D point corresponding to this point, c_{shake}. After pinching, the robot raises its arm 0.7 meters off the table and executes a shaking motion by rotating the wrist joint 3 times by 2 radians, at a frequency of 1.5Hz, with a radius of 0.1 meters. These parameters were determined empirically to maximize the disruption to the cable's visual state in consideration of the YuMi robot's limits. If exactly one endpoint is visible and graspable, and the previous move was not a shaking action, the robot pinches this endpoint, $c_{shake} = c_t(0)$ or $c_{shake} = c_t(1)$.

We also make use of the shaking action as a recovery move in case of any failures that may occur in other primitives.

Bimanual physical tracing

During bimanual physical tracing, the robot slides along the cable until it visually identifies either an endpoint or a knot.

In this motion, the robot first pinches two nearby points on the cable $c_t(s_{trace,l})$ and $c_t(s_{trace,r})$ with the left and right arms respectively. Without loss of generality, let the left arm be the *pulling arm* and the right arm be the *sliding arm*. The robot alternates between pinching and caging between the two arms to slide the cable through one gripper with the other. The robot first cages the cable with the sliding arm while the pulling arm pinches and pulls the cable through by 0.1 meter segments as shown in Fig 3.5. After each segment, the sliding arm pinches the cable and the pulling arm cages the cable, allowing it to move forward to meet the sliding arm without dropping or pushing the cable. The robot repeats this procedure until it either detects an endpoint or a knot approaching the sliding arm from overhead images. The detection algorithms are described in Section 3.2.

While sliding along the cable, the robot dangles the cable 0.45 meters above the work surface, causing loose loops to fall away while knots remain. SGTM 1.0 uses this behavior to closely inspect the cable to actively locate knots in the presence of excess slack.

This motion begins at an endpoint, so that during execution the portion of cable that has been physically traced contains no knots, an invariant useful for managing slack in subsequent steps and indicating untangling success. If the robot slides uninterrupted from one endpoint to the other without detecting knots, it can verify that the entire cable contains no more knots. This is used as a termination condition by SGTM 1.0.

Knot isolation

Immediately after physical cable tracing terminates at a knot, we execute this primitive, which improves the likelihood of successful untangling in future steps. The sliding arm deposits the knot it is holding onto the workspace, while the other arm sweeps aside the remaining slack not pertaining to the knot.

Dual-cage separation

To untie individual knots, the robot attempts to cage two points inside the knot and then slowly pull its arms apart while wiggling the wrist joint 15 times by 0.2 radians. This wiggling helps reduce friction between segments of cable sliding past each other. Precisely, given a knot $[a, b]$ in cable c_t, the robot grasps two graspable points $c_t(s_{knot,l})$ and $c_t(s_{knot,r})$ with the left and right arms respectively, such that $s_{knot,l}, s_{knot,r} \in (a, b)$.

The double cage grasps allow the cable to freely slip through the fingers as the knot loosens, allowing the robot to untie knots in one action where the arms move as far apart as kinematically feasible, approximately 1 m apart. If the robot encounters sufficient resistance due to an endpoint or knot at either gripper, exceeding the YuMi torque limits, our algorithm will automatically stop and reset.

Perception Systems

In contrast to some prior work which performs full state estimation of the cable [27], we rely on learned perception methods, since the significant lengths of slack introduce occlusions that make full state estimation difficult. In addition, to initiate certain manipulation primitives in Section 3.2, SGTM 1.0 also relies on additional manipulation primitives to expose key areas of the cable, such as knots, to predict task-relevant untangling keypoints.

Endpoint detection

In the Reidemeister move and bimanual physical tracing primitives, SGTM 1.0 relies on detecting endpoints in the image. We collect and manually label a dataset of 700 images in the workspace of the cable either on the manipulation surface or in the gripper jaws during physical tracing. We train a fully-convolutional network [26] based on a ResNet 50 backbone to output a Gaussian heatmap over each endpoint as in [12, 46, 52].

Visual cable tracing

SGTM uses *visual* cable tracing to identify good starting points for *physical* tracing (Figure 3.4). Given an RGB image of a segmented cable and an endpoint pixel location, algorithmic cable tracing uses a modified breadth-first search (BFS) to follow the cable mask in the image and stop at the first self-crossing. It detects crossings by monitoring the frontier of added pixels at each BFS iteration. If the bounding box tightly surrounding the frontier

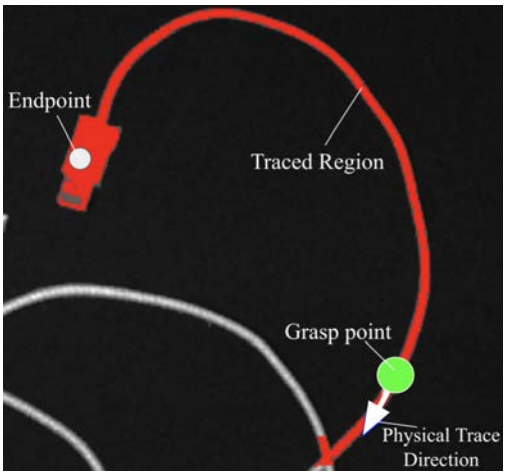


Figure 3.4: **Visual Cable Tracing**: Beginning from the detected endpoint keypoint (white), the algorithm traces the cable using BFS along cable pixels until it detects that it has reached a crossing, after which it backtracks to return the grasp point (green) and slide direction (blue) for subsequent physical cable tracing.

has side length over 12px, the algorithm terminates, as the search is now bleeding across different cable segments. After termination, the algorithm backtracks by 100px and returns a pixel location along the cable which is a safe distance from the crossing to grasp. SGTM 1.0 uses this keypoint to expedite bimanual physical cable tracing by avoiding physically tracing segments from the endpoint prior to any crossings.

Physical tracing stopping condition

During physical cable tracing, SGTM 1.0 pulls the cable through the sliding gripper in segments of 0.1 meters. After every segment, the stopping condition categorizes the next segment of cable near the sliding gripper as a knot, endpoint, or straight cable in the following manner: the robot takes an overhead image and evaluates the endpoint detector (Section 3.2) to check if any endpoints are within a $0.1 \times 0.1 \times 0.1m$ cube around the sliding arm grippers by using the depth values at each predicted endpoint. In order to suppress false positives, we ignore endpoint detections when over $0.4m$ of the cable remains to be traced. If an endpoint is found after this range, the stopping condition returns *ENDPOINT*. This is shown in Fig 3.5.s

If an endpoint is not found, the robot analyzes the pointcloud surrounding the gripper of the sliding arm to categorize whether the next segment of cable is a knot or a straight cable. The robot performs a $3 \times 13 \times 6$ cm volume crop in front of and beneath the sliding gripper to capture the next segment of cable. This volume is separated into 1 cm thick horizontally-sliced cross sections, which are each analyzed for the number of connected components in a re-projected depth image (see Fig. 3.5). If any cross section contains multiple connected components, this could either indicate a loop that has not been undone by gravity during physical tracing or a knot. If any cross section contains multiple connected components

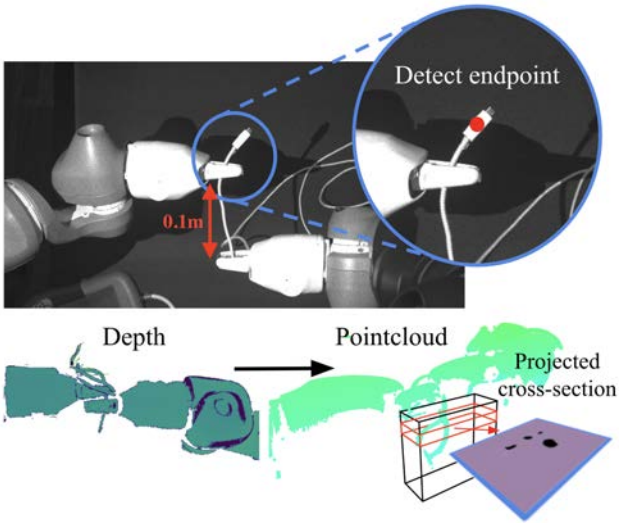


Figure 3.5: **Physical Tracing Stop Condition**: *Top*: Physical cable tracing uses an endpoint detector to locate endpoints and terminate when any reach close to the gripper. *Bottom*: Physical cable tracing detects knots by horizontally slicing the pointcloud in front of the gripper (right), then analyzing the cross-sections for the number of connected components. In this case, the figure-eight knot contributes 4 connected components (black) to a cross section and reaches the threshold for number of points, so a knot is detected. If neither an endpoint nor a knot is detected, the robot continues sliding by beginning at the position shown in the bottom image, caging the left gripper and pinching the right gripper, and pulling the right gripper back as in the top image, tracing along the cable until a knot or endpoint is reached.

and the number of points in the volume crop is at least 1000 points, the stopping condition returns *KNOT*. If no cross sections contain multiple connected components, a knot could still exist if it is very tightly zipped together or self-occluded. Thus, if the volume crop contains at least 2000 points, the stopping condition also returns *KNOT*. If none of these conditions are satisfied, the stopping condition returns *STRAIGHT*.

Knot detection

We train an object detection network, based on the Mask R-CNN architecture and implemented using the Detectron2 codebase [59], to detect and classify figure-eight and overhand knots in cable images taken by the overhead PhoXi camera.

The network is initialized with weights from a ResNet-50 FPN backbone pretrained on the COCO dataset [23] and trained on a dataset of 312 images with manually annotated knots in images containing both knots and loops.

Pull point detection

If the knot detection step locates a knot bounding box, it outputs a crop of the knot, which is passed through a two-stage cascading network. Both independently-trained stages use a Resnet34 backbone followed by a sigmoid activation to predict two pull points for performing

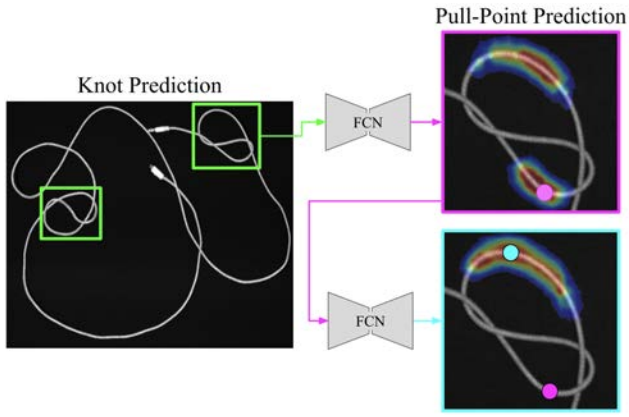


Figure 3.6: **Learned Pull Point Prediction**: Left: predictions from the knot detection method on an image containing two figure-eight knots. Right: visualization of the cascading pull point detector. The top network outputs a heatmap over potential grasp points, and we select the point at which the heatmap has the highest value. The bottom network is conditioned on the first selected pull point (magenta) to predict a heatmap for the second pull point (blue).

a dual-cage separation action. The first network outputs a heatmap for both potential pull points. Of these, we select the location of the highest heatmap value, projected onto the nearest cable segment. The second network, trained separately, takes in the same cropped image with a fourth channel encoding a heatmap centered around the already-chosen first point, and it outputs a heatmap from which the second point is chosen the same way, using an argmax. This process is visualized in Figure 3.6.

We use the two-network cascading design because with a single network, we find that pick point multimodality often causes heatmap outputs to bleed together significantly, making it difficult to choose two distinct points. Conditional action prediction has also been previously studied in deformable manipulation by Wu et al. [58] for cable and fabric manipulation.

Sliding and Grasping for Tangle Manipulation (SGTM) 1.0

We now describe the Sliding and Grasping for Tangle Manipulation (SGTM) 1.0 algorithm, which combines the manipulation primitives and perception subsystems from Sections 3.2 and 3.2 to untangle long cables while managing cable slack. An overview of the algorithm is displayed in Fig. 5.5. During each iteration, SGTM 1.0 alternates between two phases: physical manipulation to increase knot visibility (*Active Knot Perception*) and predicting keypoints for a) the dual-cage separation maneuver to loosen and untie knots once they are detected or b) physically checking if the cable is untangled when no knots are detected (*Knot Untangling/Physical Tracing*).

1. *Active Knot Perception:* During execution, at each iteration, SGTM 1.0 begins by detecting visible endpoints. If both are detected, it executes a Reidemeister move, and if none are detected it executes a shake action from the centroid of the cable cluster.

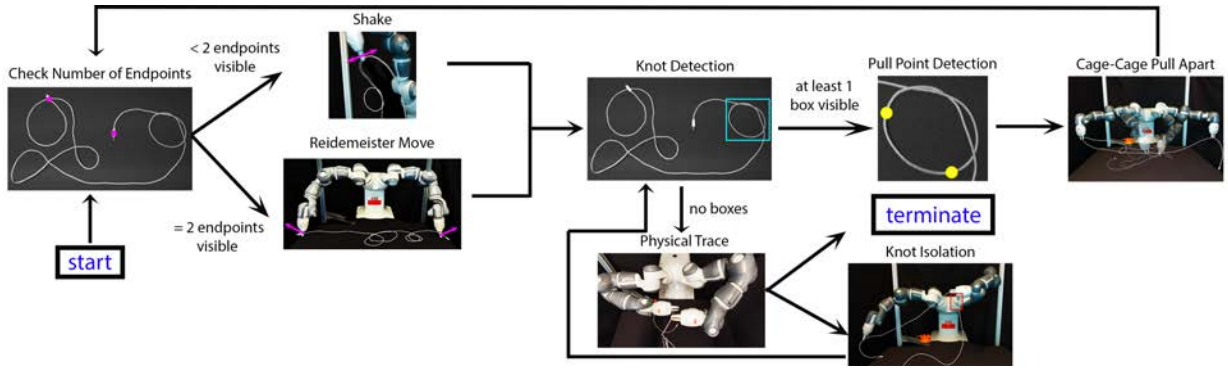


Figure 3.7: **Sliding and Grasping for Tangle Manipulation (SGTM) 1.0**: SGTM 1.0 begins by recognizing endpoints. If both are visible, it proceeds with a Reidemeister move. If not, it shakes. Afterwards, knot detection is performed after which it untangles if a knot is visible. If not, it performs physical tracing to ensure no knots remain. If a knot is found while tracing, the knot is isolated and placed in the workspace, and the algorithm returns to the knot detection step.

If a shake action was previously performed and no endpoints are visible still, the shake action is executed again from a random location on the cable. All three types of moves serve to spread out the cable and increase the chance of perceiving knots.

2. *Knot Untangling/Physical Tracing:* Next, if knot bounding boxes are detected, the robot executes a dual-cage separation action to attempt to untie the knot closest to an endpoint (found using the technique in 3.2) based on grasp keypoints the perception system outputs. If SGTM 1.0 does not detect knots, the robot initiates the physical tracing stage to confirm that the cable is untangled, either ending in termination or by detecting a knot and isolating it on the table for further untangling.

3.3 Experiments

We evaluate SGTM 1.0 on a set of physical cable untangling experiments using the bilateral ABB YuMi robot. The experiments evaluate whether SGTM 1.0 can untangle different classes of initial configurations and terminate when it has completely untangled the cable.

Difficulty Tiers and Performance Metrics

We consider several methods for initializing the cable state, with examples shown in Fig. 3.8. These are categorized into the following tiers of difficulty:

- **Tier 0**: Cable has no knots, but the slack is piled randomly within the workspace by holding both endpoints 1.5 meters above the workspace and dropping them. In this tier, the robot must successfully verify that the cable contains no knots before terminating. We report the number of trials that successfully detect that the cable is

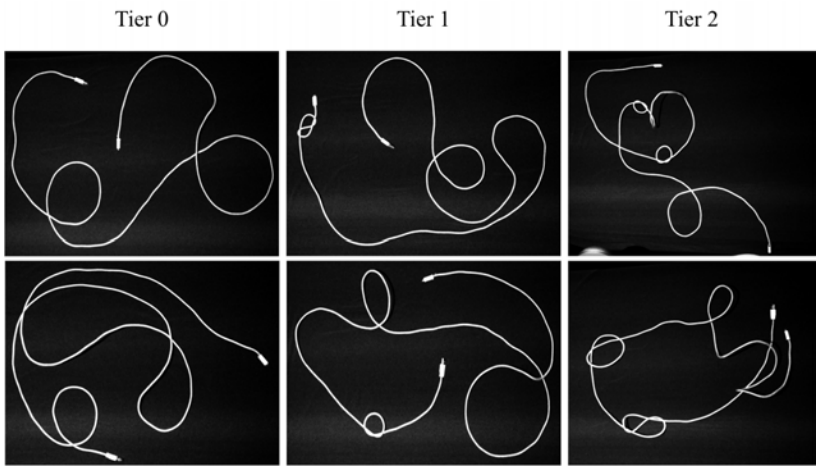


Figure 3.8: **Cable Configurations**: Shown here are example configurations of the cable across the 3 tiers we run experiments on.

untangled (Untangled Detection Rate), average number of manipulation actions (Avg. Number of Actions), and average time to detect that the cable is untangled (Avg. Untangled Detection Time).

- **Tier 1**: Cable has a single overhand or figure-eight knot that is loose (12-14 cm diameter) or tight (6-8 cm diameter), located close to an endpoint (side), 0.75 meters from the endpoint (mid-center), or 1.5 meters from an endpoint (center). It is arranged with the knot isolated from the rest of the cable slack, and the slack is randomized by lifting both endpoints as high above the workspace as possible without lifting the knot itself from the workspace and dropped. We report the number of trials that successfully untangled the knot (Untangling Success Rate), the number of trials that both untangled the knot and detected that the knot was untangled (Untangled Detection Rate), the average time taken to untangle the knot (Avg. Time to Untangle), and the average time taken to successfully identify that the cable was untangled (Avg. Untangled Detection Time).

- **Tier 2**: Cable has two overhand or figure-eight knots (including mixed types) that are loose or tight, in series, located close to each other (< 0.75 m apart) or far from each other (> 1.5 m apart). Similar to tier 1, both knots are isolated from the rest of the cable slack and the endpoints are lifted as high above the workspace as possible without lifting either knot from the workspace and dropped. We record the number of trials that successfully untangle a single knot (Untangling 1 Success Rate), number of trials that successfully untangle both knots (Untangling 2 Success Rate), the number of trials that untangle both knots and detect that it is untangled (Untangled Detection Rate), average number of manipulation actions (Avg. Number of Actions), average time to untangle the first knot (Avg. Time to Untangle 1), average time to fully untangle the

Untangled Detection Rate	8/12
Avg. Number of Actions	4.33
Avg. Untangled Detection Time (s)	266
Failures	A (1), B (1), C (2), D (0), E (0)

Table 3.1: **Tier 0 results:** In 12 tier 0 experiments, SGTM 1.0 must detect that a cable with no knots is untangled. We observe that SGTM 1.0 is able to successfully terminate in 8/12 cases using an average of 4.33 manipulation primitive actions. See Section 4.3 for analysis of the failure modes A-E.

	Loose	**Dense**
Untangling Success Rate	4/6	4/6
Untangled Detection Rate	1/4	2/4
Avg. Number of Actions	7.17	7.5
Avg. Time to Untangle (s)	154	139
Avg. Untangled Detection Time (s)	270	527
Failures	A (1), B (0), C (2), D (0), E (0), F (2)	A (1), B (0), C (1), D (1), E (0), F (1)

Table 3.2: **Tier 1 results:** In 12 experiments, the cable starts off with a single knot in its initial configuration. SGTM 1.0 is able to successfully untangle the cable 4/6 times in both the loose and dense cases of this problem. However, when the robot untangles the cable, it often fails to detect that it has successfully untangled the cable. This occurs because the robot forms loops that are not knots during execution, which are mistaken for knots during physical tracing (Failure Mode C).

cable (Avg. Time to Untangle 2), and the average time to untangle both knots and detect that the cable is untangled (Avg. Untangled Detection Time).

For each tier 1 or 2 experiment, we record the average time until the cable has been successfully untangled. Because the robot may successfully untangle the cable but not successfully recognize that the cable is untangled (Section 4.3), we separately record the average time to successfully detect the cable is untangled (when this is the case). We cap each experiment to 15 minutes in tiers 0 and 1 and to 20 minutes in tier 2. This is because tier 2 configurations are much more difficult and require more time to untangle. We analyze the observed failure modes of the algorithm in Section 4.3.

Experimental Setup

The workspace contains a bimanual ABB YuMi robot with two PC grippers. The manipulation surface is black and foam-padded to avoid end effector damage during any workspace collisions. The cable is a light-gray, braided 2.7 m micro-USB to USB cable that can be segmented from the manipulation surface via color thresholding. The workspace has an overhead PhotoNeo Phoxi Camera that captures depth and grayscale images of resolution 732 x 1142px.

	Loose	Dense
Untangling 1 Success Rate	5/6	5/6
Untangling 2 Success Rate	3/6	3/6
Untangled Detection Rate	0/3	1/3
Avg. Number of Actions	9.83	10.17
Avg. Time to Untangle 1 (s)	71	112
Avg. Time to Untangle 2 (s)	475	670
Avg. Untangled Detection Time (s)	N/A	1079
Failures	A (0), B (1), C (1), D (0), E (1), F (3)	A (1), B (1), C (1), D (1), E (1), F (0)

Table 3.3: **Tier 2 results:** In this tier of 12 experiments for dense and loose knots, the cable starts off with two knots in its initial configuration. The robot is able to successfully untangle at least one of the knots 5/6 times in both the loose and dense versions of this tier. In 3/6 of both the loose and dense cases, SGTM 1.0 successfully untangles both knots. However, similar to Tier 1, many loops and sometimes new knots are formed during untangling, which lead to timing out during a trial due to failure of untangled detection.

Results

In tier 0, SGTM 1.0 successfully detects that 8/12 cases are untangled, taking an average of 4.33 manipulation primitive actions and 266 seconds (Table 3.1). In the cases where SGTM 1.0 does not terminate successfully, the robot either moves the cable off the manipulation surface, creates a new knot during execution, or falsely detects loops as knots during physical tracing.

In tier 1, SGTM 1.0 successfully untangles the cable in 4/6 trials in both the loose and the dense categories (Table 3.2). The untangled detection rate is 1/4 for loose configurations and 2/4 for dense configurations, and the most common errors are detection of loops as knots during physical tracing and system errors with the YuMi robot.

In tier 2, SGTM 1.0 is able to untangle a single knot in 5/6 trials in both the loose and dense cases (Table 3.3). In both cases, the robot untangles both knots 3/6 times. The robot often has trouble detecting that it has successfully untangled the cable in this case, due to an accumulation of twists during untangling that lead to loops that look like knots during physical tracing.

Failure Modes

In experiments for SGTM 1.0, we observe the following failure modes:

(A) Unseen or complex knots form while untangling and are not detected by Mask R-CNN for knot bounding box detection.

(B) The cable falls out of the reachable workspace.

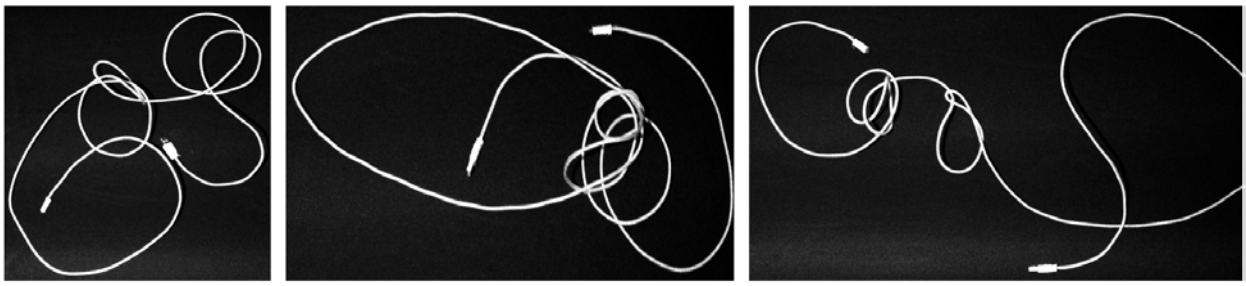


Figure 3.9: **Failure Mode A Examples**: These are examples of complex knots or unseen knots that form while untangling, but fall out of the distribution of training samples for the knot detector, causing a failed knot detection.

(C) Repeated false positive knot detections during physical tracing result in hitting the time limit.

(D) Missed knots during physical cable tracing cause knots to tighten.

(E) Knot untangled, but not enough time to check termination.

(F) YuMi robot system or reset errors.

(A) During execution, SGTM 1.0 sometimes introduces new knots by pulling endpoints through existing loops on top of them as shown in Figure 3.9. Though unlikely, these occurrences are often difficult to recover from since knots formed in this way are visually distinct from the training set of knot bounding box detection model. This failure happens particularly often after no endpoints are detected and the robot must shake from a random cable location, after which endpoints are buried underneath slack.

(B) Because we shake near the edge of the table, sometimes the motion causes part of the cable to drop off the side, which can cause the rest of the cable to follow after it is released.

(C) One particularly confusing case for knot detection during physical tracing is tightly twisted loops which do not fall away due to gravity. Sometimes this happens multiple times in a rollout, wasting time and eventually causing the time limit to be reached.

(D) If SGTM 1.0 fails to detect a knot during physical cable tracing, the sliding actions can tighten the knot until it is too dense to fit grippers inside, resulting in failure.

(E) Sometimes, SGTM 1.0 is able to recover from situations like failure mode (A), but this can happen very late in the experiment, resulting in not enough time to check termination.

(F) YuMi errors result from 1) the robot getting stuck at a singularity from which it is unable to reset or 2) the robot attempting to reach a point just outside of its reachable workspace and producing errors from which it cannot recover.

We observe that the physical tracing termination condition is low-recall but high-precision; that is, if it claims that the cable is untangled, it most likely is (only in one case out of 36 experiments did it output a false positive). However, while false positives terminate by presenting a tangled cable as untangled, a false negative results in the robot continuing to try

to untangle an untangled cable. The latter is less severe, because the robot often eventually discovers that the cable is untangled; thus, we prefer SGTM 1.0 to err on the side of false negatives.

3.4 Discussion

For untangling a single long (up to 3m) cable, this work presents a formal problem definition, a novel jaw design, novel perception-based primitives, and Sliding and Grasping for Tangle Manipulation (SGTM) 1.0, an algorithm for untangling long cables. SGTM 1.0 introduces 3 novel manipulation primitives for cable manipulation: shaking, physical tracing, and dual-cage separation. These primitives aid in easing perception and managing slack in long cables. SGTM 1.0 also introduces perception systems that guide the usage of these primitives. Experiments show that SGTM 1.0 can untangle long cables with a 58.3% success rate overall across tiers containing one to two knots.

3.5 Individual Contribution

This chapter is an adaptation of the award-winning Best Systems Paper titled "Autonomously Untangling Long Cables," which is published in RSS 2022. The paper is a collaborative effort with the following co-authors: Kaushik Shivakumar, Justin Kerr, Brijen Thananjeyan, Ellen Novoseller, Jeffrey Ichnowski, Alejandro Escontrela, Michael Laskey, Prof. Ken Goldberg, and Prof. Joseph E. Gonzalez.

My specific contributions included assistance on developing manipulation primitives and developing a learning-based knot detection, learning-based pull point detection, and the physical trace stop condition. I was also involved in the development and full stack integration of the SGTM 1.0 algorithm on the ABB YuMi robot.

I would like to acknowledge my co-first authors, Kaushik Shivakumar and Justin Kerr. Kaushik's contributions encompass the development of the key primitive dual cage separation primitive, analytic cable tracing, and collaboration on knot detection, pull point detection, the full SGTM 1.0 algorithm, and the physical trace stopping condition. Additionally, Kaushik created Figures 3.1 and 3.4 for this chapter. Justin played a pivotal role in designing the PC grippers and developing the majority of the manipulation primitives, including the reidemeister move, cable shaking, and bimanual physical tracing. Prior to the project, he was instrumental in setting up the YuMi robot and writing Python bindings with Michael Danielczuk, which greatly facilitated interfacing with the YuMi. Justin's contributions to this chapter also include Figures 3.2 and 3.6.

I am grateful to Brijen Thananjeyan, Ellen Novoseller, Alejandro Escontrela, and Jeffrey Ichnowski for their guidance throughout the project's development and the framing of the paper's contributions. Brijen also contributed the learning-based endpoint detection. Their

input and valuable feedback were invaluable during the multiple drafts of the submitted version of this paper. Alejandro also contributed Figure 3.5.

Finally, I would like to express appreciation to Michael Laskey at TRI and Joseph E. Gonzalez for their insightful feedback and suggestions on the paper drafts. Last but not least, I extend my thanks to Professor Ken Goldberg for not only providing feedback on the paper but also for his invaluable insights and creative ideas throughout the entire duration of this project.

Chapter 4

SGTM 2.0

4.1 Problem Statement

As in [53], we consider untangling long ($\sim 3\,\text{m}$) cables from RGB-D image observations. However, now we focus on improving speed and accuracy of the system using interactive perception (introduced in Section 2.2). We use a bimanual robot to execute manipulation primitives until the cable reaches a fully untangled state with no knots.

Workspace Definition and Assumptions

The bilateral robot operates in an (x, y, z) Cartesian coordinate frame with two 6-DOF robot arms. The robot is equipped with cage-pinch jaws introduced in [53] to allow for both sliding along and tightly pinching the cable. The workspace lies in the xy-plane and the only inputs to the algorithm consist of RGB-D images. The workspace contains a single incompressible electrical cable of length l_c and cross-sectional radius r_c. Cable state $s \in \mathcal{S}$ can be described as a continuous path $c_n(u) : [0, 1] \rightarrow (x, y, z)$ in the workspace, where u indexes the position along the length of the cable. $c_n(0)$ and $c_n(1)$ always refer to the position of the endpoints of the cable. We initialize the cable's state before $n = 0$ with the procedures specified in Section 4.3. One challenge in this problem is that parts of the cable may rest outside the reachable and observable workspace at any point in a rollout (defined as a single experiment aiming to remove all knots in the cable). Moreover, self-occlusions in the cable are possible due to only one overhead camera view. This partial observability motivates the need for actions that reveal more information about the cable state s.

We make the following assumptions: (1) the cable can be segmented from the background via color thresholding; (2) transformations between the camera, workspace, and robot frames are known; and (3) the cable start state contains overhand or figure 8 knots of dense (6-8 cm diameter) or loose (12-14 cm diameter) configurations in series.

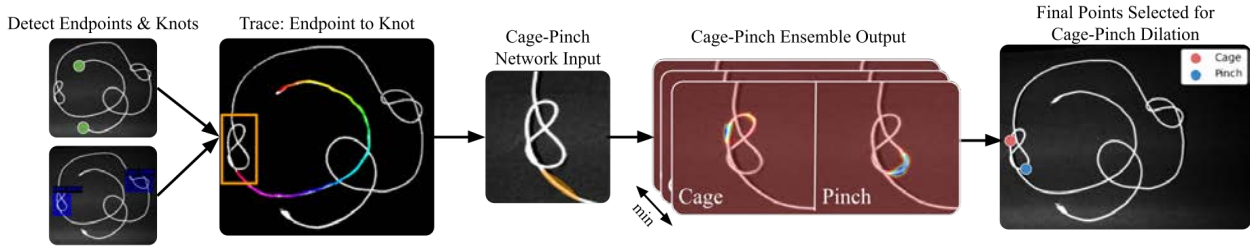


Figure 4.1: **Perception system:** This is the pipeline used to determine which points to cage and pinch for a cage-pinch dilation move, the crucial action for untangling a knot. First, we detect the endpoints and knots. Next, we trace from the endpoint to the first bounding box. If the trace is certain, we run the cage-pinch network ensemble on the cropped knot in the bounding box with the trace tail encoded into one of the channels. We take the pixelwise minimum across the cage-pinch network ensemble outputs, leading to 1 heatmap encoding "worst-case" untangling success probabilities each for the cage and pinch point. We take the argmax of each of the two heatmaps to determine the final points to pinch and cage during the cage-pinch dilation action.

Task Objective and Metrics

The goal of the robot is to untangle the cable and terminate at time $t < T_{\max}$, specified in Section 4.3. After each step of a rollout r, a new observation o of the cable state s is taken. Each primitive constitutes at least one step.

The goal of the robot over the course of each rollout is to untangle the cable and output a termination signal (`DONE`). We use $H_{\texttt{DONE}}$ to represent a step function, with the step occurring when the robot outputs `DONE`. We define an untangled cable as one that has no knots when its endpoints are grasped top-down and extended the maximum feasible distance, with knots defined identically to [53]. We use k_t^r to denote the number of knots in the cable at time t in rollout r and assume the cable is initialized with k_0^r knots.

We use the following metrics to measure performance, where $0 < K \leq k_0^r$ refers to the number of knots untangled and R is the total set of rollouts:

1. *Untangling K Success Rate,* the percentage of rollouts that untangle K knots: $\frac{1}{|R|}\sum_{r\in R}\mathbf{1}_{\{\exists t<T_{\max}\ :\ k_t^r\leq k_0^r-K\}}$
2. *Untangling Verification Rate*, the percentage of rollouts that untangle all knots and terminate successfully: $\frac{1}{|R|}\sum_{r\in R}\mathbf{1}_{\{\exists t<T_{\max}\ :\ k_t^r=0\ \wedge\ \texttt{DONE}_t\}}$
3. *Average Untangling K Time*, the average time to untangle K knots across all applicable rollouts R_a where this occurs before $T_{\max}$: $\frac{1}{|R_a|}\sum_{r\in R_a}(\min t\ :\ k_t^r \leq k_0^r - K)$
4. *Average Untangling Verification Time,* the average time to reach $k_t^r = 0$ and declare termination across all applicable rollouts R_a like above: $\frac{1}{|R_a|}\sum_{r\in R_a}(\min t\ :\ k_t^r = 0 \wedge \texttt{DONE}_t)$

4.2 Methods

Approach Overview

Unlike SGTM 1.0, SGTM 2.0 uses interactive perception primitives designed to better manage slack during untangling and to reveal additional information about the cable state $s \in \mathcal{S}$. As s is difficult to estimate from the provided observation $o \in \mathcal{O}$, SGTM 2.0 uses a policy $\pi : \mathcal{O} \to \mathcal{A}$ built as described in Section 4.2 from the components in Sections 4.2 and 4.2 to directly predict actions to execute. Unlike prior work, SGTM 2.0 includes perception components that lend themselves to probabilistic interpretation and manipulation primitives that are sensitive to perception uncertainty. We note that while the distribution of states the robot encounters may have high variance, SGTM 2.0 is sensitive only to variance that may affect the next untangling action. Practically, this means that for example, even if much of the cable is bunched up and occluded, as long as the path from an endpoint to the first knot is clearly visible, the algorithm can still take an untangling action with high confidence.

Uncertainty-Aware Perception Systems

Endpoint Detection

We train a Faster R-CNN model with a Resnet-50 FPN (Feature Pyramid Network) backbone [37] on 305 hand-labeled examples to detect cable endpoints. We discard all bounding boxes with lower than 99% confidence, achieving an average precision and recall for endpoint detection are 86.7% and 100% respectively.

Knot Detection

To identify all knots in the observable workspace, we use the same architecture as the endpoint model trained on 688 hand-labeled images. The real-world dataset is augmented with flip, contrast, brightness, rotation, saturation, and scale augmentations. We use a 99% detection threshold, which achieves an average precision of 91.3% and an average recall of 95.5%. We analytically filter out misclassified knots by checking if a simple loop fills the bounding box. Because the model's output is dependent on the orientation of the cable, in certain cases, we use multiple observations of the underlying cable state s as described in Section 4.2. This allows SGTM 2.0 to be sensitive to what we define as *observational uncertainty*.

Analytic Cable Tracing

The objective of cable tracing is to outline all likely paths of the cable from an overhead image. Given an RGB image and a start pixel (the center of one endpoint), the tracer we introduce in this work outputs a set of possible splines. It maintains a set of valid splines and iteratively expands it by exploring candidate successor points, preferring those that do not

deviate sharply from the current spline's trajectory. An example of candidate trace paths is shown in Figure 4.2.

Sliding and Grasping for Tangle Manipulation (SGTM) 2.0 uses this in 2 scenarios: (1) finding a knot to untangle and (2) achieving robust grasps near cable endpoints. When used for finding a path from endpoint to knot, the tracer terminates once all potential traces intersect the knot within a bounding box. For grasping endpoints, the trace terminates after traveling a fixed distance along the cable. At the termination of tracing, we fit a bounding box B_t to the ending points of all the final traces; if either dimension of B_t is greater than 24 pixels for knot tracing and 12 pixels for endpoint tracing (which requires more precision), the traces diverge and thus `TRACE_UNCERTAIN` is returned; otherwise, `TRACE_CERTAIN` is returned. Note that while traces with very different topologies may be returned, as long as they end near the same point (Figure 4.2, left and center), the untangling-relevant uncertainty remains low. The reason for tracking uncertainty in this phase is that different paths from the endpoint to the knot may lead to different untangling actions, especially with respect to which point to cage and which to pinch.

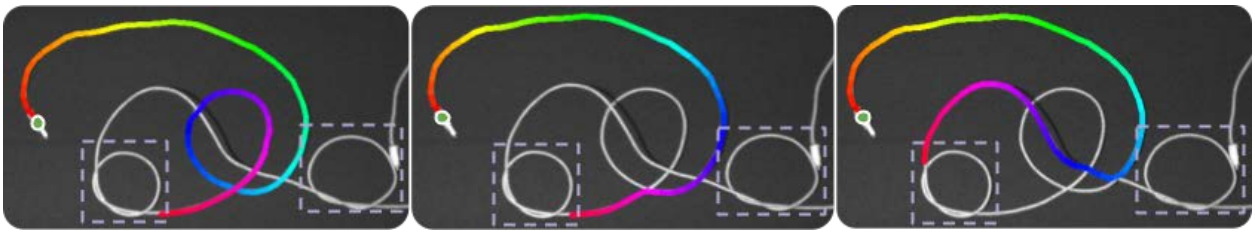


Figure 4.2: **Cable Tracing:** multiple candidate trace paths returned by the tracer to find the closest knot from the top-left endpoint. The tracer finds the correct knot in all cases, but is unclear as to which side of the knot is attached to the free endpoint, and therefore returns `TRACE_UNCERTAIN`.

Cage-Pinch Dilation Point Selection

We use an FCN [25] with a ResNet-34 backbone trained on 568 knot crops to output two heatmaps: one for the cage point and one for the pinch point. We augment this data with random rotations, flips, shear, and synthetically added distractor cables to improve robustness. Approximately 250 of these images are gathered during rollouts to mitigate distribution shift in a style similar to DAgger [38]. The input is a 3-channel image, where one of the channels contains a Gaussian heatmap around the segment of cable entering the image determined by the cable tracing algorithm. This additional input conditions the network and breaks the symmetry between the cage and pinch points. We train the network to predict the cage point as the first graspable point beyond the undercrossing forming the knot, and the pinch point as the place to secure the cable to create an opening for the free end to slide through. Example cage and pinch points and the perception pipeline to obtain the points are shown in Figure 4.1.

The goal of this network is to model the probability $P_p(U_s)$ per grasp pixel p, where U_s corresponds to untangling success on the cropped knot. We train an ensemble of 3 models on the same data but with different initializations. For the cage point, to sample from each of the ensemble heatmaps $h^{\text{cage}} \in H^{\text{cage}}$, we create a new heatmap as such: $h'^{\text{cage}}_{i,j} = \min_{h^{\text{cage}} \in H^{\text{cage}}} h^{\text{cage}}_{i,j}$. The same procedure is used for the pinch point. We return the cage and pinch point as the $\text{argmax}_{i,j}\, h'^{\text{cage}}_{i,j}$ and $\text{argmax}_{i,j}\, h'^{\text{pinch}}_{i,j}$, respectively. If $\max_{i,j} h'^{\text{cage}}_{i,j} \max_{r,s} h'^{\text{pinch}}_{r,s} < \kappa$ where $\kappa = 0.35$ (calibrated empirically), we output `NETWORK_UNCERTAIN`. Otherwise, we output `NETWORK_CERTAIN`.

The above methods help reveal whether the worst-case probability (across our predictive distribution modeled by an ensemble) of untangling success is high enough to proceed with untangling the cable at the specified points. If not, the network is too uncertain in its predicted points to proceed confidently as the next action may instead tighten the knot or lead the cable into an irrecoverable state.

Novel Manipulation Primitives for Interactive Perception

Cage-Pinch Dilation

To untangle an individual knot, the robot leverages the flexibility of the cage-pinch grippers introduced in [53] to cage one point and pinch another point inside the knot and pull apart the arms to a distance determined by the length of the trace from the endpoint to the knot, while moving its wrist joint in a high-frequency sinusoidal motion. A major benefit of cage-pinch actions compared to cage-cage actions from [53] is the ability to better manage slack, preventing accidentally tightening another knot. Following this action, the robot lays the cable down as far forward as kinematically feasible to isolate the newly untangled portion from the remaining cable.

Partial Cage-Pinch Dilation

This primitive is similar to the Cage-Pinch Dilation, but the distance the arms move apart is fixed to 5 cm beyond their starting separation. This is meant to perturb the state and to later retry perception rather than to completely untangle a knot.

Reidemeister Move

In this primitive, the robot uses tracing to find robust grasp points slightly down the cable from the endpoints. Next, the robot moves both arms outward horizontally and up, lifting the cable off the workspace, allowing for loops to fall away. Compared to prior work, we add (1) the vertical component of the action, forming a large "U" with the cable, and (2) the wide lay-out action, which places the cable on the workspace in a "U" shape.

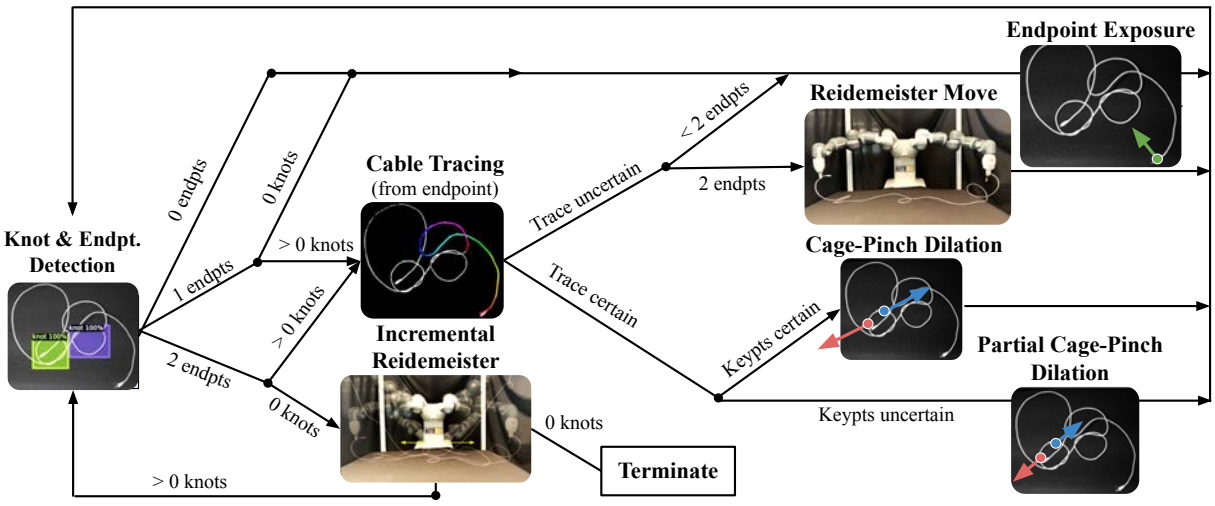


Figure 4.3: **SGTM 2.0 Algorithm:** SGTM 2.0 first detects the number of knots and endpoints in the scene. If the endpoints are not visible, there is no way to verify any knot's relative position to the endpoint. This is necessary because SGTM 2.0 only untangles knots adjacent to an endpoint to avoid knots colliding into each other and creating irrecoverable configurations. If fewer than two endpoints and no knots are visible, the algorithm is also unable to perform a termination check as that requires performing an incremental Reidemeister, which grasps the cable at the endpoints. In both these cases, SGTM 2.0 performs an endpoint exposure. If two endpoints are visible and no knots are visible, SGTM 2.0 proceeds to the incremental Reidemeister move. If one or two endpoints are visible and there are knots in the scene, it attempts to untangle, beginning by tracing from the visible endpoint(s). Here, if it is not able to confidently trace from either endpoint to a knot, SGTM 2.0 performs a Reidemeister move or endpoint exposure (based on the number of endpoints visible) to increase likelihood of unambiguous traces in future steps. Otherwise, it assesses the cage-pinch network uncertainty on the predicted points. If it is confident, it proceeds with a full cage-pinch dilation. Else, it performs a partial cage-pinch dilation to disambiguate the state.

Incremental Reidemeister Move

This primitive performs the exact same motions as a Reidemeister move, but uses a multi-stage, perception-based approach where the cable is observed at certain waypoints along the action. We use our knot detection network at these intermediate points to determine whether any knots remain. This can be interpreted as ensembling via perturbation of the observation of the same underlying cable topology. Being sensitive to observational uncertainty allows us to eliminate the time-consuming physical tracing action used in [53] for termination.

Exposure Action

When one or more endpoints are missing for an action, we uniformly at random sample a segment of the cable leaving the reachable workspace and pull it towards the center of the workspace to increase visibility. We also do this for unreachable knots we wish to act on.

Sliding and Grasping for Tangle Manipulation (SGTM) 2.0 Algorithm

Sliding and Grasping for Tangle Manipulation (SGTM) 2.0 ties together the aforementioned perception components and manipulation primitives to untangle cables. SGTM 2.0 alternates between the perception and manipulation components, using uncertainty from the former to determine whether to untangle or disambiguate the cable state. The algorithm is covered in detail in Figure 4.3.

4.3 Experiments

Experimental Setup

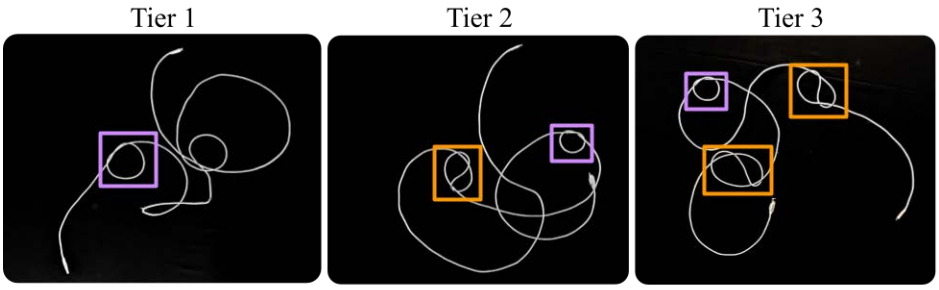


Figure 4.4: **Example starting configurations for all 3 tiers:** Overhand knots are outlined in purple and figure-8 knots are outlined in orange.

For our experiments, we use the bimanual ABB YuMi robot with an overhead Phoxi camera, operating on a black foam-padded workspace of width $1.0\,m$ and depth $0.75\,m$. Due to hardware constraints, we slightly extend the workspace with cardboard (by 0.1 meters on either side) not previously present in SGTM 1.0, but this does not make a significant difference as the cable mostly remains in the original foam-padded workspace. We use a $2.7\,m$-long white, braided electrical cable with USB adapters on both ends.

We evaluate SGTM 2.0 on 3 tiers of difficulty:

1. **Tier 1:** A cable with 1 overhand or figure-8 knot.
2. **Tier 2:** A cable with 2 overhand and/or figure-8 knots.
3. **Tier 3:** A cable with 3 overhand and/or figure-8 knots.

The knots in all tiers are evaluated equally in both loose and dense configurations and evenly across positions along the cable (closer to an endpoint vs. closer to the middle). Example start configurations are shown in Figure 4.4. The cable is initialized by laying the knot(s) flat on the workspace, raising the endpoints as high as possible without lifting the knot(s), and then dropping the endpoints. We enforce a time limit of 15 minutes for all tiers. Note

that the cable initialization procedures in Tiers 1 and 2 of this work are *exactly* the same as Tiers 1 and 2 in SGTM 1.0, the prior state-of-the-art [53].

For each tier, we report the average time to fully untangle the cable (for the rollouts that succeed in doing so) as well as the average time to correctly report that the cable is untangled (for the rollouts that succeed in doing so). Additionally, we report the success rates for untangling alone and untangling with termination detection. For Tiers 2 and 3, we present ablation results where SGTM 2.0(-IP) represents SGTM 2.0 with the interactive perception components removed. For Tiers 1 and 2, we also report the speedup of SGTM 2.0 and SGTM 2.0(-IP) from SGTM 1.0. Our baseline for experiments is SGTM 1.0 because to our knowledge, there are no prior algorithms for untangling long cables.

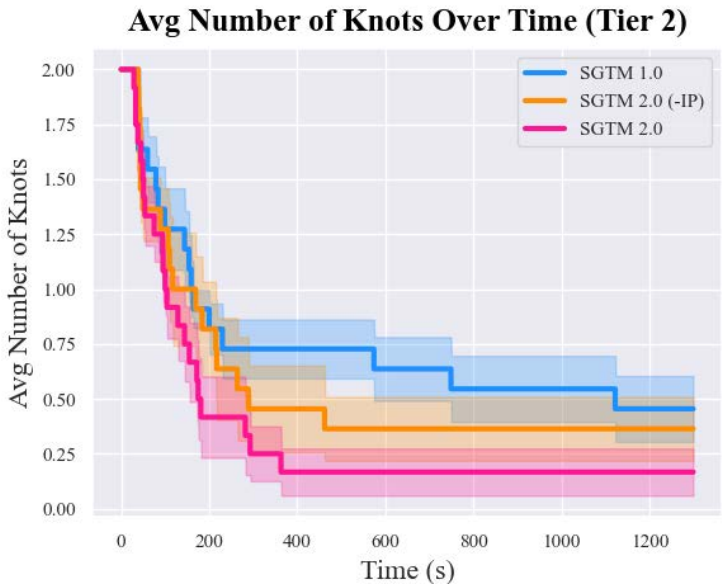


Figure 4.5: **Average Knots over Time (Tier 2)**: SGTM 2.0 most quickly reduces the number of knots on average. SGTM 2.0(-IP) is less successful in untangling over the given time period than SGTM 2.0. Further, performance of even SGTM 2.0(-IP) exceeds that of the prior state-of-the-art, SGTM 1.0, at 15 minutes, showing the effectiveness of improved untangling primitives like cage-pinch dilations introduced in SGTM 2.0.

Results

Results show that across Tiers 1 and 2, SGTM 2.0 outperforms SGTM 1.0 not only on untangling and verification success rate, but also achieves statistically significant speedups in the untangling time in Tier 1, untangling time for both knots in Tier 2, and verification time in Tier 2. Tier 3 was unachievable in SGTM 1.0 due to algorithmic constraints, but is now possible with SGTM 2.0, which achieves 9/12 successes in untangling 2 out of the 3 knots in Tier 3. In 4.3, we discuss difficulties that result in failures, leading to 3/12 untangling success on all 3 knots in Tier 3.

Table 4.1: Results from Physical Experiments (84 total trials). Key takeaways include 1) higher success rates in SGTM 2.0, which ablations suggest is a result of interactive perception. Reduced completion times compared to SGTM 1.0 because of novel interactive manipulation primitives.

	Tier 1		**Tier 2**			**Tier 3**	
	SGTM 1.0	SGTM 2.0	SGTM 1.0	SGTM 2.0(-IP)	SGTM 2.0	SGTM 2.0(-IP)	SGTM 2.0
Knot 1 Succ. Rate	8/12	**10/12**	10/12	10/12	**12/12**	12/12	12/12
Knot 2 Succ. Rate	-	-	6/12	7/12	**10/12**	9/12	9/12
Knot 3 Succ. Rate	-	-	-	-	-	3/12	3/12
Verification Rate	3/12	**7/12**	1/12	5/12	**7/12**	0/12	**2/12**
Avg. # of Actions	**5.0±1.5**	5.6±1.7	12.0	**6.0±1.2**	7.7±1.6	N/A	**12.0±0.0**
Knot 1 Time (s)	189.8±69.4	**53.1±7.5**	93.3±16.2	128.6±41.9	**75.6±21.0**	88.7±25.5	**69.8±15.1**
Knot 2 Time (s)	-	-	586.4±165.3	**160.9±26.4**	180.9±26.2	**177.3±28.3**	233.1±57.6
Knot 3 Time (s)	-	-	-	-	-	**417.7±104.1**	476.7±160.1
Verif. Time (s)	330.8±127.8	**295.9±61.4**	1079.0	**359.6±74.6**	406.9±22.0	N/A	**704.5±13.5**
Failure Modes (See Section 4.3)	-	(A) 2, (B) 1, (C) 1, (D) 1	-	(A) 1, (B) 0, (C) 5, (D) 1	(A) 2, (B) 1, (C) 1, (D) 1	(A) 1, (B) 4, (C) 6, (D) 1	(A) 5, (B) 2, (C) 2, (D) 1

Failure Modes

(A) Timeout, unable to determine termination: This is the most common failure case of SGTM 2.0 across all tiers, especially in Tier 3. Causes include:

1. The system inadvertently manipulates the cable into a state that is difficult to perceive and manipulate, most common in Tier 3 due to higher complexity and a higher chance that a rare failure in disambiguation may accidentally tighten or complicate a knot.

2. A substantial portion of the cable leaves the workspace. The system repeatedly attempts exposure actions, but due to the large mass of cable, the cable continually slips back down into its prior configuration.

3. Though the entire cable may enter a difficult configuration, the algorithm slowly disambiguates it and given more time, may have untangled and terminated.

(B) Cable or knot leaves observable/reachable workspace: This is the next most common failure mode of SGTM 2.0. While performing a cage-pinch dilation or Reidmeister move, one gripper may miss the grasp, causing the entire cable to slide to one side of the workspace and fall off entirely and irrecoverably. This failure mode shows that slack management can be improved in future work.

(C) False termination due to missed knot detection: False termination is the most common failure mode in SGTM 2.0(-IP), mostly resolved by SGTM 2.0 with uncertainty-based components. This failure also occurs in SGTM 2.0, largely due to rarer cases where the knotted portion inadvertently lands outside the observable workspace during an incremental Reidemeister move, causing early termination. This can be addressed with improved motion primitives.

(D) Irrecoverable YuMi system error: These relatively rare issues result from the YuMi losing connection to the computer running the algorithm and freezing.

Ablations

We run ablations on Tier 2 and Tier 3 to compare the performance of SGTM 2.0 to the performance of SGTM 2.0(-IP), which uses the exact same algorithm as SGTM 2.0, but with the following uncertainty-based components removed:

- Reidemeister move due to tracing uncertainty.
- Ensemble network for keypoint predictions and partial cage-pinch dilation in the case of ensemble uncertainty in the cage-pinch dilation network.
- Intermediate views for incremental Reidemeister move.

We find, as shown in table 5.2, that SGTM 2.0(-IP) achieves a lower success rate than SGTM 2.0 on Tier 2. While SGTM 2.0 and SGTM 2.0(-IP) achieve the same success rate on Tier 3, the main failure case for SGTM 2.0 is timeout as higher complexity cases tend to require more time to disambiguate and untangle. In comparison, the main failure case for SGTM 2.0(-IP) are false termination. In fact, the most common failure case in SGTM 2.0(-IP) across Tier 2 and 3 is false termination, suggesting that sensitivity to observational uncertainty may be important for higher performance. Another implicit failure that results in more false terminations is over-tightening of knots to a diameter $\leq 3cm$. If the ensemble cage-pinch network has low confidence, SGTM 2.0 performs a partial cage-pinch dilation rather than a full cage-pinch dilation, preventing over-tightening knots in the case of poorly predicted cage-pinch points. Additionally, if the trace to a knot is uncertain, the Reidemeister move disambiguates the cable state, preventing poor grasps that may tighten or complicate the knots. The ablations suggest that these uncertainty-based primitives may prevent the over-tightening of knots and thus reduce false terminations.

4.4 Discussion

In this paper, we significantly extend our prior work on SGTM 1.0 to present SGTM 2.0, with novel uncertainty-based and active perception actions. SGTM 2.0 achieves an average untangling success rate of 83% and average rollout time of 351 seconds on cables with 1 or 2 knots, outperforming the prior state-of-the-art, SGTM 1.0, in untangling success by 43% and in untangling speed by 3x. Lastly, we find that introducing interactive perception – actively manipulating the cable to facilitate perception – improves untangling success on complex cases by 21%.

4.5 Individual Contribution

This chapter presents the content of our paper titled "SGTM 2.0: Autonomously Untangling Long Cables using Interactive Perception," which was published in ICRA 2023. The paper

represents a collaborative effort involving Kaushik Shivakumar, Anrui Gu, Yahav Avigal, Justin Kerr, Jeffrey Ichnowski, Richard Cheng, Thomas Kollar, and Prof. Ken Goldberg.

In this research, my contributions include an improved endpoint and knot detection approach, the development of the incremental reidemeister and exposure move, the implementation of observation uncertainty for termination, and the creation of the SGTM 2.0 algorithm to be deployed on the ABB YuMi robot.

I would like to commend Kaushik Shivakumar for his substantial contributions to this project. His involvement encompasses the development of the analytic cable tracer, the cage-pinch dilation point selection network, the cage-pinch dilation primitive, and the partial cage-pinch dilation primitive. He also proposed and implemented innovative ideas and methods for quantifying and utilizing trace and network confidence as a signal for executing interactive perception actions and integrated these into the system deployed on the robot. Furthermore, Kaushik created Figure 4.2 included in this chapter.

I would also like to acknowledge the contributions of Anrui Gu, who collaborated on the robot primitives, specifically the incremental reidemeister move and cage-pinch dilation. Anrui conducted experiments to evaluate the average precision and average recall of the endpoint and knot detection networks, and these findings are presented in Section 4.2. She also provided assistance in conducting physical experiments and contributed to the development of the website for this work: https://sites.google.com/view/sgtm2.

I extend my gratitude to Yahav Avigal, Justin Kerr, Jeffrey Ichnowski, and Professor Ken Goldberg for their guidance in shaping the project's direction. I would also like to express appreciation to Richard Cheng, Thomas Kollar, Yahav Avigal, and Professor Ken Goldberg once again for their meticulous review of multiple paper revisions, offering valuable feedback on framing the contributions and enhancing the overall writing.

Chapter 5

TUSK

5.1 Problem Statement

Previous chapters considered untangling long cables with only overhand and figure-8 knots. This chapter will break that assumption and aims to bring a long (3 m) cable containing *any semi-planar knots* into an untangled configuration, where no knots remain (knots defined in Section 5.1).

The workspace is defined by an (x, y, z) coordinate system and consists of a bilateral robot and a foam-padded manipulation surface, which lies in the (x, y) plane. The workspace also contains a fixed overhead RGB-D camera that faces the manipulation surface and outputs grayscale images and depth data. However, depth data is not used in TUSK. Rather, it is only used during manipulation. We work with a 300 cm cable. We assume the cable is visually distinguishable from the manipulation surface, its initial configuration has at least one endpoint visible, and is semi-planar as assumed in Grannen et al. [12], meaning each crossing in the knot has at most 2 intersecting cable segments. For perception experiments, we work with knots as tight as 5 cm in diameter. For physical experiments, due to robot graspability constraints, we work with knots of varying density, or approximate diameter, upwards of 10 cm in diameter. We define cable state to be $\theta(s) = \{(x(s), y(s), z(s))\}$ where s is an arc-length parameter that ranges $[0, 1]$, representing the normalized length of the cable. Here, $(x(s), y(s), z(s))$ is the location of a cable point at a normalized arc length of s from the cable's first endpoint. We also define the range of $\theta(s)$—that is, the set of all points on the cable at time t—to be $\mathcal{C}_t$.

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 [36]. This imaginary cable segment passes above the manipulation surface to complete the loop between p_1 and p_2 ("$p_1 \to p_2$ loop"). A knot exists between p_1 and p_2 at time t if no combination of Reidemeister moves

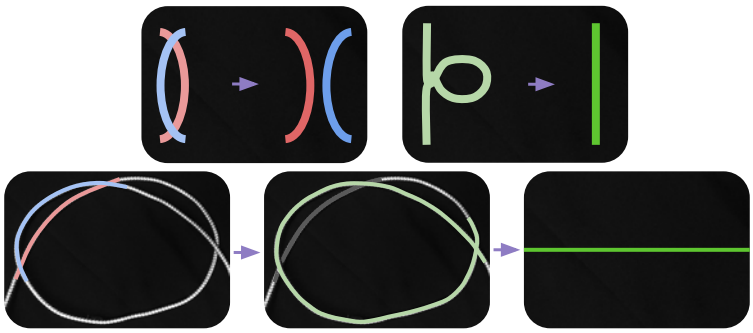


Figure 5.1: **Reidemeister Moves and Crossing Cancellation**: Top left depicts Reidemeister Move II. Top right depicts Reidemeister Move I. The bottom row shows that by algorithmically applying Reidemeister Moves II and I, we can cancel trivial loops, even if they visually appear as knots.

I, II (both shown in Figure 5.1), and III can simplify the $p_1 \to p_2$ 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 $k(s) : [0, 1] \to \{0, 1\}$ 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.

5.2 Methods

We present TUSK (Tracing to Untangle Semi-planar Knots) a system that takes grayscale images as input and reconstructs the state of a cable with semi-planar knots and crossings, performs knot detection, and selects graspable points for untangling. The first component is an iterative, learned cable tracer which estimates the path the cable takes through an image observation, combined with a crossing classifier which classifies over and under-crossings. Together, these estimate the state of the cable. TUSK then analyzes the state to detect knots and find graspable points for untangling. We will reference these points as *cage-pinch points* since during manipulation, one of these points receives a pinch grasp and the other a cage grasp (Section 5.3).

Learned Cable Tracer

We frame the problem of tracing a cable as estimating the most likely sequence of points that the cable passes through, where the goal of each step is to produce a probability distribution over the next point given the past points. This module estimates the spline ("trace") of the cable in an image by sequentially performing inference using a neural network on image crops. At each step, the model takes in a crop of the image along with trace points from previous iterations and predicts a heatmap, where we interpret the argmax as the next

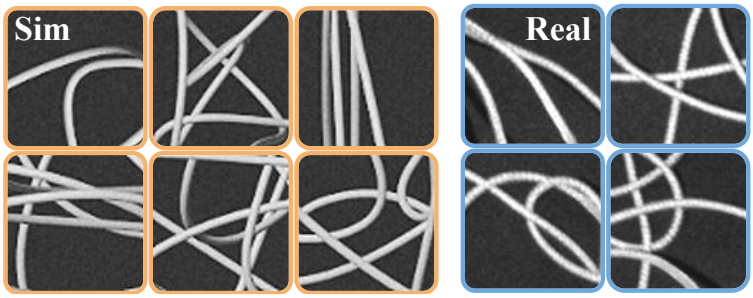


Figure 5.2: **Simulated and real crops used for training TUSK**: On the left are simulated cable crops augmented with Gaussian noise, brightness, and sharpening to match the real images (right) as closely as possible.

point along the trace. Since it operates on crops, the model suffers less from overfitting and benefits from more easy sim-to-real transfer as it operates on local information about the cable, rather than global visual and geometric appearance of knots.

Initialization

To initialize the trace, we supply a start pixel along the cable (in practice, one endpoint). We use an analytic tracer as in [44] to trace approximately 96 pixels, then use these points to initialize the learned tracer, as the model requires previous trace points to predict the next point along the trace.

Model Architecture and Inference

After initialization, the tracer sequentially applies a learned model to grow the trace. At each step, the network receives an input of an overhead image cropped to the center of the last predicted trace point, 64×64 pixels. To provide the model with information about the cable's previous path, we fuse the previous points of the trace into a gradient segmented line with the same thickness as the cable. The most recently traced point is brightest and the line decreases in brightness until it exits the crop. This is included in one channel of the input image. The other two channels contain an identical version of the grayscale image. The input dimension to the model is thus $64 \times 64 \times 3$. The model outputs a $64 \times 64 \times 1$ heatmap indicating the likelihood of each pixel being the next step in the cable trace. We choose the highest point in this heatmap as the next point in the trace. This process is applied iteratively until the tracer reaches another endpoint or leaves the visible workspace.

We use the UNet architecture for the model, which is known to be effective in image segmentation tasks [17]. Section 5.2 describes the dataset and training process. Each point in the trace is approximately 12 pixels apart, chosen by grid search to provide a balance between adding context and reducing overfitting. To reduce the input space for the model and thus increase data efficiency, we pre-rotate the input image such that the last two points of the trace are aligned horizontally and the trace always travels left to right with the most recently traced point being the right most point. We explore another important tradeoff

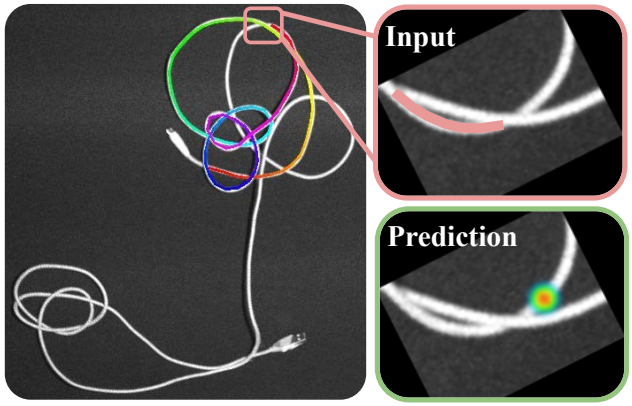


Figure 5.3: **Input and Prediction of Iterative Learned Cable Tracer**: the iterative learned cable tracer takes small crops around the cable and one step at a time, predicts the next point in the trace. The input crop is a 64x64x3 crop centered on the previous trace point. The first channel contains the previous trace points within the crop. The second and third channel are the gray scale image of the crop. The prediction is a 64x64 heatmap, where we infer the argmax of the heatmap to be the next point in the trace.

between context and overfitting via grid search by tuning the size of the crop, 64×64, and number of previous points inputted into the model, 3.

Dataset and Model Training

To train the Learned Tracer model, we leverage simulation to procedurally generate a dataset which encompasses a wide distribution of crossing configurations. sing Blender [7], we collect a dataset of 30,000 simulated grayscale images, whose visual appearance closely matches real observations (Fig. 5.2). Cable configurations are generated via random Bezier curves through the following 3 methods: (1) a weighted combination of successively choosing random points outside of a small exclusion radius around the current point, (2) segments intentionally designed to be near-parallel, and (3) other sections of cable designed to appear dense and knot-like by confining certain segments of the cable to regions in space.

We sample image crops randomly along the cable, with 95% of samples distributed on cable crossings, as these represent the challenging cases. The simulated images are augmented with Gaussian noise with standard deviation of 6, brightness with standard deviation of 5, and sharpening to imitate the appearance of real cable crops. Additionally, we augment the dataset with a smaller dataset of 568 real, grayscale cable crop images hand-labeled with splines, sampled during training such that real images are approximately 20% of the examples seen. During training we use the Adam optimizer [20] with pixelwise binary cross-entropy loss, using a batch size of 64 and learning rate of 10^{-5}.

Over/Undercrossing Predictor

To convert the 2D trace of a cable into a topology for the downstream task of untangling, we use a convolutional neural network (CNN) to classify over and under-crossings in the cable.

Data and Model Input

We use simulated and real over/undercrossing crops of size 20x20. Similar to the cable tracer, the 568 real images are oversampled such that they are seen 20% of the time during training. The simulated data is augmented in the same manner to the cable tracer to imitate the appearance of real cable crossings when observed as 20x20 crops. We provide the network with a 20x20x3 crop as input. The first channel encodes the points of the trace indicating the cable segment of interest (in other words, the cable segment with respect to which we aim to classify the crossing as an over/undercrossing). Similar to how the points are inputted for the learned tracer, the points are fused together into a line segment, but now the line segment does not decrease in brightness as points become less recent. The crop is rotated so the first and last point in the line segment are horizontal. The second channel is a Gaussian heatmap centered at the position of the crossing we aim to classify. This helps deal with dense configurations that can have nearby consecutive crossings captured in the same crop. By receiving a position of interest, the network learns to ignore other crossings. Lastly, as all images are grayscale, the third channel encodes the grayscale image of the cable crossing.

Model Architecture and Inference

We use a ResNet-34 classification model to output a prediction score in the interval $[0, 1]$. This model is trained using binary cross-entropy loss with a single output unit with sigmoid activation. We tune a threshold to binarize the output by determining 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. Based on the tuning results, we obtain a threshold of 0.275 such that a prediction score < 0.275 corresponds to an undercrossing prediction and a score ≥ 0.275 corresponds to an overcrossing prediction. We output the raw prediction score along with a scaled confidence value (ranging $[0.5, 1]$) indicating the probability associated with the classifier's prediction.

Analytic Knot Detection

We construct line segments between consecutive points on the trace outputted by the learned cable tracer (Section 5.2). Crossings are located at the points of intersection of these line segments. We use the crossing classifier (Section 5.2) 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 5.2.

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. To reduce the number of actions required to successfully untangle the cable, we algorithmically apply Reidemeister moves I and II to discard non-essential crossings (Fig. 5.1). We exclude Reidemeister move III from this scheme as it does not lead to a direct reduction in the number of crossings, unlike moves I and II. We

are allowed to perform this algorithmic manipulation as Reidemeister moves maintain knot equivalence [36].

Crossing Correction

Given the assumption of knot semi-planarity, a single crossing location must contain one overcrossing and one undercrossing. In situations where the over/undercrossing classifier incorrectly predicts that the crossings at a location are both overcrossings or both undercrossings, we defer to the detection with higher confidence to correct the crossing assignment. The algorithm updates the probability associated with the corrected crossing to $1-$ its original value. This is to take into account model uncertainty when calculating confidence scores for the overall knot.

Crossing Cancellation

Crossing cancellation allows for the simplification of cable structure by removing non-essential crossings, shown in Figure 5.1. It allows the system to filter out some trivial configurations. 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 $\mathcal{X}'$.

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 5.1): 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.

Algorithmic Cage-Pinch Point Detection

As per the definition introduced in Section 5.2, 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 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 5.4.

5.3 Robot Untangling using TUSK

Manipulation Primitives

We use the same primitives as in SGTM 2.0 (Sliding and Grasping for Tangle Manipulation 2.0) [44] to implement TUSK as shown in Figure 5.5 for untangling long cables. We add a *perturbation* move.

Cage-Pinch Dilation

We use cage-pinch grippers as in Viswanath et al. [53]. 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. [44] separates the grippers to a small, fixed distance of 5 cm.

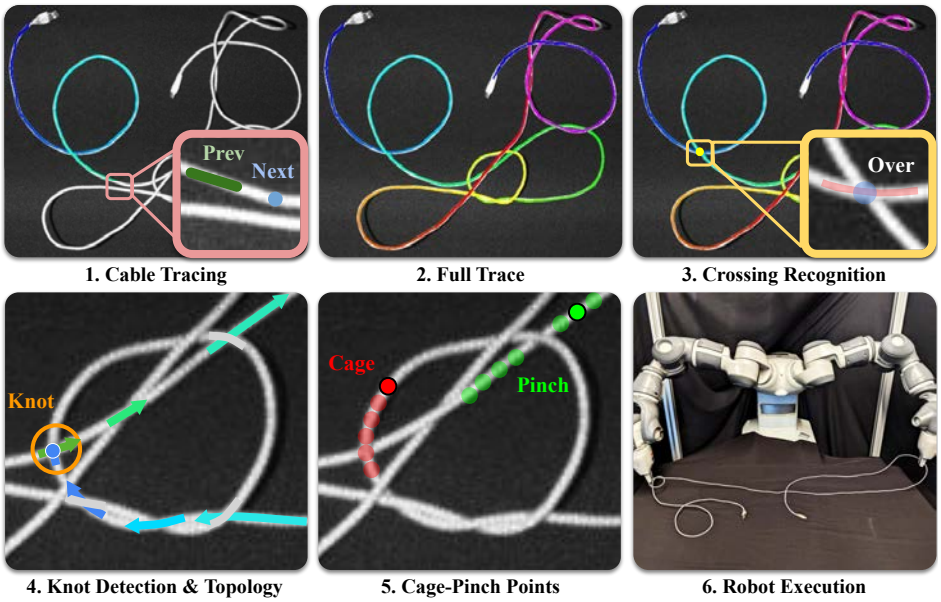


Figure 5.4: **TUSK:** TUSK first performs cable tracing (1, 2). The trace is shown through a rainbow gradient (from violet to purple), depicting the sequence in which the cable is traced. After tracing, TUSK does crossing recognition (3) to obtain the full topology of the cable. Next, using crossing cancellation rules from knot theory, it analytically determines knots (4) in the cable. Next, TUSK surveys possible cage-pinch points (5) and selects the best candidate points to grasp to execute a cage-pinch dilation action, untangling the knot (6).

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.

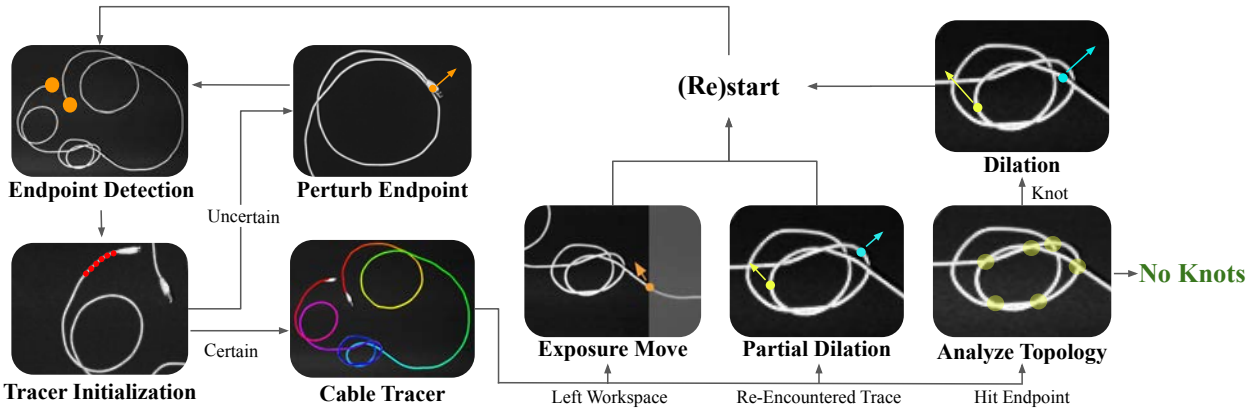


Figure 5.5: **Untangling Algorithm with TUSK**: We first detect the endpoints and initialize the tracer with start points. If we are not able to obtain start points, we perturb the endpoint and try again. Next, we trace. While tracing, if the cable exits the workspace, we pull the cable towards the center 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.

Cable Untangling System

Combining TUSK and the manipulation primitives from Section 5.3, the cable untangling algorithm works as follows: First, detect endpoints and initialize the learned tracer with 6 steps of the analytic tracer. If TUSK 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 [44]. 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 5.5.

5.4 Experiments

We test the performance of 1) TUSK, 2) the learned cable tracer, and 3) TUSK applied to autonomous robot untangling.

Workspace

The workspace consists of a 1 m $\times$ 0.75 m surface with a bimanual ABB YuMi robot and an overhead Photoneo PhoXi camera with 773 $\times$ 1032 $\times$ 4 RGB-D observations. Although

there are 3 color channels, images outputted by the PhoXi are grayscale. Additionally, the workspace is padded with a 5 cm tall piece of foam and covered with a black cloth.

TUSK Setup

To test TUSK, we use a single 3 m, white, braided USB-A to micro-USB cable to the workspace.

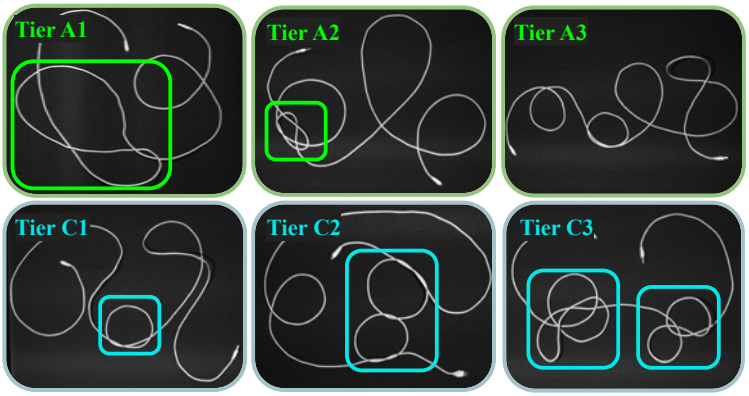


Figure 5.6: **Starting configurations for** the 3 categories for TUSK experiments and the 3 levels for physical experiments.

We test TUSK on 3 different categories of cable configurations, shown in Figure 5.6. The ordering of the categories for these experiments does not indicate varying difficulty. Rather, they are 3 categories of knot configurations to test TUSK on.

1. **Tier A1**: Loose (35-40 cm in diameter) figure 8, overhand, overhand honda, bowline, linked overhand, and figure 8 honda knots.

2. **Tier A2**: Dense (5-10 cm in diameter) figure 8, overhand, overhand honda, bowline, linked overhand, and figure 8 honda knots.

3. **Tier A3**: Fake knots (trivial configurations positioned to appear knot-like from afar).

We evaluate TUSK on the 3 categories across the following ablations:

1. SGTM 2.0 perception system: using a Mask R-CNN model trained on overhand and figure-8 knots for knot detection.

2. TUSK (-LT): replacing the Learned Tracer with the same analytic tracer from Shivakumar et al. [44] as described in Section 5.4 combined with the topology identification and knot detection methods without learned tracing.

3. TUSK (-CC): using the learned tracer and topology identification scheme to do knot detection without Crossing Cancellation, the iterative algorithmic application of Reidemeister moves I and II.

4. TUSK: the full perception system.

We report the success rate of each of these algorithms in the following manner. If a knot is present, the algorithm is successful if it correctly detects the first knot and correctly labels the first undercrossing corresponding to that knot. If there are no knots, the algorithm is successful if it correctly detects no knots.

Tracing in Multi-Cable Settings Setup

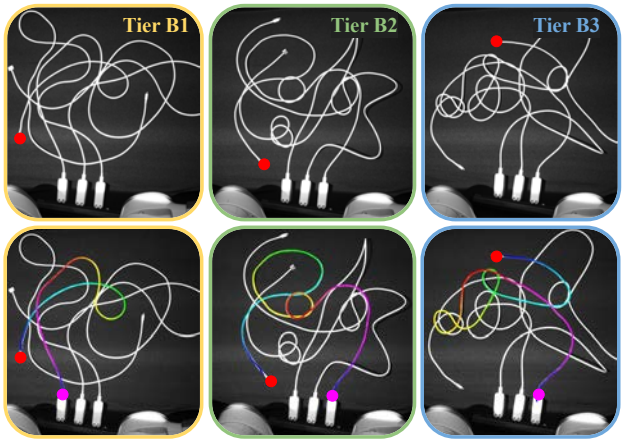


Figure 5.7: **Multi-cable tracing**: **Top row:** illustrative examples of each of the 3 tiers of difficulty for multi-cable tracing experiments. **Bottom row:** Corresponding successful traces outputted by the learned tracer.

For this set of perception experiments, the workspace contains a power strip. Attached to the power strip are 3 MacBook adapters, with two 3m USB-C to USB-C cables and one 2m plain white USB-C to MagSafe 3 cable. This setup is depicted in the top row of Figure 5.7. We evaluate perception on multi-cable settings on 3 tiers of difficulty.

1. **Tier B1**: No knots; cables are dropped onto the workspace, one at a time.
2. **Tier B2**: Each cable is tied with a single knot that is 5-10 cm in diameter (one of figure 8, overhand, overhand honda, bowline, linked overhand, and figure 8 honda). The cables are then randomly dropped onto the workspace, one after another.
3. **Tier B3**: Each cable is tied with another cable through the following 2 cable knot types (square, carrick bend, and sheet bend) with up to 3 knots in the scene. As in the other tiers, the cables are randomly dropped onto the workspace, one by one.

Across all 3 tiers, we assume the cable of interest cannot exit and re-enter the workspace and that crossings must be semi-planar. Additionally, we pass in the locations of all 3 adapters to the tracer and an endpoint to initialize from. To account for noise in the input images, we take 3 images of each configuration and count the experiment a success if 2/3 have the correct trace and reach their corresponding adapter, otherwise we report a failure.

We evaluate the performance of the learned tracer from TUSK against an analytic tracer from Shivakumar et al. [44] as a baseline with scoring rules inspired by the work of Lui et al. [28] and Keipour et al. [19]. The analytic tracer explores all potential paths and determines the most correct trace through a scoring metric from [44]. The scoring metric prefers paths that reach an endpoint, discarding traces that do not reach adapters. This is because the scoring metric sees reaching an endpoint as indicative of completing a trace. Of the paths that reach an endpoint, the trace returned is the one with the least sharp angle deviations and the highest coverage score.

Physical Robot Untangling Setup

Physical experiments are conducted on a single 3 m, white, braided USB-A to micro-USB cable, which is added to the workspace.

We evaluate TUSK in untangling performance on the following 3 levels of difficulty (Figure 5.6), where all knots are upward of 10 cm in diameter, and compare performance to SGTM 2.0 [44]:

1. **Tier C1:** A cable consisting of an overhand, figure 8, or overhand honda knot. The full cable configuration has ≤ 6 crossings.
2. **Tier C2:** A cable consisting of a bowline, linked overhand, or figure 8 honda knot. The full cable configuration has ≥ 6 and < 10 crossings.
3. **Tier C3:** A cable consisting of 2 knots (one of a knot class from Tier C2 and one of a knot class from Tier C1). The full cable configuration has ≥ 10 and < 15 crossings.

Similar to Shivakumar et al. [44], we use a 15-minute timeout on each rollout. We report metrics including success rate for untangling 1 and 2 knots, as well as the time to do so. We also report the success rate for termination, as well as the time required to do so, as a fraction of the number of rollouts that succeeded in fully untangling the cable.

Generalizing Tracing to Varying Cables Setup

For this perception experiment, the workspace contains a single cable with one of the following knots: overhand, figure-8, overhand honda, or bowline. In this experiment, we provide the two endpoints so we test the tracer in isolation, independent of endpoint detection. We report a trace as successful if the progression of the trace correctly follows the path of the cable. At any point, if it leaves the cable, even if it re-enters at a correct location, we report that as a failure.

Table 5.1: TUSK Experiments

	SGTM 2.0	TUSK (-LT)	TUSK (-CC)	TUSK
Tier A1	2/30	14/30	20/30	**24/30**
Tier A2	**28/30**	8/30	21/30	26/30
Tier A3	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 5.2: TUSK and Physical Robot Experiments (90 total trials)

	Tier C1		**Tier C2**		**Tier C3**	
	SGTM 2.0	TUSK	SGTM 2.0	TUSK	SGTM 2.0	TUSK
Knot 1 Success Rate	11/15	**12/15**	6/15	**11/15**	9/15	**14/15**
Knot 2 Success Rate	-	-	-	-	2/15	**6/15**
Verification Rate	**11/11**	8/12	**6/6**	6/11	**1/2**	2/6
Avg. Knot 1 Time (min)	1.09±0.12	2.11±0.25	3.45±0.74	3.88±1.09	1.84±0.38	2.00±0.42
Avg. Knot 2 Time (min)	-	-	-	-	3.11±1.18	7.45±1.55
Avg. Verif. Time (min)	5.71±0.88	6.13±1.44	6.35±1.81	10.10±0.67	5.38	9.58±1.48
Failures	(7) 4	(1) 2, (2) 1	(1) 3, (5) 6 (5) 1	(2) 2, (4) 1, (5) 3 (6) 2, (7), 2	(1) 3, (2) 3	(1) 2, (2) 3 (3) 1, (6) 3

5.5 Results

TUSK

As summarized in Table 5.1, TUSK outperforms SGTM 2.0, TUSK (-LT), and TUSK (-CC) on categories 1 and 2. SGTM 2.0 outperforms TUSK in tier A2. This is because the Mask R-CNN is trained on dense overhand and figure 8 knots. While other knots in tier A2 are out of distribution, they visually resemble the overhand and figure 8 knots. The network is, therefore, able to still detect them as knots.

Failure Modes

(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.

Table 5.3: Multi-Cable Tracing Results

	Analytic	Learned
Tier B1	3/30	**27/30**
Tier B2	2/30	**23/30**
Tier B3	1/30	**23/30**
Failures	(I) 3, (II) 45, (III) 36	(I) 14, (II) 1, (III) 2

(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 TUSK (-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 TUSK (-CC), the most common failure modes are (B) and (E). This is because TUSK (-CC) is unable to distinguish between trivial loops and knots without the crossing cancellation scheme. By the same token, TUSK (-CC) is also unable to fully isolate a knot from surrounding trivial loops. For TUSK, the most common failure mode is (B). However, this is a derivative of failure mode (D), which is present in TUSK (-LT), TUSK (-CC), and TUSK. 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.

Tracing through Multi-Cable Settings

Table 5.3 shows that the learned tracer significantly outperforms the baseline analytic tracer on all 3 tiers of difficulty with a total of 81% success across the tiers.

Failure Modes:

(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 B3, is (I). One reason for such failures is the presence of multiple twists along the cable path (particularly in Tier B3 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 B2, 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.

Physical Robot Untangling

Results in Table 5.2 show that our TUSK-based untangling system (29/45) outperforms SGTM 2.0 (19/45) in untangling success rate across 3 tiers of difficulty. SGTM 2.0 is, however, faster than TUSK in each of the 3 tiers. This is due to the fact that TUSK requires a full trace of the cable. TUSK also requires the full cable to be in view in order to claim termination, which is difficult to achieve as the cable is 3 $\times$ as long as the width of the workspace. Because the cable falls in varying complex configurations, many of which leave the visible workspace, the untangling algorithm performs cable reveal moves before detecting knots. This increases the time needed to untangle and verify that the cable is untangled, causing some runs to time out before verification.

On the other hand, SGTM 2.0 has false termination as its main failure mode because it does not account for the cable exiting the workspace. This is beneficial for speed because the system terminates as early as possible. However, the system fails when an off-workspace knot remains and goes undetected. This allows rollouts to end quickly, even if the cable is not untangled.

Failure Modes:

(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 TUSK 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

Table 5.4: Generalizing Tracing Cable Information

Cable Reference	Length (m)	Color	Texture	Physical Properties
TR (trained with)	2.74	White/gray	Braided	Slightly stiff
1	2.09	Gray	Rubbery	slightly thicker than TR
2	4.68	Yellow with black text	Rubbery and plastic	Very stiff
3	2.08	Tan	Rubbery	highly elastic
4	1.79	Bright red	Braided	Flimsy
5	4.61	White	Braided	Flimsy

Table 5.5: Generalizing Tracing 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	

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.

Generalizing Tracing to Varying Cables

Results in Table 5.5 show TUSK can generalize to various cable appearances, textures, lengths, and physical properties. TUSK performs just as equally on cable TR (the cable trained with) as it does on any of the other cables.

Failure Modes:

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

(2) Missteps onto a parallel cable.

(3) Skips a loop.

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.

5.6 Conclusion

This work presents TUSK, a perception pipeline that iteratively traces and determines the topology of semi-planar cable configurations, detects knots given the cable state, and detects graspable points for untangling the knots. Experiments show that TUSK can successfully trace a single cable in a multi-cable setting with 81% accuracy, significantly outperforming an analytic baseline. TUSK is also able to detect knots with 77% accuracy. When TUSK is applied to a robot untangling problem, the system is able to achieve 64% success in untangling.

5.7 Individual Contribution

This chapter is derived from our paper titled "Learning to Trace and Untangle Semi-planar Knots (TUSK)," a collaborative effort involving Kaushik Shivakumar, Jainil Ajmera, Mallika Parulekar, Justin Kerr, Jeffrey Ichnowski, Richard Cheng, Thomas Kollar, and Prof. Ken Goldberg.

My contributions include the development of the models for the learned cable tracer and the over/undercrossing predictor, the algorithmic cage-pinch point detection, the perturbation primitive, and the TUSK and cable untangling system full-stack integration onto the ABB YuMi robot.

I would like to acknowledge Kaushik Shivakumar for his valuable contributions, including the creation of a framework for generating simulation data used in training the learned cable tracer and over/undercrossing predictor, learned cable tracer development and refactoring of the model code-base which greatly facilitated efficient iteration and tuning. He also jointly contributed to Figure 5.4.

I also commend Jainil Ajmera and Mallika Parulekar for their exceptional work in ideating and developing an analytic knot detection framework, which comprised three key components: crossing correction, crossing cancellation through algorithmic Reidemeister moves, and algorithmic knot detection. They also worked on generating ground truth simulation data with Kaushik. Additionally, Jainil and Mallika conducted physical experiments and contirbuted to writing the paper, including the creation of Figures 5.1 and 5.5.

I express my gratitude to Justin Kerr and Jeffrey Ichnowski for their pivotal roles in scoping the project and providing guidance every week through weekly project meetings. I would also like to extend my thanks to Richard Cheng and Thomas Kollar from TRI for their valuable insights in shaping the project within the context of home robotics. Lastly, I am sincerely grateful to Professor Ken Goldberg for his continuous guidance and invaluable feedback at every stage of this project.

Chapter 6

Limitations

6.1 Limitations of SGTM 1.0

SGTM 1.0 was limited in that first, the perception approach in this work is limited to overhand and figure-8 knots due to reliance on geometric rather than topological features. Additionally, untangling success rates and verification rates were low. This may be attributed to unrefined primitives and a need for more interactive perception primitives that will disambiguate the cable state. Lastly, the physical tracing stop condition perception component is reliant on depth data. Obtaining a high quality depth camera for a skill-full task such as untangling cables is expensive, making this method limited in its ability to be deployed in various settings.

6.2 Limitations of SGTM 2.0

In SGTM 2.0, the perception system operates now only on RGB data. However, grasping is still reliant on depth. Another limitation that carried over from SGTM 1.0 is the limitation to only overhand and figure-8 knots. Lastly, this algorithm has been tested on only a single white cable on a black background and it would be good to improve that to varying cable appearances and varying backgrounds.

6.3 Limitations of TUSK

TUSK can now generalize to all semi-planar knots rather than only overhand and figure-8 knots. The robot system executing TUSK still depends on a depth camera for grasping, which future work will address. Details on plans for addressing this can be found in Section 7.2. We conduct experiments that show TUSK generalizing to cables of difference appearances, discussed in Section 5.5. Visual results can be found at https://sites.google.com/view/tusk-rss/home. However, it cannot generalize to varying cable thickness. Additionally, TUSK is

still dependent on a mono-color workspace, limiting its use to more cluttered and patterned areas such as homes or warehouse settings. Lastly, while the perception system can handle dense cable configurations, on the manipulation side, the untangling system struggles to grasp tight loops, which future work will aim to address.

Chapter 7

Conclusions and Future Work

7.1 Conclusion

This thesis aims to generalize perception learning methods through two avenues: interactive perception and state estimation.

In Chapter 3, SGTM 1.0 first aims to ease perception with generic manipulation primitives such as shaking. We realized this was suboptimal and in fact, was a main source of failure modes as without precise, targeted primitives, the robot ends up further disrupting the cable state rather than disambiguating it.

In Chapter 4, SGTM 2.0 aims to generalize perception to out of distribution cases using interactive perception. In cases of trace uncertainty, observational uncertainty, or network uncertainty, SGTM 2.0 performs interactive perception moves such as reidemeister move, exposure move, or partial cage-pinch dilation to slightly perturb the state and retry perception, testing if the cable state is more in distribution to what is seen in training. Results show interactive perception greatly improved accuracy and speed, improving untangling success by 43% and untangling speed by 3x.

In Chapter 5, TUSK takes a different spin on the idea of generalizing perception and aims to perform broader full cable state estimation to remove the restriction of untangling algorithms to knots seen in training. This was necessary to improve untangling success as a main failure mode of SGTM 1.0 and SGTM 2.0 was that the system inadvertently manipulates the cable into a state that is difficult to perceive and too difficult for even interactive perception from SGTM 2.0 to resolve. By creating a general state estimator, we are able to untangle a broader class of knots and better address a variety of cable configurations. This is seen in physical untangling experiments where on the harder tiers (tier 2 and 3), TUSK applied to complex knots achieves 73% and 40% success, while SGTM 2.0 achieves 40% and 13% success. More importantly, this method is able to function beyond the untangling framework and can be applied to various downstream tasks that require 1D deformable object state estimation.

7.2 Future Work

As future work, we are actively exploring learned interactive perception which will aim to combine TUSK state estimation techniques with interactive perception. The goal is to learn interactive perception in the form of an RL framework that can generalize to various tasks, such as increasing knot visibility (1D), cloth smoothing (2D), and bag opening (3D). This work is still in early stages, but the team is gaining inspiration from the prior Recovery RL papers in our lab [48] to generate a framework with a π_{task} and learn a $\pi_{recovery}$ and switching policy where based on a threshold perception confidence, switches between the two. Confidence for now is a plug-in heuristic value based on the task and potentially considers history. This project aims to resolve limitations from prior work with a method that is better robust to handling cases out of training distribution for π_{task}.

Another avenue future work we will explore is removing depth data and substituting servoing policies to grasp thin, 1D deformable objects in dense configurations. Some ideas currently under exploration include estimating the angle of travel of a cable using stereo vision to determine gripper orientation and a small perturbation or wiggling action observed in a video loop to verify a successful grasp.

Ending Notes

The past three years in AUTOLab have been an invaluable experience. The main things I learned from this experience were 1) how to perform quality research and 2) and arguably just as important, presenting your work such that it is clear and accessible to all. Coming in with little research experience, I learned that quality research comes from inspiring yourself from the nuances of daily life and exploring existing literature. I also learned that presenting your work clearly facilitates quality research as it becomes more accessible to people, fostering fruitful conversations that inspire more ideas. My journey on general cable state estimation and interactive perception for perception confidence does not end here and I hope to see future work on this project thrive. I hope to continue research in the future as I begin a job at Google.

Bibliography

[1] Yahav Avigal et al. *SpeedFolding: Learning Efficient Bimanual Folding of Garments*. 2022.

[2] Ruzena Bajcsy. "Active perception". In: *Proceedings of the IEEE* 76.8 (1988), pp. 966–1005.

[3] Ruzena Bajcsy, Yiannis Aloimonos, and John K Tsotsos. "Revisiting active perception". In: *Autonomous Robots* 42.2 (2018), pp. 177–196.

[4] Jeannette Bohg et al. "Interactive perception: Leveraging action in perception and perception in action". In: *IEEE Transactions on Robotics* 33.6 (2017), pp. 1273–1291.

[5] Lawrence Yunliang Chen et al. *AutoBag: Learning to Open Plastic Bags and Insert Objects*. 2022.

[6] Cheng Chi et al. "Iterative Residual Policy for Goal-Conditioned Dynamic Manipulation of Deformable Objects". In: *Proceedings of Robotics: Science and Systems (RSS)*. 2022.

[7] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation. Stichting Blender Foundation, Amsterdam, 2018.

[8] Michael Danielczuk et al. "Mechanical search: Multi-step retrieval of a target object occluded by clutter". In: *2019 International Conference on Robotics and Automation (ICRA)*. IEEE. 2019, pp. 1614–1621.

[9] Peter R Florence, Lucas Manuelli, and Russ Tedrake. "Dense object nets: Learning dense visual object descriptors by and for robotic manipulation". In: *Conf. on Robot Learning (CoRL)*. 2018.

[10] Aditya Ganapathi et al. "Learning to Smooth and Fold Real Fabric Using Dense Object Descriptors Trained on Synthetic Color Images". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2021.

[11] Kenneth Y Goldberg and Ruzena Bajcsy. "Active touch and robot perception". In: *Cognition and Brain Theory* 7.2 (1984), pp. 199–214.

[12] Jennifer Grannen et al. "Untangling dense knots by learning task-relevant keypoints". In: *Conference on Robot Learning* (2020).

[13] Huy Ha and Shuran Song. "Flingbot: The unreasonable effectiveness of dynamic manipulation for cloth unfolding". In: *Conference on Robot Learning*. PMLR. 2022, pp. 24–33.

[14] Ryan Hoque et al. "Visuospatial foresight for multi-step, multi-task fabric manipulation". In: *Robotics: Science and Systems (RSS)* (2020).

[15] Ryan Hoque et al. "Learning to Fold Real Garments with One Arm: A Case Study in Cloud-Based Robotics Research". In: ().

[16] Xuzhao Huang et al. "Untangling Multiple Deformable Linear Objects in Unknown Quantities With Complex Backgrounds". In: *IEEE Transactions on Automation Science and Engineering* (2023), pp. 1–13.

[17] Pavel Iakubovskii. *Segmentation Models Pytorch*. https://github.com/qubvel/segmentation_models.pytorch. 2019.

[18] Russell C. Jackson et al. "Real-Time Visual Tracking of Dynamic Surgical Suture Threads". In: *IEEE Transactions on Automation Science and Engineering* 15.3 (2018), pp. 1078–1090.

[19] Azarakhsh Keipour, Maryam Bandari, and Stefan Schaal. "Deformable One-Dimensional Object Detection for Routing and Manipulation". In: *CoRR* abs/2201.06775 (2022). arXiv: 2201.06775.

[20] Diederik P Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations*. 2015.

[21] Thomas Kollar et al. "Simnet: Enabling robust unknown object manipulation from pure synthetic data via stereo". In: *Conference on Robot Learning*. PMLR. 2022, pp. 938–948.

[22] Robert Lee et al. "Learning arbitrary-goal fabric folding with one hour of real robot experience". In: *Conference on Robot Learning* (2020).

[23] Tsung-Yi Lin et al. "Microsoft coco: Common objects in context". In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[24] Xingyu Lin et al. "Learning visible connectivity dynamics for cloth smoothing". In: *Conference on Robot Learning*. PMLR. 2022, pp. 256–266.

[25] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully Convolutional Networks for Semantic Segmentation". In: *CVPR* (2015).

[26] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[27] Wen Hao Lui and Ashutosh Saxena. "Tangled: Learning to Untangle Ropes with RGB-D Perception". In: *2013 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*. 2013.

[28] Wen Hao Lui and Ashutosh Saxena. "Tangled: Learning to untangle ropes with RGB-D perception". In: *2013 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*. IEEE. 2013, pp. 837–844.

[29] Jan Matas, Stephen James, and Andrew J Davison. "Sim-to-real reinforcement learning for deformable object manipulation". In: *Conference on Robot Learning*. PMLR. 2018, pp. 734–743.

[30] Hermann Mayer et al. "A system for robotic heart surgery that learns to tie knots using recurrent neural networks". In: *Advanced Robotics* 22.13-14 (2008), pp. 1521–1537.

[31] Dale McConachie et al. "Learning When to Trust a Dynamics Model for Planning in Reduced State Spaces". In: *CoRR* abs/2001.11051 (2020). arXiv: 2001.11051.

[32] Ashvin Nair et al. "Combining Self-Supervised Learning and Imitation for Vision-Based Rope Manipulation". In: *CoRR* abs/1703.02018 (2017). arXiv: 1703.02018.

[33] Tonci Novkovic et al. "Object finding in cluttered scenes using interactive perception". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 8338–8344.

[34] Nicolas Padoy and Gregory Hager. "Deformable Tracking of Textured Curvilinear Objects". In: *Proceedings of the British Machine Vision Conference*. BMVA Press, 2012, pp. 5.1–5.11.

[35] Paritosh Parmar. "Use of computer vision to detect tangles in tangled objects". In: *2013 IEEE Second International Conference on Image Information Processing (ICIIP-2013)*. IEEE, Dec. 2013.

[36] Kurt Reidemeister. *Knot theory*. BCS Associates, 1983.

[37] Shaoqing Ren et al. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6 (2017), pp. 1137–1149.

[38] Stephane Ross, Geoffrey J Gordon, and J Andrew Bagnell. "A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning". In: 2011.

[39] Jose Sanchez et al. "Robotic manipulation and sensing of deformable objects in domestic and industrial applications: a survey". In: *The International Journal of Robotics Research* 37.7 (2018), pp. 688–716.

[40] John Schulman et al. "Tracking deformable objects with point clouds". In: *2013 IEEE International Conference on Robotics and Automation*. 2013, pp. 1130–1137.

[41] Daniel Seita et al. "Deep imitation learning of sequential fabric smoothing from an algorithmic supervisor". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. 2020.

[42] Daniel Seita et al. "Learning to Rearrange Deformable Cables, Fabrics, and Bags with Goal-Conditioned Transporter Networks". In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2021.

[43] Yu She et al. "Cable Manipulation with a Tactile-Reactive Gripper". In: *The International Journal of Robotics Research* 40.12-14 (2021), pp. 1385–1401.

[44] Kaushik Shivakumar et al. "SGTM 2.0: Autonomously Untangling Long Cables using Interactive Perception". In: *arXiv preprint arXiv:2209.13706* (2022).

[45] Priya Sundaresan et al. "Learning rope manipulation policies using dense object descriptors trained on synthetic depth data". In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 9411–9418.

[46] Priya Sundaresan et al. "Untangling dense non-planar knots by learning manipulation features and recovery policies". In: *Proc. Robotics: Science and Systems (RSS)* (2021).

[47] Te Tang and Masayoshi Tomizuka. "Track deformable objects from point clouds with structure preserved registration". In: *The International Journal of Robotics Research* 41.6 (2022), pp. 599–614. eprint: https://doi.org/10.1177/0278364919841431.

[48] Brijen Thananjeyan et al. "Recovery rl: Safe reinforcement learning with learned recovery zones". In: *IEEE Robotics and Automation Letters* 6.3 (2021), pp. 4915–4922.

[49] Brijen Thananjeyan et al. *All You Need is LUV: Unsupervised Collection of Labeled Images using Invisible UV Fluorescent Indicators*. 2022.

[50] Constantine J Tsikos and Ruzena K Bajcsy. "Segmentation via manipulation". In: *Technical Reports (CIS)* (1988), p. 694.

[51] Jur Van Den Berg et al. "Superhuman performance of surgical tasks by robots using iterative learning from human-guided demonstrations". In: *2010 IEEE International Conference on Robotics and Automation*. IEEE. 2010, pp. 2074–2081.

[52] Vainavi Viswanath et al. "Disentangling Dense Multi-Cable Knots". In: *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)* (2021).

[53] Vainavi Viswanath et al. "Autonomously Untangling Long Cables". In: *Robotics: Science and Systems (RSS)* (2022).

[54] Vainavi Viswanath et al. "Learning to Trace and Untangle Semi-planar Knots (TUSK)". In: *arXiv preprint arXiv:2303.08975* (2023).

[55] Angelina Wang et al. "Learning robotic manipulation through visual planning and acting". In: *Robotics: Science and Systems (RSS)* (2019).

[56] Thomas Weng et al. "FabricFlowNet: Bimanual Cloth Manipulation with a Flow-based Policy". In: *Conference on Robot Learning*. PMLR. 2022, pp. 192–202.

[57] Bryan Willimon, Stan Birchfield, and Ian Walker. "Classification of clothing using interactive perception". In: *2011 IEEE International Conference on Robotics and Automation*. IEEE. 2011, pp. 1862–1868.

[58] Yilin Wu et al. "Learning to manipulate deformable objects without demonstrations". In: *Robotics: Science and Systems (RSS)* (2020).

[59] Yuxin Wu et al. *Detectron2*. https://github.com/facebookresearch/detectron2. 2019.

[60] Yuji Yamakawa et al. "One-handed knotting of a flexible rope with a high-speed multifingered hand having tactile sensors". In: *2007 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*. IEEE. 2007, pp. 703–708.

[61] Wilson Yan et al. "Learning Predictive Representations for Deformable Objects Using Contrastive Estimation". In: *Proceedings of the 2020 Conference on Robot Learning*. Ed. by Jens Kober, Fabio Ramos, and Claire Tomlin. Vol. 155. Proceedings of Machine Learning Research. PMLR, 16–18 Nov 2021, pp. 564–574.

[62] Tianhao Zhang et al. "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 5628–5635.