Probabilistic State Estimation to Enable Manipulation and Interactive Perception for Robotic Cable Untangling and Object Search

Kaushik Shivakumar
Ken Goldberg

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2023-129
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2023/EECS-2023-129.html

May 12, 2023

Acknowledgement

The work in this thesis is a result of substantial collaborations with Vainavi Viswanath, Jainil Ajmera, Mallika Parulekar, Justin Kerr, Satvik Sharma, and Raven Huang, among others. I'm forever grateful to Professor Ken Goldberg for the opportunity to work on research in his lab, as well as my friends and family for supporting me through it all.

Probabilistic State Estimation to Enable Manipulation and Interactive Perception for Robotic Cable Untangling and Object Search

by

Kaushik Shivakumar

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

Spring 2023

Probabilistic State Estimation to Enable Manipulation and Interactive Perception for Robotic Cable Untangling and Object Search

Kaushik Shivakumar

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

DocuSigned by:
[illegible]
32B42AC2D1E3498...

Professor Ken Goldberg
Research Advisor

5/10/2023

(Date)

★ ★ ★ ★ ★ ★ ★

DocuSigned by:
Joseph Gonzalez
6B2E0BF2F5F9486...

Professor Joseph E. Gonzalez
Second Reader

5/11/2023

(Date)

Probabilistic State Estimation to Enable Manipulation and Interactive Perception for Robotic Cable Untangling and Object Search

Abstract

Probabilistic State Estimation to Enable Manipulation and Interactive Perception for Robotic Cable Untangling and Object Search

by

Kaushik Shivakumar

Master of Science in Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Ken Goldberg, Chair

Probabilistic state estimation is crucial in robotics when full state reconstruction is not possible due to partial observability. Outputting distributions over the state space allows for expression of uncertainty in a useful way for a downstream planner, which can interact with the scene to increase confidence via a method called interactive perception and eventually make task progress. We investigate probabilistic state estimation and interactive perception for cable untangling and object search in semantically organized shelves. First, we introduce Tracing to Untangle Semi-planar Knots (TUSK), a learned cable tracing algorithm that resolves overcrossings and undercrossings to recognize knot structure and grasp points for untangling from a single RGB image. This work focuses on semi-planar knots, containing crossings each with at most 2 cable segments. We conduct experiments on 3-meter cables with up to 15 semi-planar crossings across 6 different knot types. We find that in scenes with multiple identical cables, TUSK can trace a single cable with 81% accuracy on 7 new knot types. In single-cable images, TUSK can trace and identify the correct knot with 77% success on 3 new knot types. We incorporate TUSK into a bimanual robot untangling system and find it successfully untangles 64% of cable configurations, including those with new knots unseen during training, across 3 levels of difficulty. Second, we introduce Semantic Spatial Search on Shelves (S^4) to improve efficiency when locating a fully occluded target object in a shelf. Shelves in pharmacies, restaurant kitchens, and grocery stores are often organized such that semantically similar objects are placed close to one another. With Semantic Spatial Search on Shelves (S^4), we use large language models (LLMs) to generate affinity matrices, where entries correspond to semantic likelihood of physical proximity between objects. We derive occupancy distributions by synthesizing semantics with learned spatial constraints. Simulation experiments suggest that S^4 combined with an interactive perception policy reduces search time relative to pure spatial search by an 24% across three domains: pharmacy, kitchen, and office shelves, and physical experiments in a pharmacy shelf suggest 47.1% improvement. We conclude with limitations and areas for future work.

To my parents, grandparents, sister, and incoming puppy.

Contents

List of Figures

List of Tables

Acknowledgments

Working on research at UC Berkeley has been an extremely enriching experience for me. I'd like to thank Professor Joe Hellerstein for accepting me into his group at RISE lab as a freshman and David Chu for mentoring me during my first three years at Berkeley when I was working on distributed systems research on automatic protocol optimization. Professor Hellerstein was extremely helpful when I expressed interest in robotics research, recommending AUTOLab to me.

I want to thank Professor Goldberg for welcoming me into his research group, providing me the opportunity even as an undergraduate to co-lead a project, and putting me in a position to learn and collaborate with experts in the field. It is difficult to even express how much I've learned since joining AUTOLab; I went from a complete amateur to someone who feels confident reading most papers from the deep learning and robotics community. Obviously, I'm no expert and have a long way to go, but thank you to Professor Goldberg and the AUTOLab community for getting me started on this journey.

Specifically, I'd like to thank my collaborators, without whom none of the work I've done would be possible. I'd like to thank Justin Kerr, Raven Huang, Ryan Hoque, Brijen Thananjeyan, Ashwin Balakrishna, Ellen Novoseller, Jeffrey Ichnowski, and Yahav Avigal for providing me with invaluable guidance and mentorship as Ph.D. students and Postdocs.

I also want to acknowledge the other undergraduates and MS students I've had the opportunity to collaborate with: Vainavi Viswanath, Satvik Sharma, Mallika Parulekar, Jainil Ajmera, Anrui Gu, Shrey Aeron, Gabriel Deza, and Aditya Ganapathi. All of you are always so full of great ideas, and I've learned so much from working with you.

I'm of course extremely grateful to my primary collaborator for the past few years, Vainavi Viswanath. Thank you for welcoming me into the cable untangling project, teaching me, and inspiring me with your dedication, organization, and work ethic in pursuit of research goals. I've learned so much from you, and your energetic and bubbly presence in the lab never fails to make me smile.

None of the work I'm presenting in the thesis is entirely my own; rather, it is a result of long-standing collaborations, cross-fertilization of ideas, and countless hours of hard work from everyone I've mentioned. Specifically, for the first half on TUSK, I want especially to acknowledge Vainavi's hard work developing the tracer and grasp point selection methods, and Jainil and Mallika's work on ideating on and implementing analytic knot detection techniques. For the second half on S^4, I want to thank Satvik for his development of affinity value extraction and normalization, persistence during physical experiments, and numerous ideas like those on refining object detections using semantics, and Raven for her expertise from prior mechanical search projects.

I want to thank my friends and roommates at Berkeley for making my college experience incredibly enjoyable, and all the teachers and professors who've bestowed me with much of the knowledge I have today.

Finally, I have infinite gratitude for my parents, who always provided me every opportunity to learn and grow and supported me unconditionally through my education, my sister who never misses an opportunity to tease me, and my grandparents, who have looked after me and taken interest in my work ever since I was very young.

Chapter 1

Introduction

1.1 Overview

State Estimation

State estimation is a challenging but crucial task in robotics. In its most general form, the problem boils down to estimating the state $s_t \in \mathcal{S}$ of a system based on a sequence of observations $o_1, o_2, ...o_t \in \mathcal{O}$. While this is a trivial problem if the observation o_t contains the state s_t, full state estimation can become challenging or even impossible in the presence of sensor noise, occlusions, and partial observability. Given these challenges, instead of producing a single state estimate, outputting a probability distribution over states $p(s_t|o_1...o_t)$ is often sufficient to enable downstream planning and control in robotics. This thesis focuses on providing probabilistic state estimates using just a single observation for two problem settings to enable manipulation policies and interactive perception.

Active and Interactive Perception

In 1984, Goldberg and Bajcsy [29] explored active perception of shape using a robot to actively move a touch sensor to trace object contours. In 1988, Bajcsy [9] 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 [10] and moving a tactile sensor according to haptic input [29]. Recently, Bohg et al. [13] 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 [13, 60]. Within robotic manipulation, several works have focused on improving understanding of the environment through scene interaction. Tsikos and Bajcsy [79] propose interacting with random heaps of unknown objects through pick and push actions for scene segmentation. Danielczuk et al. [24] 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. [60] 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. [86] interact with a pile of laundry to isolate and identify individual clothing items.

Outline

In this thesis, we demonstrate methods to enable the application of learned state estimation and interactive perception to the following problems:

1. **Cable Untangling:** We study the problem of estimating the trace of a cable from an overhead RGB image to enable downstream tasks such as cable topology estimation, knot detection, and untangling. In this work, we introduce a novel algorithm called Tracing to Untangle Semi-planar Knots, building on our prior keypoint-based work on long cable untangling [70, 82].

2. **Semantic Spatial Search on Shelves:** We study the problem of searching for objects in a shelf, in the presence of significant occlusions. This operates via a combination of a learned spatial probability distribution combined with a semantic distribution extracted from large language models (LLMs). These two distributions can be seen as probabilistic state estimates over the state of the target object, which are used for the downstream task of object search.

1.2 Cable Untangling

In industrial and household settings, tangled long cables can pose a threat to the safety of individuals, especially older or at-risk adults, by impeding their movement. Additionally, in environments where heavy machinery is operated, cables can get caught in moving parts and cause potential damage or harm [65, 55, 80, 88].

Untangling long cables can be difficult due to challenges in manipulation and perception alike. Developing manipulation primitives that can adapt to different knot topologies is non-trivial since knot dynamics are challenging to predict and can depend on unobservable parameters of the cable like stiffness. Also, estimating cable state from an RGB image is difficult since long cables often fall into complex configurations with many crossings. Long cables can also contain a significant amount of free cable (referred to as *slack*), which can occlude and inhibit the perception of true knots from an overhead image. The task of autonomously untangling cables requires a generalizable system that can track a cable path in complex configurations and handle the wide distribution of knots present in long cables.

Much of prior work bypasses full state estimation by employing object detection networks and keypoint selection networks to identify knots and grasp points directly based on geometric patterns [82, 70]. These works can disentangle 2 types of knots (overhand and

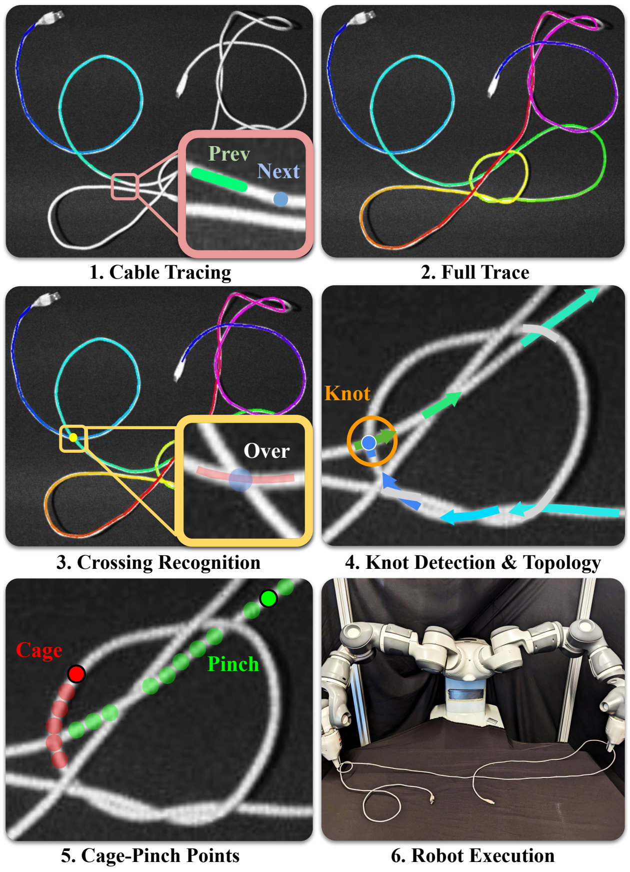

Figure 1.1: **TUSK:** TUSK first performs cable tracing (1, 2). The trace is shown through a rainbow gradient (from violet to purple), depicting the sequence in which the cable is traced. After tracing, TUSK does crossing recognition (3) to obtain the full topology of the cable. Next, using crossing 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).

figure-8), but the methods do not generalize well to the large number of complex configurations long cables can form. Other prior work is able to achieve some generality for knot disentanglement, but only for short cables up to 15 cm in length. In such short cables, the knot configurations are less complex, and there is little difficulty with slack management, eliminating the need to estimate the cable path [83, 30, 76, 74]. This work considers long cables up to 3 meters in length consisting of semi-planar knots, i.e. knots comprised of semi-planar crossings, where each crossing consists of at most 2 cable segments when viewed from above. The single-cable semi-planar knots considered in this work 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.

This work focuses on high-accuracy state estimation techniques and algorithms grounded in knot theory to process the results of state estimation and select untangling actions for a broader class of knots.

We presents TUSK, which makes the following contributions:

1. A novel cable state estimator consisting of a learning-based iterative tracer and a crossing classifier with a crossing correction algorithm.
2. An analytic knot detection algorithm and untangling point selection algorithm given the cable state estimates.
3. An untangling point selection algorithm that determines optimal graspable points for executing untangling action primitives.
4. A cable untangling system using TUSK and incorporating interactive perception.
5. Data from physical experiments using TUSK to trace and untangle configurations with semi-planar knots. Results suggest TUSK can correctly trace and segment a single cable in multi-cable settings with 81% accuracy, detect knots with 77% accuracy, and function in a physical system for untangling semi-planar knots with 64% untangling success in under 8 minutes.

1.3 Object Search

Robotic mechanical search [23] aims to retrieve a desired target object from a scene where the target object is fully occluded. This problem is especially challenging in shelves due to limited camera views and robot access. However, these environments are often organized semantically for ease of retrieval. For instance, pain relief medications such as Tylenol and Ibuprofen are often stored near each other. Can the semantic structure of an environment guide mechanical search by informing the robot about what areas are likely to contain the target object? Prior work in mechanical search either fails to consider semantic information [35, 34, 36] or uses manually constructed semantic similarities [45].

We propose using large language models (LLMs) [25, 4] as semantic knowledge bases for guiding robotic mechanical search. Since LLMs are trained on large corpora of human language, we hypothesize that they effectively encode the semantics of both common and rare objects. In contrast to recent systems like SayCan [7], we apply LLMs offline to facilitate searching and manipulating cluttered shelf environments where many objects are fully occluded. We generate an *occupancy distribution*, a probability distribution over the potential locations of the target object. We develop a novel method to estimate a *semantic* occupancy distribution by precomputing an affinity matrix from an LLM. The semantic affinity matrix consists of many rows, where values in row i encode physical proximity of every object to object i. We combine the semantic occupancy distribution with a learned

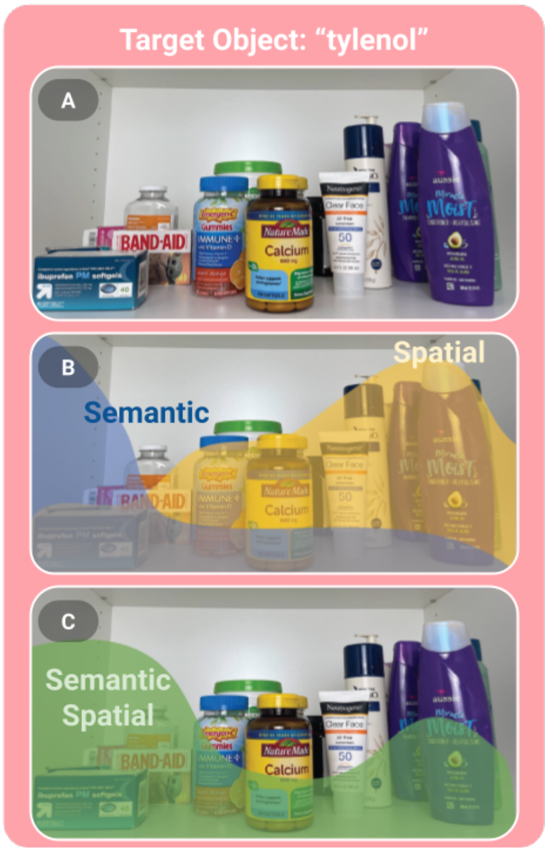

Figure 1.2: (A) Consider a robot search for a target object such as Tylenol in a semantically arranged environment such as a pharmacy shelf. (B) Semantic Spatial Search on Shelves (S^4) computes a spatial distribution and a semantic distribution (see shaded areas) over where the target object could be. The spatial distribution is based on object shapes while the semantic distribution encodes likelihood of physical proximity (e.g., Tylenol is likely closest to Ibuprofen). (C) These are combined into a single semantic spatial distribution.

spatial occupancy distribution which encodes constraints due to object geometry and camera perspective, shown to be helpful in prior work [34, 36] (Figure 1.2). We then use the resulting semantic spatial distribution to efficiently search in the target domain.

Since generating a semantic distribution requires objects in the shelf to be identified correctly, we use object detection and segmentation to identify objects in the shelf. To improve performance, we refine the object detection class probabilities to more accurately identify heavily occluded objects and distinguish between visually similar objects (e.g., shampoo and conditioner), in two ways: 1) by extracting brand names or labels from the images through Optical Character Recognition (OCR), and 2) by using clues from nearby, more confident object detections to refine detections. Hence, we apply LLMs to improving object detection without any additional annotated data, which is often required to deploy object detection models in robotics. We find that OCR with an off-the-shelf text embedding model combined with context-based semantic refinement with LLMs significantly improves object detection.

Existing product taxonomies such as the open-source Google Product Taxonomy [16] provide semantic context for objects, but they have a limited vocabulary and mostly consist of categories (e.g., "supplements") rather than specific products (e.g., "Omega-3"). In this work, we evaluate our method in the pharmacy, kitchen, and office domains indexed in the Google Product Taxonomy, which we treat as ground truth for generating semantically organized scenes and evaluating affinity matrices.

This work makes the following contributions:

1. A novel approach for synthesizing semantic affinity matrices from offline LLM queries.

2. A systematic comparison of BERT [25], CLIP [63], OpenAI Embeddings [2], OPT-13B [5], and PaLM with the Google Product Taxonomy for inferring the semantic similarity of pharmacy products.

3. Semantic Spatial Search on Shelves (S^4), a novel algorithm that combines object semantics with geometric constraints to guide mechanical search using **interactive perception**.

4. A method for refining the predictions of an off-the-shelf object detection model using Optical Character Recognition (OCR) and context using LLMs.

5. Simulation and physical experiments evaluating S^4 with and without semantics, OCR, and semantic arrangement of shelves. S^4 outperforms prior work by 24% in the simulated pharmacy, kitchen, and office domains and 47.1% in the physical experiments for the pharmacy domain.

Chapter 2

Cable Untangling: Background

2.1 Related Work

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 [30, 76, 83, 51], smoothing and folding fabric [67, 85, 28, 33, 44, 78, 32], and placing objects into bags [68, 17].

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 [27], which have been applied to cable knot tying [75] and fabric smoothing [28], as well as visual dynamics models for non-knotted cables [89, 84] and fabric [33, 89, 49]. Model-free approaches include reinforcement or self-supervised learning for fabric smoothing and folding [54, 87, 46, 8] and straightening curved ropes [87], or directly imitating human actions [67].

Cable Perception, Manipulation, and Untangling

Pioneering research in untangling, such as that conducted by Lui and Saxena [50], 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 [58], investigate iteratively refining dynamic actions [20], or use approximate state dynamics along with a learned error function [56]. Fusing point clouds across time has shown success in tracking segments of cable provided they are not tangled on themselves [66, 77].

Studies on dense knots, such as those by [30] and [76], employ learning-based keypoint detection to parameterize action primitives for untangling isolated knots. The work of [82] expands on this idea to long (3m) cables, with the addition of a learned knot detection pipeline. This approach works well for a certain range of knot types within the model's training distribution as it uses end-to-end perception systems trained on human labels. Scaling these methods to arbitrary knot types would require an intractable amount of human labels, motivating the local topology estimation approach in this work.

Prior cable state estimation work includes that led by [37], 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 [42], which models cables as chains of jointed cylindrical bodies, then uses the model for routing tasks. The works of [39] and [61] 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 [62] but does not fully estimate cable state, only identifying crossings.

2.2 Problem Statement

The objective is to bring a long (3 m) cable containing semi-planar knots into an untangled configuration, where no knots remain (knots defined in Section 2.2).

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 [30], 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 [64]. 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 I, II (both shown in Figure 2.1), 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.

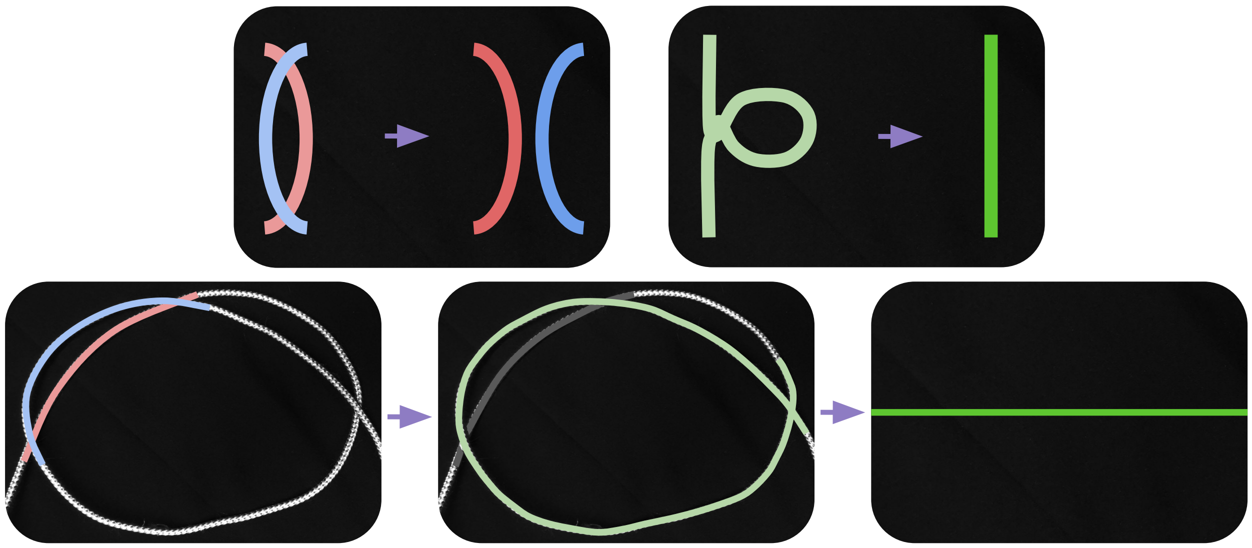

Figure 2.1: **Reidemeister Moves and Crossing Cancellation**: Top left depicts Reidemeister Move II. Top right depicts Reidemeister Move I. The bottom row shows that by algorithmically applying Reidemeister Moves II and I, we can cancel trivial loops, even if they visually appear as knots.

Chapter 3

TUSK: Tracing to Untangle Semi-Planar Knots

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 3.5).

3.1 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 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.

More formally, representing the RGB image as I, and the spline in the image s_{tot} as a sequence of pixels $s_0, s_1, ...s_n$, we break the probability distribution over splines conditioned on the image into smaller, tractable pieces using the chain rule of probability, where f_θ represents our learned neural network and $\text{crop}(I, p)$ represents a crop of image I centered at pixel p.

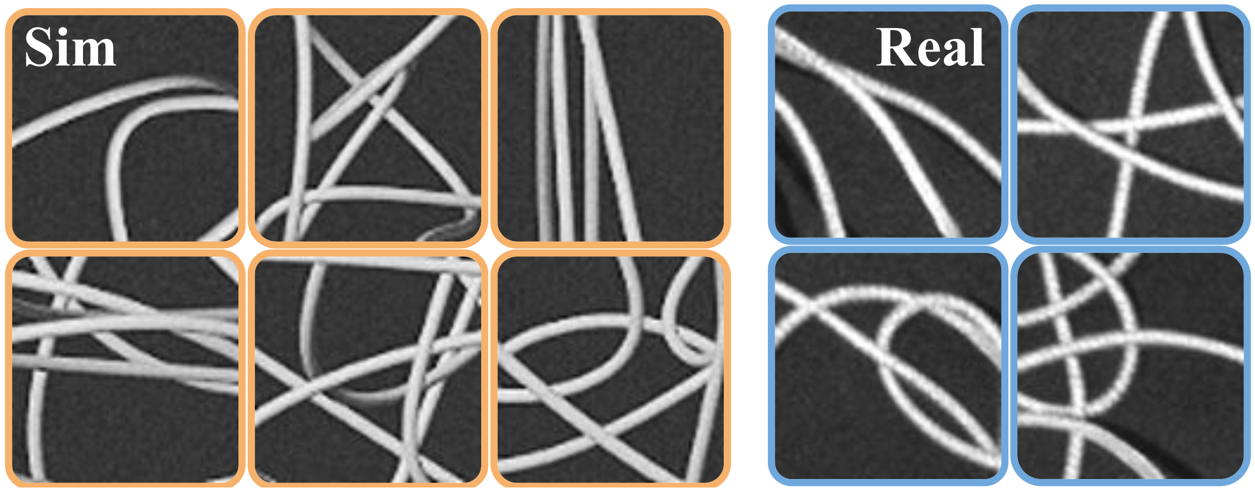

Figure 3.1: **Simulated and real crops used for training TUSK**: On the left are simulated cable crops augmented with Gaussian noise, brightness, and sharpening to match the real images (right) as closely as possible.

$$P(s_{\text{tot}}|\text{s}_0, \text{I}) = \prod_{i=1}^{n} P(s_i|s_0...s_{i-1}, I) \approx \prod_{i=1}^{n} P(s_i|s_0...s_{i-1}, \text{crop}(I, s_{i-1}))$$

The favorable properties of the autoregressive method are that it allows calculating the probability of a given trace, and also allows easy sampling from the posterior.

Initialization

To initialize the trace, we supply a start pixel along the cable (in practice, one endpoint). We use an analytic tracer as in [70] 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 [38]. Section 3.1 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 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 [21], we collect a dataset of 30,000 simulated grayscale images, whose visual appearance closely matches real observations (Fig. 3.1). 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 [43] with pixelwise binary cross-entropy loss, using a batch size of 64 and learning rate of 10^{-5}.

3.2 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.

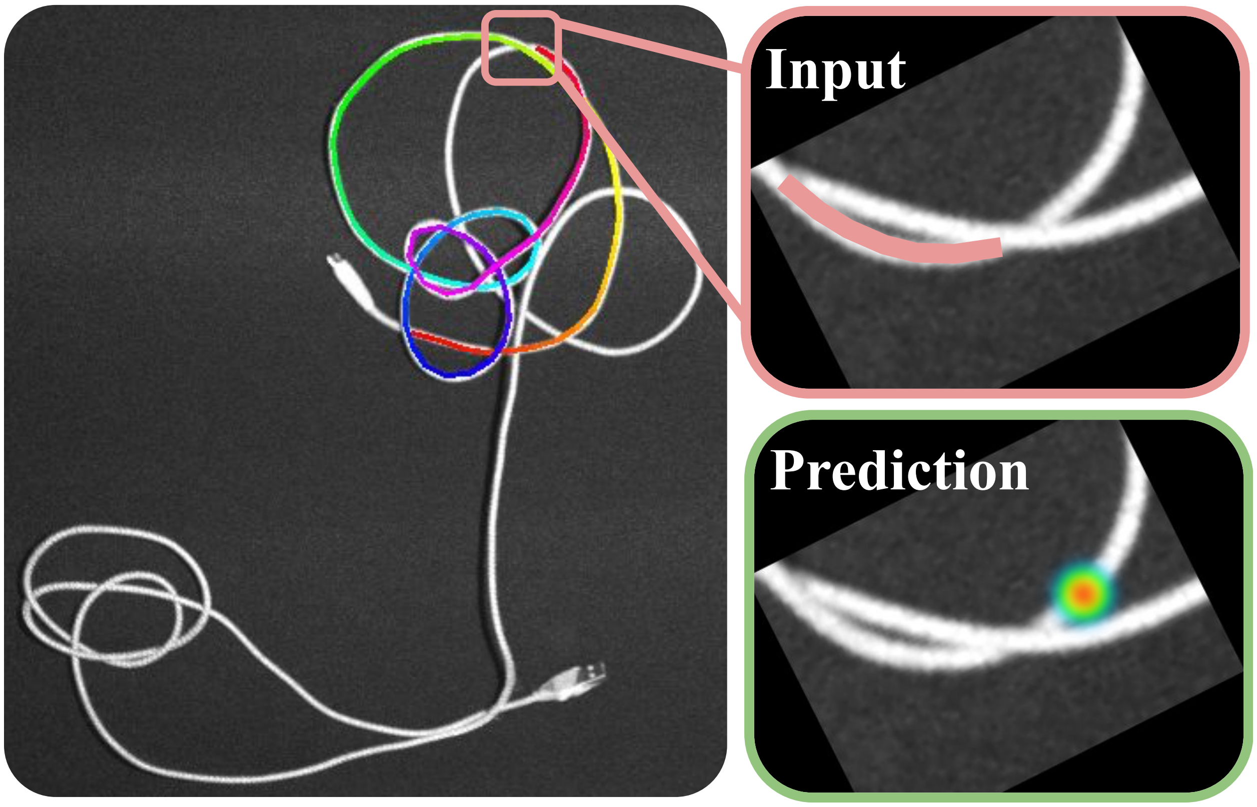

Figure 3.2: **Input and Prediction of Iterative Learned Cable Tracer**: the iterative learned cable tracer takes small crops around the cable and one step at a time, predicts the next point in the trace. The input crop is a 64x64x3 crop centered on the previous trace point. The first channel contains 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.

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.

3.3 Analytic Knot Detection

We construct line segments between consecutive points on the trace outputted by the learned cable tracer (Section 3.1). Crossings are located at the points of intersection of these line segments. We use the crossing classifier (Section 3.2) 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 3.3.

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. 2.1). 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 [64].

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 2.1. 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 2.2): 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.

3.4 Algorithmic Cage-Pinch Point Detection

As per the definition introduced in Section 3.3, 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 1.2.

3.5 Robot Untangling using TUSK

Manipulation Primitives

We use the same primitives as in SGTM 2.0 (Sliding and Grasping for Tangle Manipulation 2.0) [70] to implement TUSK as shown in Figure 3.3 for untangling long cables. We add a *perturbation* move.

Cage-Pinch Dilation

We use cage-pinch grippers as in [82]. 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 [70] separates the grippers to a small, fixed distance of 5 cm.

Reveal Moves

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

Perturbation Move

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

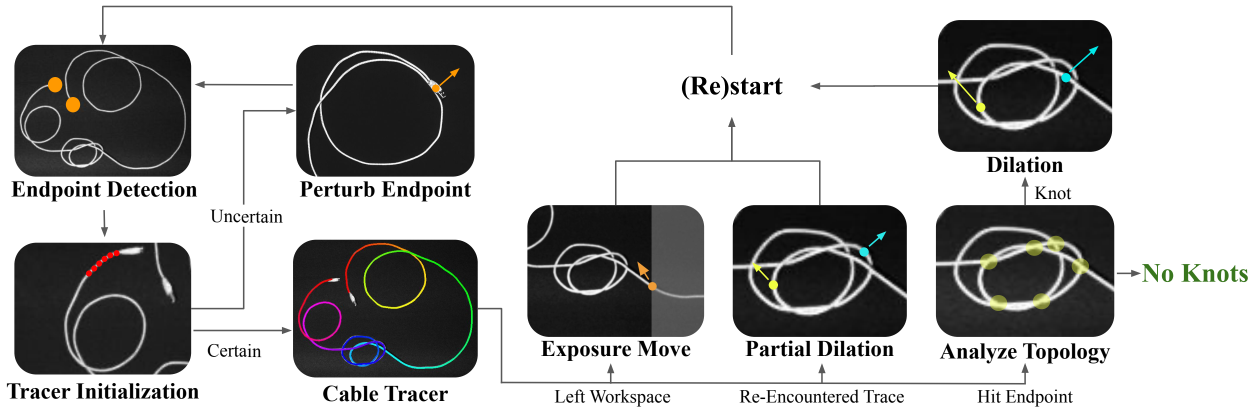

Figure 3.3: **Untangling Algorithm with TUSK**: We first detect the endpoints and initialize the tracer with start points. If we are not able to obtain start points, we perturb the endpoint and try again. Next, we trace. While tracing, if the cable exits the workspace, we pull the cable towards the center 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 3.5, 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 [70]. 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 3.3.

Importantly, this algorithm makes use of similar **interactive perception** components to SGTM 2.0; because we can tell whether the trace is successful or confident, we can take disambiguation actions as necessary to increase the likelihood of successful traces in the future (for example, the partial cage-pinch dilation if a ”retrace” is encountered). Certainly, there is more to explore with regards to extracting probability distributions from the tracer, and that is left to future work. Please refer to SGTM 2.0 [70] for a more extensive look at interactive perception.

Chapter 4

TUSK Results

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

4.1 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.

4.2 TUSK Setup

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

We test TUSK on 3 different categories of cable configurations, shown in Figure 4.1. 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.

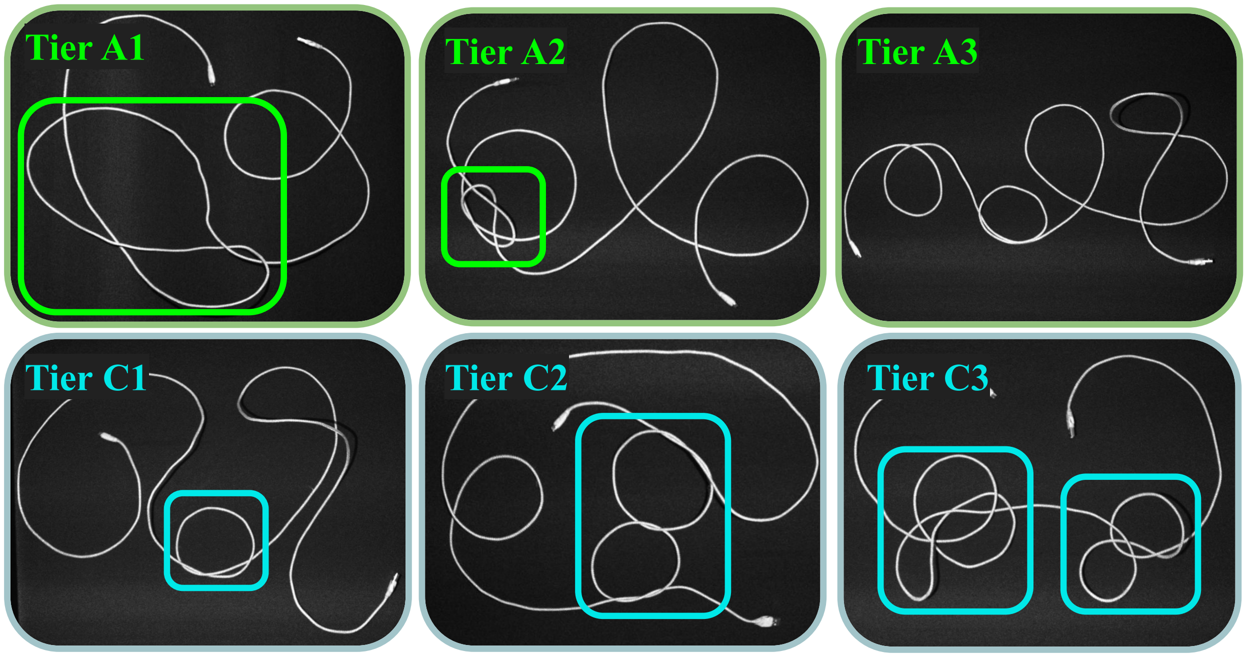

Figure 4.1: **Starting configurations for** the 3 categories for TUSK experiments and the 3 levels for physical experiments.

2. TUSK (-LT): replacing the Learned Tracer with the same analytic tracer from [70] as described in Section 4.2 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

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 4.2. We evaluate perception on multi-cable settings on 3 tiers of difficulty.

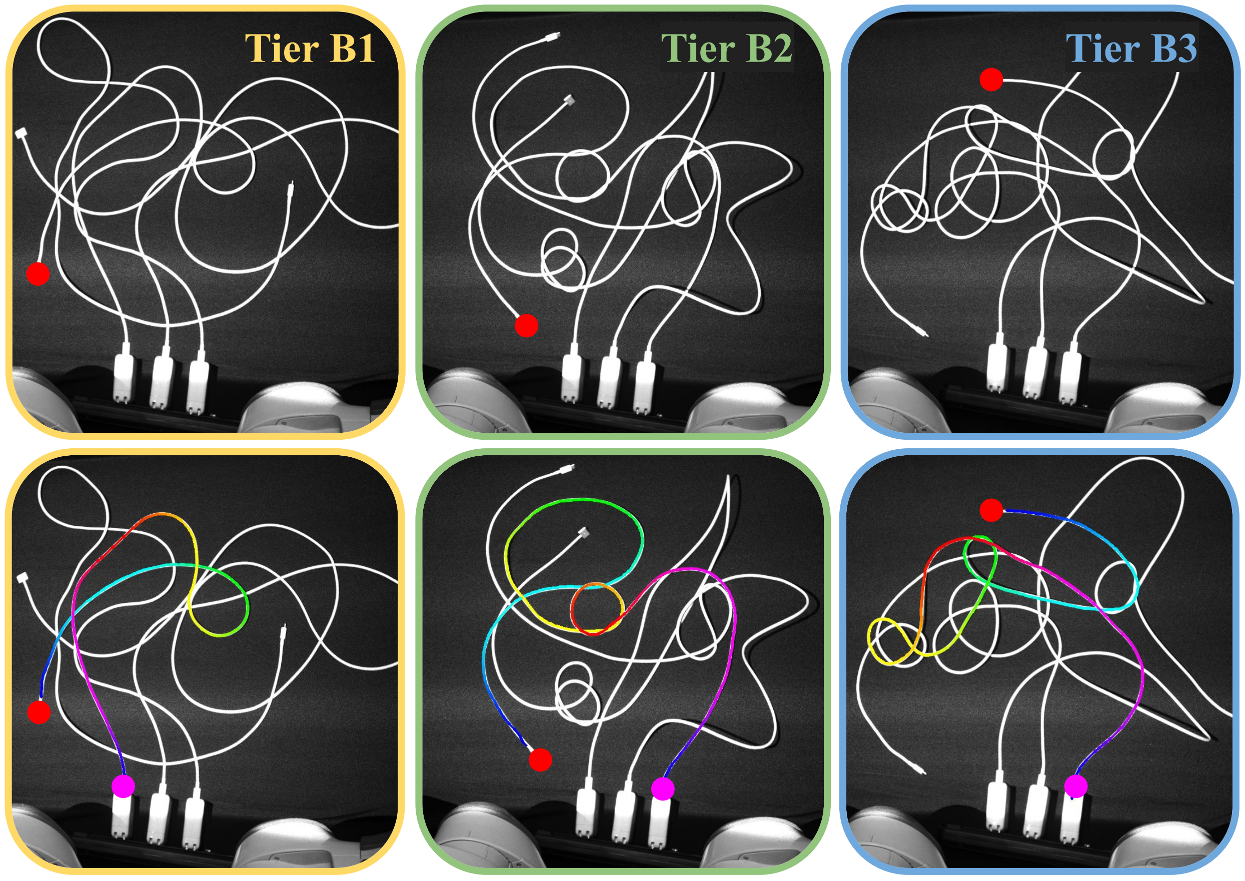

Figure 4.2: **Multi-cable tracing**: **Top row:** illustrative examples of each of the 3 tiers of difficulty for multi-cable tracing experiments. **Bottom row:** Corresponding successful traces outputted by the learned tracer.

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 [70] as a baseline with scoring rules inspired by the work of [51] and [42]. The analytic tracer explores all potential paths and determines the most correct trace through a scoring metric from [70]. 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 4.1), where all knots are upward of 10 cm in diameter, and compare performance to SGTM 2.0 [70]:

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 [70], 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.

4.3 Results

TUSK

As summarized in Table 4.1, 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.

The following are the *failure modes:*

Table 4.1: 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 4.2: 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	(2) 2, (4) 1 (5) 1	(1) 3, (2) 3 (5) 3, (6) 2, (7) 2	(1) 2, (2) 3 ((3) 1, (6) 3

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

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

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

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

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

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

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

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

For SGTM 2.0, the most common failure modes are (A) and (B), where it misses knots or incorrectly identifies knots when they are out of distribution. For 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

Table 4.3: 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

(-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 4.3 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.

The following are the *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 4.2 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.

The following are the *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 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.

Chapter 5

Semantic Spatial Search: Background

In the prior chapters, we introduced TUSK to estimate the trace of a cable in a semi-planar configuration. The following work focuses on providing a state estimate not for a single, complex, deformable object, but instead for the position of a fully occluded target object which is fully occluded at the start of a search process.

5.1 Related Work

Large Language Models

Language models (LMs) output probability distributions over sequences of natural language tokens $w_1, w_2, \ldots, w_n$. LMs typically factor the probability of the sequence with the chain rule and autoregressively perform next-token prediction: $p(w_{1..i}) = p(w_1) \cdot p(w_2|w_1) \cdots p(w_i|w_{1..i-1})$. In recent years, the Transformer neural network architecture [81] has enabled LMs to scale to "large language models" (LLMs) that train billions of parameters on terabytes of data, such as GPT-3 [6] and PaLM [1]. LLMs encode semantic context and have achieved state-of-the-art results on tasks such as machine translation, question answering, text generation, and text summarization [25, 6, 1].

Natural Language for Robotics

Prior work at the intersection of natural language processing and robotics includes language-conditioned imitation learning [71, 53, 72, 52], language-conditioned reinforcement learning [57, 40, 59], and online correction of robot policies through language feedback [69, 22]. [47] use code synthesis with LLMs to write robot control policies. Ichter et al. [3] propose SayCan, which uses a LLM for high-level planning and generating associations between language instructions and robot action primitives (e.g., "pick up the sponge"). In contrast, S^4 uses a LLM to create an occupancy distribution for mechanical search of fully occluded objects. Unlike SayCan, S^4 queries LLMs offline (before robot execution) because LLM inference is costly in terms of both time and compute [6].

Similar to our work, [15] uses language models to compute affinity scores among objects for spatial scene understanding. They propose HOLM, a system for determining where objects may be in partially observable scenes based on semantics. However, HOLM is evaluated only in simulation and considers camera adjustment actions rather than physical interactions with the environment. In contrast, we use object affinities to generate a semantic occupancy distribution, a novel way to interpret the scene and integrate geometric constraints for real-world mechanical search.

Robotic Mechanical Search

Much of prior work on mechanical search considers shelves, but not in semantically arranged environments [34, 36, 35, 18]. Prior work proposes the LAX-RAY system [35], which uses a neural network to predict a spatial "occupancy distribution" for where an occluded target object could be located at a given time, by considering object geometries and camera perspective effect (e.g., high target object can't be occluded by short object and object in the center of the camera frame occludes more areas). At each timestep, the spatial occupancy distribution is updated to be the minimum of the previous distribution and the current predicted distribution. LAX-RAY proposes Distribution Area Reduction (DAR) as a search policy that specifies the sequence of lateral pushing actions to take in order to reveal the target object as quickly as possible. DAR is a greedy strategy that first moves the object whose segmentation mask maximally overlaps with the spatial occupancy distribution to an area of minimal overlap. In this work, we extend the notion of occupancy distributions to semantics and use DAR as our mechanical search policy.

[45] consider both semantics and geometry for mechanical search. However, they manually generate semantic categories with typically 1-2 categories in the shelf, which can deviate from real world distribution. Their approach does not scale to the complexity of real world semantics and is tested only in simulation. In contrast, we harness the knowledge of large language models to extract open-vocabulary semantic information and accelerate robotic mechanical search.

Object Detection Refinement

Accurate object detection is a critical step in many downstream robotics applications, including mechanical search. We propose using OCR and LLMs in S^4 to refine object detection by considering the conditional probabilities of all the object classes given the text present on them. While OCR has been used in prior work to aid object detection [41], we use text embedding combined with OCR for better performance. S^4 also uses the semantic context of nearby objects to refine object detection. There exists prior work [11, 52, 26, 19] that considers using context from surrounding regions and objects to inform object detection, but these approaches do not use large language models as semantic knowledge bases. There are some recent object detection models such as DETR [14, 73] that are based on Transformers [81] and use full scene context to detect objects. However, these models learn these relationships from scratch, which requires an extensive amount of training data. By using off-the-shelf large language models, we can provide this context without any additional data or learning from pixels.

5.2 Problem Statement

The starting state of a scene is drawn from the space of *semantically organized* scenes, sampled proportionally to its approximate likelihood of occurrence in the real world. We base these likelihoods on the Google Product Taxonomy [16] and describe our procedure for scene generation in Section 7.2.

We consider the problem of robotic mechanical search for a target object $\mathcal{O}_T$ in a cluttered, semantically organized shelf containing the target and N other rigid objects $\{\mathcal{O}_1, ..., \mathcal{O}_N\}$ of cuboidal shapes in stable poses. We build on the problem statement and assumptions in [34]. We model the setup as a finite-horizon Partially Observable Markov Decision Process (POMDP). States $s_t \in \mathcal{S}$ consist of the full geometries and poses of the objects in the shelf at timestep t and observations $y_t \in \mathcal{Y} = \mathcal{R}^{H\times W\times 4}$ are RGBD images from a robot-mounted depth camera at timestep t. Actions $a_t \in \mathcal{A} = \mathcal{A}_p \cup \mathcal{A}_s$ are either *pushing* or *suction* actions, where the former are horizontal linear translations of an object along the shelf and the latter pick up an object with a suction gripper and translate it to an empty location on the shelf with no other objects in front of it.

We make the following assumptions:

- The dimensions of the shelf are known.
- Each dimension of each object is between size $S_{\min} = 5\,\text{cm}$ and size $S_{\max} = 25\,\text{cm}$.
- The shelf is semantically organized.
- The names of all objects in the shelf are a subset of a known list of object names.
- Actions cannot inadvertently topple objects or move multiple objects simultaneously.

The objective is to minimize the total number of actions required to reveal at least $X\%$ of the target object to the camera. A trial is successful if this threshold visibility X is reached within $H = 2N$ actions, and unsuccessful otherwise.

Chapter 6

Semantic Spatial Search on Shelves (S^4)

6.1 Affinity Matrix Generation

LLMs are best known for their ability to generate unstructured free-form text. However, in our work, we use them to provide probabilistic completions to textual prompts in order to generate an occupancy distribution encoding the semantics of any scene that may contain it along with other objects. This is possible because training data for such models contains a vast amount of information about object semantics [1, 6]. We extract this information from language models offline (i.e., before robot task execution).

We query the language model with a specific prompt: "In a shelf, the X goes next to the ____." Note that this prompt queries the LLM for the notion of physical proximity rather than pure semantic similarity. We use the scoring mode of the language models to find the probabilities of each of the object names appearing in the blank space. This creates a row of the affinity matrix corresponding to object X. We iterate over all object names to create the full matrix, M^{raw}.

M^{raw}, however, still requires additional modification. Let $\mathcal{N}(B, A)$ represent the event that B is the nearest object to A in the shelf, and let S_A denote the event that object A is in the shelf. The affinity matrix should provide the probability that B is the nearest object to A given that both are present (but not necessarily visible) in the scene:

$$P(\mathcal{N}(B, A)|S_A, S_B) = \frac{P(\mathcal{N}(B, A), S_B|S_A)}{P(S_B|S_A)} \tag{6.1}$$

The affinity matrix, however, gives $P(\mathcal{N}(B, A), S_B|S_A)$, which differs from the desired quantity by a normalizing constant. This causes the objects corresponding to words that are used less frequently in the LLM training data to be incorrectly penalized. Because we assume $S_i \perp S_j \ \forall i, j$, we remedy this by using the sum of the probabilities across each column (object B) as a proxy for the denominator to perform per-column normalization. We follow this with a per-row normalization to obtain a probability vector for each row (object A). We also set the main diagonal of the matrix to zero probability as these elements reflect the affinity of objects with themselves, which is not required for mechanical search. This leads to matrix M^{norm}.

We only need to control the hyperparamter temperature γ, meant to control the "confidence" by regulating the uniformity of the language model's predictions. For each row i,

$$M_i \propto \exp\left(\gamma^{-1} \log M_i^{\text{norm}}\right).$$

This procedure leads to our ready-to-use affinity matrix M.

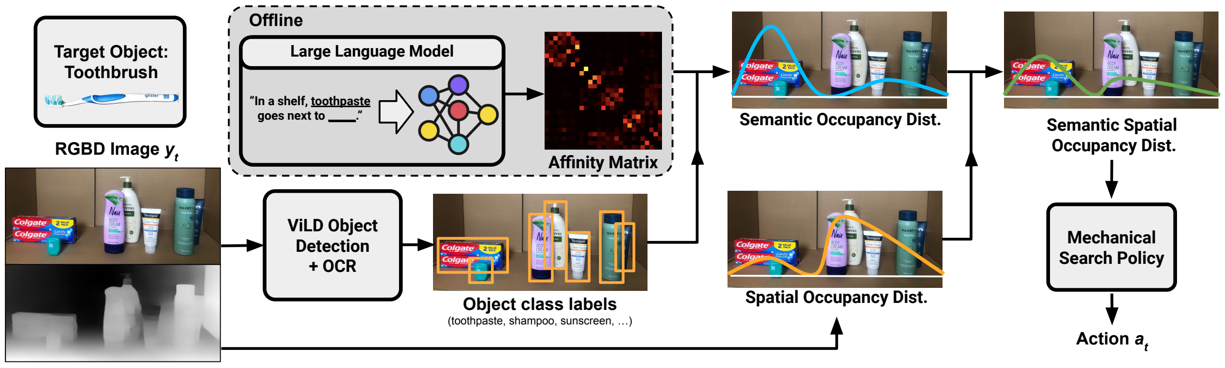
Target Object: Toothbrush
RGBD Image y_t
Offline
Large Language Model
"In a shelf, toothpaste goes next to ____."
Affinity Matrix
ViLD Object Detection + OCR
Object class labels
(toothpaste, shampoo, sunscreen, ...)
Semantic Occupancy Dist.
Spatial Occupancy Dist.
Semantic Spatial Occupancy Dist.
Mechanical Search Policy
Action a_t

Figure 6.1: System overview of Semantic Spatial Search on Shelves (S^4). The affinity matrix is computed offline. Given an RGBD image, we use object detection combined with refinement to query the affinity matrix and construct a semantic occupancy distribution. We multiply this by a spatial occupancy distribution to use in a mechanical search policy.

6.2 Object Detection Refinement

OCR

We use an object detection and segmentation model that returns object segmentation masks from an RGB input. We filter out all objects less than size $S_{\min}$ and greater than size $S_{\max}$, as well as objects contained fully within other objects, in order to avoid false positives. We use the general-purpose object detector ViLD [31] to detect, segment, and classify objects in our shelf.

Because ViLD is a general-purpose detector, it cannot easily distinguish between objects belonging to the same domain (e.g., Advil versus Ibuprofen). Because of this, we use OCR with Keras OCR (`https://pypi.org/project/keras-ocr/`) to improve the quality of the object detections. For each object, we concatenate the text observed on it and compute the text embedding using OpenAI Embeddings. Then, we compute the dot product between the embeddings of the concatenated text and every class label. We then normalize this probability vector by subtracting the minimum value and then adjusting the vector with some temperature. Then, we finally multiply this by the object detection probability vector.

Let C_i denote the class label of object $\mathcal{O}_i$ (e.g., "Tylenol" as opposed to the broader category "medication"); I_i represent the general shape, size, and color-related features of $\mathcal{O}_i$; and T_i be the detected text on $\mathcal{O}_i$. Recall that all objects belong to some class C_i. We calculate

$$\begin{aligned}
&P(C_i|\ I_i, T_i) \\
&= \frac{P(I_i, T_i|C_i) \cdot P(C_i)}{P(I_i, T_i)} \\
&= \frac{P(T_i|I_i, C_i) \cdot P(I_i|C_i) \cdot P(C_i)}{P(I_i, T_i)} \\
&= \frac{P(T_i|C_i) \cdot P(I_i|C_i) \cdot P(C_i)}{P(I_i, T_i)} \\
&= \frac{P(C_i|T_i)P(T_i)}{P(C_i)} \cdot \frac{P(C_i|I_i)P(I_i)}{P(C_i)} \cdot \frac{P(C_i)}{P(I_i, T_i)} \\
&\propto P(C_i|T_i)P(C_i|I_i)
\end{aligned}$$

as T_i is independent of I_i when conditioned on C_i, and $P(C_i)$ is uniform. This illustrates that the multiplication of the OCR probabilities and the object detection probabilities can give us a refined estimate of the category probabilities.

Semantics

We further refine the object detection probabilities by using other objects in the scene which we are more confident about. We sort the object bounding boxes from most to least confident using the Shannon entropy of each distribution over object labels after OCR refinement. We then threshold the entropy values on a confidence threshold c to determine which object bounding boxes to refine. We iterate over these high-entropy bounding boxes in increasing order of entropy and refine the distribution for the i-th bounding box using the following equation:

$$P(i)' = P(i) \cdot \frac{\sum_{j=0}^{i-1} d(i,j) \cdot M_j}{\sum_{j=0}^{i-1} d(i,j)}$$

Here, $P(i)$ is the existing class distribution for the i-th bounding box, $d(\cdot)$ is an exponentially decaying kernel function based on distance between any two objects in the scene, and M_j is the row of the affinity matrix corresponding to the label of the j-th object. Thus, we update probabilities to be a weighted average of the affinities of known objects, weighted by the distance to the uncertain object. The kernel function we use is of the form $d(x) = e^{-ax}$, where x is the pixel distance between the centers of the objects and a is a hyperparameter that depends on the dimensions of the image to normalize the distance metric across different resolutions.

6.3 Semantic Occupancy Distribution

Here we describe the process of converting affinities and object detections into a semantic probability distribution over possible locations of the target object.

Tracking the Original Scene

To build the semantic distribution, the system requires knowledge of object locations in the original scene s_0. Once mechanical search begins, the shelf may become semantically disorganized. One technique to avoid this issue is to freeze the semantic distribution at $t = 0$, but this prevents adding information about initially occluded objects. To remedy this, we keep track of the location of every object the first time it is seen and add new objects that are later unveiled. At every step t in a rollout, we compute our semantic distribution on our most up-to-date understanding of the object classes and their locations in the original scene.

Calculating the Distribution

The semantic occupancy distribution models the probability that the target object occupies a given location, given the classes of observed objects in the scene. The semantic distribution takes the form

$$P(L_T = l \mid L_{1\ldots n} = l_{1\ldots n}, C_{1\ldots n} = c_{1\ldots n}),$$

where L_T is the location of the target object, $L_{1\ldots n}$ are the inferred positions of the *visible* objects ($n \leq N$, the total number of objects in the shelf), and $C_{1\ldots n}$ are the inferred classes of the visible objects (i.e., the class labels with the highest probabilities) from Section 6.2. We abbreviate this quantity as $P(L_T = l \,|L, C)$.

We interpret affinity values M_{ij} to be the probability of object j being the closest to object i in expectation across scenes. However, given the current scene, there may be more or less space that is nearest to a particular object, so we interpret these affinity values as being normalized per unit area. This means that the height of the semantic distribution is directly proportional to the affinity value of the nearest object. Formally, given that $N(l)$ is a function returning the index of the object closest to location $l = (x_l, y_l)$,

$$P(L_T = l \,|L, C) \propto M_{target,N(l)}.$$

In simulation experiments, $N(\cdot)$ is computed using the 3D coordinates of the visible objects obtained from depth image. We compute the 2D semantic occupancy distribution

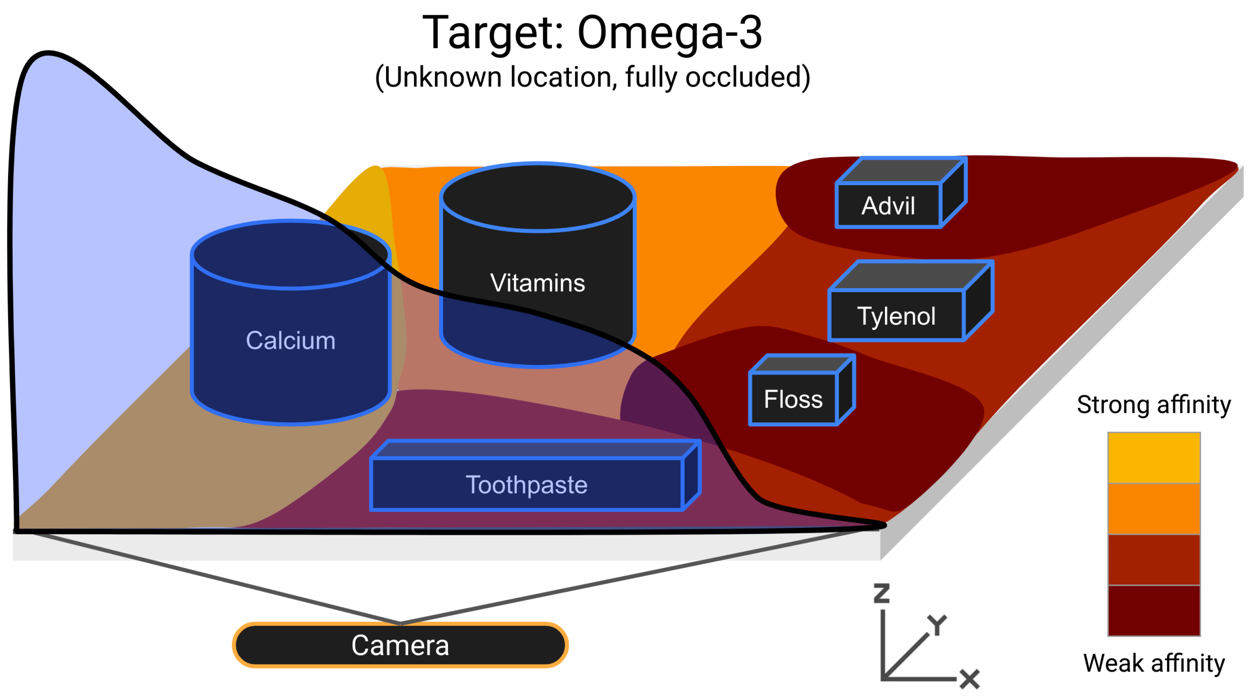

Figure 6.2: Here we have an example scene where the target is Omega-3 and the visible objects are labeled. We illustrate the process of calculating the 2D semantic distribution (xy-plane of the shelf) and then projecting to a 1D semantic distribution (shaded blue in the xz-plane), as described in Section 6.3.

(in the xy plane of the shelf) and reduce it to 1D by summing along camera rays. In physical experiments, to avoid errors due to noisy depth readings we compute the distribution directly in 2D, using pixel distance for $N(\cdot)$ instead of world coordinates. As a final post-processing step to account for noise when objects are moved or bumped around slightly, we apply smoothing using a Gaussian kernel with standard deviation σ. The process of calculating the semantic distribution is illustrated in Figure 6.2.

6.4 Semantic Spatial Search on Shelves (S^4) Algorithm

S^4 combine the semantic occupancy distribution with a spatial distribution and provide it as input to a mechanical search policy. We use the method from [34] to learn a spatial occupancy distribution from a depth image of the scene. This distribution estimates where the target object can be based on possible occlusions of the known dimensions of the target object. We multiply this spatial occupancy distribution element-wise with the semantic

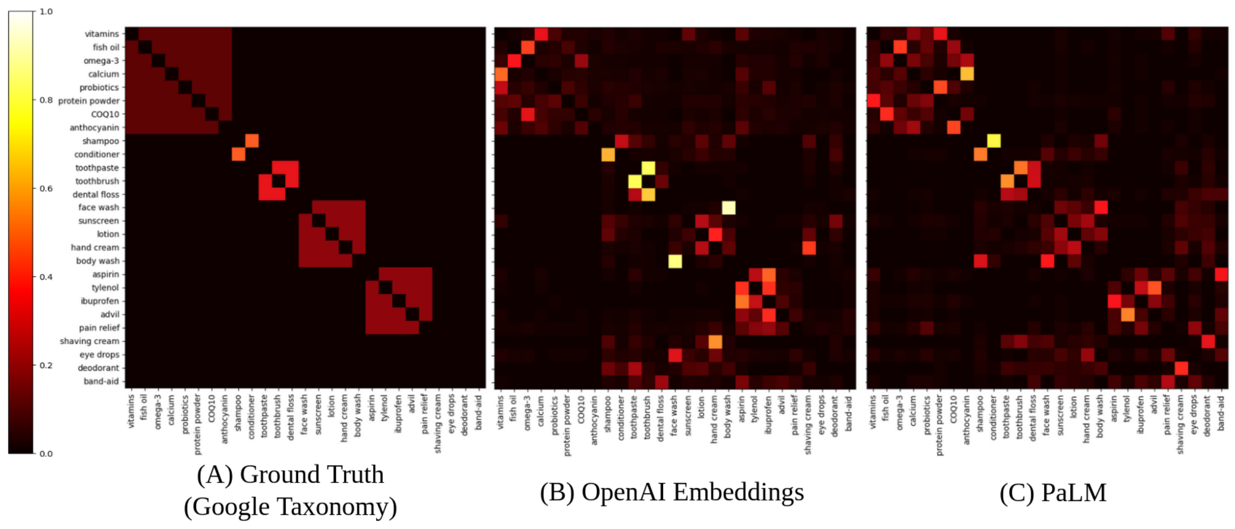

Figure 6.3: Ground truth affinity matrix and affinity matrices generated by OpenAI Embeddings and PaLM. The matrices are able to roughly capture the block diagonal structure of the ground truth matrix.

occupancy distribution from Section 6.3. This weights the spatial distribution with where the object is semantically likely to be. We then use the Distribution Area Reduction (DAR), an **interactive perception** policy from [34] with the synthesized distribution to perform mechanical search. A full overview of S^4 is shown in Figure 6.1. The use of interactive perception is absolutely critical for this problem due to the fact that the target object is occluded by at least one other object, making it necessary to build on the probabilistic state estimate that S^4 provides in the form of a semantic spatial probability distribution.

Chapter 7

S^4 Results

In this section, we evaluate (1) the quality of affinity matrices generated by different LLMs via comparison to the Google Product Taxonomy (Section 7.1), (2) ablations of key components of the object detection system such as OCR (Section 7.3), and (3) the effect of LLM-based semantics on the speed of mechanical search in simulation and physical environments (Sections 7.4 and 7.5).

7.1 Affinity Matrix

The choice of LLM affects the values in the affinity matrix. In order to compare different LLMs and quantitatively evaluate the quality of affinity matrices, we use the open-source Google Product Taxonomy [16] as the "ground truth" matrix. In the pharmacy domain, we use the following 6 categories and items from the taxonomy:

1. Supplements: vitamins, fish oil, omega-3, calcium, probiotics, protein powder, COQ10, anthocyanin
2. Hair Care: shampoo, conditioner
3. Oral Care: toothpaste, toothbrush, dental floss
4. Cosmetics: face wash, sunscreen, lotion, hand cream, body wash
5. Medication: aspirin, tylenol, ibuprofen, advil, pain relief
6. Outliers: shaving cream, eye drops, deodorant, band-aid

For the ground-truth matrix, all elements in a category are given uniform affinities to each other, and each row is normalized to sum to 1.0 probability. Note that each item in the "outliers" category (e.g., eye drops) does not belong to any of the other 5 categories and is treated as its own category. With the categories listed in order along both axes of the matrix, the ground truth affinity matrix has a block-diagonal structure with a uniform block for each category (Figure 6.3A). We evaluate the following LLMs and embedding models off-the-shelf, without finetuning: BERT [25], CLIP [63], embeddings from the OpenAI API [2], OPT-13B [5], and PaLM. For LLMs, we generate affinity matrices as described in 6.1 with $\gamma = 1$, and for embedding models we take the dot product of the embeddings of the object names and optimize the γ to minimize the Jensen-Shannon Distance (JSD) [48] to the ground truth. JSD measures the similarity between two probability distributions, so we measure the similarity between each row of the affinity matrix and the corresponding row of the ground truth. Then, we average across the rows to get the average distance from each object's probability distribution to that object's ground truth. We observe that the choice of LLM has a significant impact on the affinity matrix (Table 7.1), and that the LLMs can approximately recover the block diagonal structure of the ground truth matrix (Figure 6.3). PaLM attains the highest accuracy, with a 44.6% improvement over a uniform affinity matrix.

Table 7.1: Affinity matrix results. We report the average Jensen-Shannon Distance (JSD) between each row of the affinity matrix and the ground truth matrix, as well as the percentage improvement over the uniform JSD (i.e., (uniform JSD - method JSD) / uniform JSD).

Method	**JSD** ($\downarrow$)	**% Improvement** ($\uparrow$)
Uniform	0.65	N/A
BERT Embedding	0.64	1.5
CLIP Embedding	0.52	20.0
OpenAI Embedding	0.43	33.8
OPT-13B	0.38	41.5
PaLM	**0.36**	**44.6**

7.2 Semantic Scene Generation

We consider three shelf domains to apply our method: a pharmacy, an industrial kitchen, and an office. We select 27, 24, and 40 representative objects respectively in these domains from the Google Product Taxonomy [16]. We provide the full lists of objects in the appendix. We use the taxonomy as a ground truth reference for producing realistic semantically arranged scenes in simulation and physical experiments. Both simulation and real experiments take place in a $0.8\,\text{m} \times 0.35\,\text{m} \times 0.57\,\text{m}$ shelf.

The taxonomy defines a tree where each category is a node and each object name is a leaf node. To create a scene with N objects in a given domain, we begin by uniformly sampling N objects without replacement from the total objects available in that domain. We then generate scenes in a top-down recursive manner using the taxonomy tree. At the root, we start with the whole shelf available to us. At each node, we split the shelf in half either horizontally or vertically with 50% probability each and recursively continue scene generation in these sub-shelves. If a node has more than 8 descendants, however, we always split the scene horizontally to avoid overcrowding resulting from the aspect ratio of the shelf. At each level of recursion, we accumulate random noise to the eventual placement of each object in the current branch, uniformly sampled from -2 cm to 2 cm. At the last non-leaf node, we place all leaves in random positions within the current level's sub-shelf. We resolve collisions by iteratively moving objects along the displacement vector between colliding objects and discard scenes where such a procedure takes longer than 1 second to run. We also discard scenes where there is no potential target object that is invisible from the camera's perspective at the start of the rollout. We reiterate that the taxonomy is *independent* of the language models used to generate affinities. The LLMs are applicable beyond manual semantic categorizations like the Google Taxonomy, but we use this resource for evaluation purposes. The scenes for all simulation, physical, and object detection experiments are generated by this procedure.

We use approximate sizes of these items to generate collision-free scenes. In simulation, we also scale these objects down in order to be able to run experiments on the same-sized shelf, which has an effect similar to running experiments in a larger shelf where more items could originally fit. The scaling factors for the pharmacy and kitchen domains are 0.7, but 0.4 in the office domain due to overall larger objects unable to easily fit and move within a small shelf.

7.3 Object Detection

Table 7.2: Object Detection Refinement Results. We ablate the components of our object detection system (Section 6.2) and report the mean average precision (mAP) of the predicted bounding boxes and top-K accuracy of the predicted labels.

Method	**mAP** (↑)	**Top-K Accuracy %** (↑)		
		k=1	k=3	k=5
ViLD	2.4	14.7	32.3	41.6
ViLD + OCR	28.9	45.0	62.0	69.5
ViLD + OCR +Semantic Refinement	**30.6**	**49.9**	**67.4**	**74.6**

We test object detection performance on scenes generated through isolated perception experiments. We take RGB images of 100 scenes of the Pharmacy domain using a high-resolution camera and run three object detection methods:

1. **ViLD:** Off-the-shelf ViLD [31].

2. **ViLD + OCR:** Refinement of ViLD with OCR as described in Section 6.2.
3. **ViLD + OCR + Semantic Refinement:** The full object detection method in Section 6.2.

Results for this experiment are in Table 7.2. As is standard in the computer vision literature, we report mAP (mean Average Precision) averaged over intersection-over-union (IOU) thresholds from 0.50 to 0.95 with a step size of 0.05, as well as top-k classification accuracy (i.e., if the ground truth label appears in the k labels with the highest probabilities). The results show that OCR leads to a significant improvement across all metrics, with mAP improving by a factor of 12 and top-1 accuracy improving by a factor of 3. Adding semantic refinement further improves mAP by 1.7% and all of the top-k accuracy metrics by approximately 5%, suggesting that LLMs can provide zero-shot improvement to object detection. Such a technique is especially useful when the data and model capacity required to learn such semantic relationships is limited.

Table 7.3: Simulation Experiment Results. Please refer to the paper for detailed performance with 18 objects, which is omitted to save space.

	Pharmacy Domain						
	12 objects		**15 objects**		**18 objects**	**21 objects**	
	Successes	**# Actions**	**Successes**	**# Actions**	**Successes**	**Successes**	**# Actions**
Spatial NN	168/190	4.06 ± 0.23	160/186	5.17 ± 0.28	144/188	104/177	8.24 ± 0.67
S^4 (Embed.)	**176/190**	2.90 ± 0.18	159/186	3.77 ± 0.26	146/188	110/177	5.69 ± 0.54
S^4 (PaLM)	**176/190**	$\mathbf{2.66 \pm 0.14}$	**162/186**	$\mathbf{3.26 \pm 0.19}$	**150/188**	**118/177**	$\mathbf{5.47 \pm 0.43}$
	Kitchen Domain						
	12 objects		**15 objects**		**18 objects**	**21 objects**	
	Successes	**# Actions**	**Successes**	**# Actions**	**Successes**	**Successes**	**# Actions**
Spatial NN	185/192	2.15 ± 0.14	182/194	2.97 ± 0.23	177/193	159/191	4.36 ± 0.38
S^4 (Embed.)	**186/192**	$\mathbf{1.56 \pm 0.08}$	**188/194**	2.15 ± 0.15	**184/193**	**167/191**	$\mathbf{3.07 \pm 0.25}$
S^4 (PaLM)	184/192	1.60 ± 0.10	184/194	$\mathbf{2.04 \pm 0.13}$	179/193	163/191	3.17 ± 0.28
	Office Domain						
	12 objects		**15 objects**		**18 objects**	**21 objects**	
	Successes	**# Actions**	**Successes**	**# Actions**	**Successes**	**Successes**	**# Actions**
Spatial NN	172/194	2.60 ± 0.18	152/188	4.15 ± 0.38	136/190	115/181	5.86 ± 0.56
S^4 (Embed.)	**173/194**	3.01 ± 0.22	152/188	3.80 ± 0.31	140/190	115/181	$\mathbf{5.33 \pm 0.50}$
S^4 (PaLM)	172/194	$\mathbf{2.33 \pm 0.13}$	**161/188**	$\mathbf{3.50 \pm 0.31}$	**142/190**	**123/181**	5.50 ± 0.49

Table 7.4: Simulation Experiment: Performance in Non-Semantic (RAND) Scenes with 15 objects. "RAND" refers to randomly arranging the objects in the scene as opposed to semantically arranging them with the procedure in Section 7.2.

	Pharmacy		**Kitchen**		**Office**	
	Successes	**# Actions**	**Successes**	**# Actions**	**Successes**	**# Actions**
Spatial NN	170/190	4.65 ± 0.29	**164/189**	4.14 ± 0.30	**140/173**	4.34 ± 0.32
S^4 (Embed.)	**172/190**	5.35 ± 0.31	162/189	4.70 ± 0.31	135/173	4.74 ± 0.32
S^4 (PaLM)	169/190	$\mathbf{4.62 \pm 0.25}$	157/189	$\mathbf{3.95 \pm 0.29}$	138/173	$\mathbf{3.85 \pm 0.35}$

Table 7.5: Simulation Experiment Results in Table 7.3 averaged over number of objects, also reported with **% Reduc.**, percentage reduction in actions from Spatial NN.

	Pharmacy Domain		**Kitchen Domain**		**Office Domain**	
	Successes	**# Actions**	**Successes**	**# Actions**	**Successes**	**# Actions**
Spatial NN	576/741	5.56 ± 0.20	703/770	3.32 ± 0.14	575/753	4.14 ± 0.19
S^4 (Embed.)	591/741	4.18 ± 0.17	**725/770**	2.43 ± 0.10	580/753	4.10 ± 0.18
S^4 (PaLM)	**606/741**	$\mathbf{3.76 \pm 0.14}$	710/770	$\mathbf{2.42 \pm 0.10}$	**598/753**	$\mathbf{3.63 \pm 0.16}$

7.4 Simulation Object Retrieval Experiments

We run an extensive suite of experiments using the same simulator as prior work in mechanical search [34]. In simulation experiments, we consider three domains: a pharmacy, an industrial kitchen, and an office. We assume perfect object detection and thus do not render the physical appearance of the objects. We use a grid search on the average number of actions required in the pharmacy domain with 15 objects to tune the Gaussian smoothing σ to be 50 pixels and γ for PaLM to be 1 and for OpenAI Embeddings to be 0.004. We use the same parameters for the other two domains.

For each domain, we generate scenes with the procedure from Section 7.2 with various numbers of objects: $N = 12$, 15, 18, and 21. We generate 200 scenes for each value of N. We also test shelves that are *not* semantically organized, with all objects placed randomly (denoted as RAND). We discard scenes where the target object starts out visible, resulting in just under 200 scenes for each value of N. Termination occurs when at least $X = 1\%$ of the target object becomes visible. The reason for the low threshold is that the DAR policy has trouble making progress on a partially revealed target object [35], which may dilute the comparison between different methods for generating semantic distributions.

We test the following algorithms on each type of scene:

1. **Spatial NN:** Mechanical search with the learned spatial occupancy distribution from [35]. No semantic information is used.

2. **S^4 (Embeddings):** S^4 using the affinity matrix built with OpenAI Embeddings [2] and the same spatial distribution model as 1).

3. **S^4 (PaLM):** S^4 using the affinity matrix built with PaLM and the same spatial distribution model as 1).

We report the following metrics in simulation and physical experiments:

1. **Successes:** The ratio of trials where the target object is found (without running out of actions or hitting the maximum action limit $(2 \cdot N)$) to the total number of trials.

2. **Number of actions:** The mean and standard error of the number of actions required to reveal the target object.

We report results for all numbers of objects N in Table 7.3, semantically arranged versus randomly arranged scenes in Table 7.4, and the results averaged across all values of N in Table 7.5. In the pharmacy domain, S^4 (PaLM) outperforms both S^4 (Embeddings), while also beating Spatial NN across various values of N in terms of success rate (by an additional 30/741 scenes) and average number of actions required (by 32.4%). A point of note is that the action differential percentage grows as the number of objects increases. At 21 objects, Spatial NN requires 8.24 actions on average, whereas S^4 requires just 5.47. This trend agrees with intuition that it is unscalable to search large environments with no semantic intuition.

In the kitchen domain, S^4 (Embeddings) and S^4 (PaLM) perform similarly to each other but both outperform Spatial NN significantly once again, improving the success rate and improving the number of actions required by approximately 27%. We hypothesize that the embeddings here are better able to approximate physical proximity between kitchen objects as the categories are distinct enough such that nearby objects in the embedding vector space would also be found closer together in simulated shelves.

For the office experiments, we see that S^4 (PaLM) consistently has the highest success rate starting at $N = 15$ and higher. Overall, it achieves a 12% reduction in the number of actions and increases the success rate by 23/753 scenes. We hypothesize that this lower improvement is due to a majority of the office environment consisting of generic office supplies which do not have a clear semantic categorization, making semantic search less effective. S^4 (Embeddings) is not effective in reducing the number of actions required in this setting. We believe that this is because embeddings are not able to capture subtle differences between categories as well as the LLM is.

In Table 7.4, we observe that when scenes are *not* semantically arranged, S^4 (PaLM) has comparable performance to Spatial NN in all three domains despite having a slightly lower success rate. However, the performance of S^4 (Embeddings) degrades more significantly. Ultimately, results with S^4 (PaLM) indicate that semantic spatial search can perform as well as pure spatial search as long the scene is not adversarial (e.g., the target object is located at a very unlikely location in an otherwise semantically arranged shelf).

Overall, the results suggest that S^4 (PaLM) can accelerate mechanical search compared to the spatial distribution in semantically arranged environments by 32.4%, 27.1%, and 12.3% in the pharmacy, kitchen, and office domains respectively, while improving success rates.

7.5 Physical Object Retrieval Experiments

We use the Kinova Gen2 robot fitted with a 3D-printed "bluction" (blade and suction) tool, as in [34]. We use an Intel RealSense depth camera mounted on the tool to provide RGB and depth observations. For physical experiments, we focus on the pharmacy domain. For these experiments, we use 3 scenes each of $N = 7, 8, 9,$ and 10 objects for a total of 12 scenes and a threshold visibility of $X = 50\%$. See Figure 7.1 for the physical setup.

Because the RealSense camera is not able to capture the fine details of the text on the objects when observing the entire scene at resolution 640×480 pixels, we perform a three-

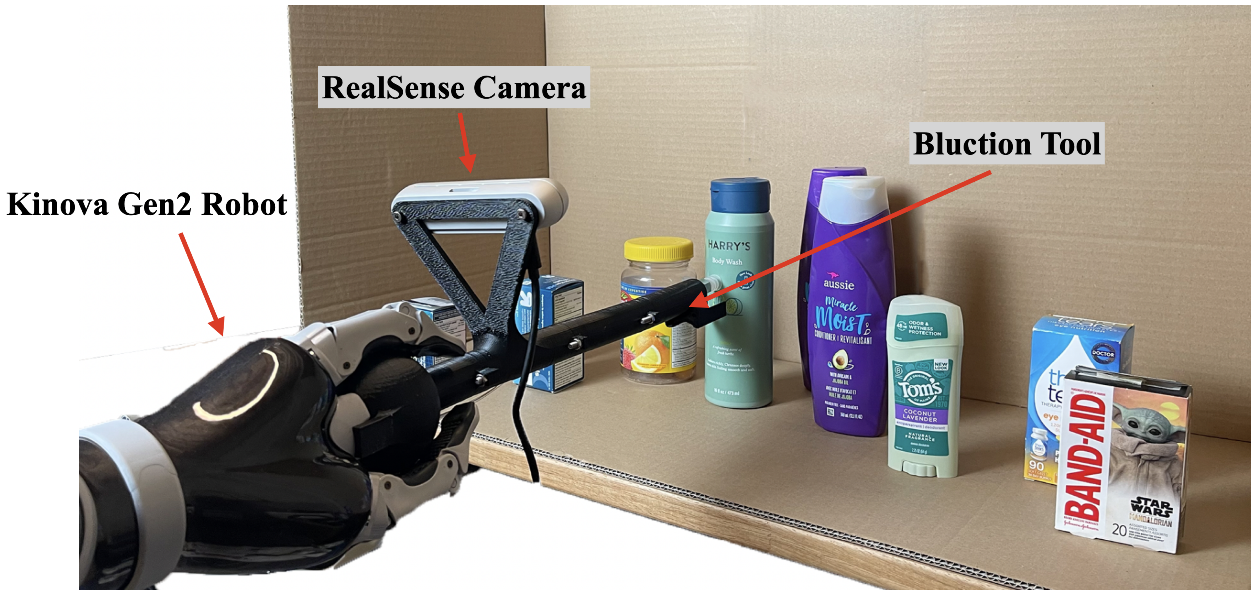

Figure 7.1: Physical setup with a cardboard shelf, pharmacy objects, Kinova Gen2 robot, a bluction tool [34] for extracting objects in the shelf, and an Intel RealSense RGBD camera mounted on the bluction tool.

stage scan of the scene by moving the end-effector to 3 adjacent positions, all of which are closer to the shelf, where the text is more easily readable. At each of these poses, we take a picture of the scene, project the known world position of the objects to the new camera frame, identify text with OCR, and assign each text detection to the object it is contained in. If there are detections on the same object from multiple scan locations, we use the OCR that has the lowest entropy for its distribution, a measure of confidence. During the physical experiments rollouts, when the action given by the policy causes unintentional toppling or a missed grasp due to depth sensor noise, we reset the object to undo the action and run the policy again.

We evaluate the following algorithms in physical experiments:

1. **Spatial NN:** Same as in simulation.
2. **S^4 (PaLM)-Heuristic:** Due to the cost of deploying neural networks in the real world, we test replacing the Spatial NN in S^4 with a simple geometric heuristic: summing the bitwise `OR` of all the segmentation masks in the image along the y-axis to get a 1D distribution along the x-axis.
3. **S^4 (PaLM):** Same as in simulation.

Results are in Table 7.6. An identical set of 12 semantically arranged scenes (starting configurations) generated by the procedure in Section 7.2 is used for each method. We

Table 7.6: Physical Experiment Results (12 trials each). We report the average number of actions taken to reveal the target object as well as the percentage reduction in the number of actions over the spatial neural network.

Method	**# Actions**	**% Reduc**
Spatial NN	4.25 ± 0.64	N/A
S^4 (PaLM)-Heuristic	2.50 ± 0.53	41.2
S^4 (PaLM)	$\mathbf{2.25 \pm 0.46}$	**47.1**

observe that S^4 significantly accelerates mechanical search, reducing the average number of actions by 47.1%. In physical experiments, the depth image has noise while in simulation we have ground truth depth information. This results in the spatial distribution in simulation being strictly better than the spatial distribution in real. This discrepancy between the quality of the spatial distribution makes the semantic distribution more critical in identifying where a target object may lie in physical experiments. Thus, S^4 (PaLM) outperforms the spatial distribution by a larger margin, 47.1%, in physical experiments compared to 32.5% in the simulated pharmacy domain. Moreover, although prior work has shown that the geometric heuristic does not perform as well as the Spatial NN method [35], the results suggest that it can enable comparable mechanical search times to S^4 (PaLM) when coupled with a semantic occupancy distribution.

Chapter 8

Limitations and Conclusion

8.1 TUSK

TUSK is 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. Lastly, when TUSK is applied to a robot untangling problem, the system is able to achieve 64% success in untangling. TUSK has notable limitations: The robot system executing TUSK still depends on a depth camera. Future work will address this and aim to remove depth for the 1D deformable grasping task. TUSK is also tested on only a mono-color workspace.

Future work will investigate generalizing TUSK to function independent of backgrounds, allowing the system to work in a non-solid, multi-color workspace that imitates home environments. Additionally, in real world scenarios, cables vary in color and thickness, so we will also pursue making TUSK invariant to cable appearance. Lastly, on the manipulation side, the untangling system struggles to grasp tight loops. Future work will explore servoing methods to improve grasping reliability in tight regions. A major additional extension of this work might be to build towards the optimal architecture and framework for obtaining a well-calibrated uncertainty or distribution of cable traces from the tracer. At the moment, we have indications that it is possible to obtain an uncertainty estimate or a set of candidate splines by adding perturbations to the start points, but this has yet to be rigorously explored.

8.2 S^4

S^4 is a system to facilitate mechanical search in semantically arranged environments using LLMs. It has the following limitations: (1) S^4 cannot accelerate mechanical search in shelves that are not semantically arranged (e.g., a kitchen pantry after hosting a dinner party),

(2) S^4 requires setting a temperature hyperparameter in the affinity matrix to mitigate the effects of unintuitive or incorrect semantic relationships, (3) semantic refinement of object detection is less effective if the object detection distribution is low-entropy but incorrect (i.e., confidently wrong).

In future work, we hope to mitigate the limitations of S^4 as well: for instance, adding a perception module that estimates the degree of semantic arrangement in a shelf in order to autonomously determine a temperature for spatial semantic search. We are also interested in exploring the potential applications of interactive perception [12] to this task by, for example, developing a policy that interacts with objects for the purpose of uncovering new information about the semantics of a scene. We also hope to extend S^4 to mechanical search in larger spaces such as homes or office buildings, where hierarchical occupancy distributions can model entire rooms (e.g., plates are likely to be found in the kitchen or dining area). Yet another direction for possible extensions involves using LLMs, and specifically affinity values, as featurizers to enable few-shot learning (i.e. calibration of how the affinities should be used). Finally, we would like to relax the assumption of knowing the object names ahead of time, instead being able to perform mechanical search from purely visual observations.

8.3 Conclusion

Ultimately, we have presented two instances of probabilistic state estimation and interactive perception for two different problem domains: deformable object manipulation and object search. For cable state estimation, we train a conditional autoregressive model to generate splines of a cable, with applications to isolation in multi-cable settings and untangling of long cables. For object search, we use a pre-trained LLM to construct zero-shot a semantic spatial distribution that takes into account where a target object could be hidden in addition to where it is likely to be present, enabling efficient mechanical search.

Going forward, it would be interesting to explore a way to generalize the presented ideas into a more unified framework. For example, exploring methods for jointly learning state estimation techniques and manipulation policies that reveal information about the state as useful for a downstream task would certainly be worthwhile. While everything can be framed as a Partially Observed Markov Decision Process (POMDP), it may pay off to recognize a reasonably large subclass of problems where a certain type of inductive bias would improve sample efficiency over end-to-end learning.

Yet another perhaps unrelated learning is just how much of a step up the real world is compared to simulation. Problems of workspace reachability, camera calibration, lack of high-quality sensor data, and open-loop action primitives have no clear solution and always end up costing us severely when attempting to deploy robotic systems in the real world. The methods always matter, but good system implementation and design choices often underlie impressive physical demos.

Finally, this is a really interesting time to be doing research in the field. Foundation models are rapidly on the rise, and sometimes it's hard to dismiss the ever-lingering question of whether a closed-source foundation model trained on many tasks could perform tasks like cable untangling and mechanical search as well as our specialized algorithms. As of the moment I'm writing this thesis, as impressive as these multimodal text and image models appear to be, I would guess not. I do think that the untangling problem remains harder for such a general-purpose model to accomplish. It would certainly be interesting to investigate the properties these models have when it comes to perceiving complex scenes, performing state estimation implicitly, and planning using interactive perception.

8.4 Ending Notes

I've learned a lot during my 5th year MS at Berkeley. I want to thank Professor Goldberg for this opportunity and all my collaborators for helping me learn about the research process, academia, deep learning, and robotics. Through the course of my work, I've been able to better understand the current challenges and opportunities in the field. For now, I am heading to Google DeepMind and am excited to see what the future holds.

Bibliography

[1] Aakanksha Chowdhery et al. "PaLM: Scaling Language Modeling with Pathways". In: *arXiv preprint arXiv:2204.02311* (2022).

[2] Arvind Neelakantan et al. "Text and Code Embeddings by Contrastive Pre-Training". In: *arXiv preprint arXiv:2201.10005* (2022).

[3] Brian Ichter et al. "Do As I Can, Not As I Say: Grounding Language in Robotic Affordances". In: *6th Annual Conference on Robot Learning*. 2022.

[4] Rishi Bommasani et al. "On the Opportunities and Risks of Foundation Models". In: *ArXiv preprint arXiv:2108.07258* (2021).

[5] Susan Zhang et al. "OPT: Open Pre-trained Transformer Language Models". In: *ArXiv preprint 2205.01068* (2022).

[6] Tom B. Brown et al. "Language Models are Few-Shot Learners". In: *ArXiv preprint arXiv:2005.14165* (2020).

[7] Wenlong Huang et al. "Inner Monologue: Embodied Reasoning through Planning with Language Models". In: *arXiv preprint arXiv:2207.05608* (2022).

[8] Yahav Avigal et al. *SpeedFolding: Learning Efficient Bimanual Folding of Garments*. 2022. DOI: `10.48550/ARXIV.2208.10552`. URL: `https://arxiv.org/abs/2208.10552`.

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

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

[11] Ehud Barnea and Ohad Ben-Shahar. "Contextual Object Detection with a Few Relevant Neighbors". In: *ArXiv preprint arXiv:1711.05705* (2017).

[12] Jeannette Bohg et al. "Interactive Perception: Leveraging Action in Perception and Perception in Action". In: *IEEE Transactions on Robotics* 33 (2016), pp. 1273–1291.

[13] 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.

[14] Nicolas Carion et al. "End-to-End Object Detection with Transformers". In: *European Conference on Computer Vision (ECCV)* (2020).

[15] Nicolas Carion et al. “HOLM: Hallucinating Objects with Language Models for Referring Expression Recognition in Partially-Observed Scenes”. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*. 2020.

[16] Google Merchant Center. *Google Product Category*. `https://support.google.com/merchants/answer/6324436?hl=en`. Accessed: 2023-01-31.

[17] Lawrence Yunliang Chen et al. *AutoBag: Learning to Open Plastic Bags and Insert Objects*. 2022. DOI: `10.48550/ARXIV.2210.17217`. URL: `https://arxiv.org/abs/2210.17217`.

[18] Lawrence Yunliang Chen et al. “Optimal Shelf Arrangement to Minimize Robot Retrieval Time”. In: *IEEE International Conference on Automation Science and Engineering (CASE)* (2022).

[19] Zhe Chen, Shaoli Huang, and Dacheng Tao. “Context Refinement for Object Detection”. In: *European Conference on Computer Vision*. 2018.

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

[21] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation. Stichting Blender Foundation, Amsterdam, 2018. URL: `http://www.blender.org`.

[22] Yuchen Cui et al. “No, to the Right: Online Language Corrections for Robotic Manipulation via Shared Autonomy”. In: *ACM/IEEE International Conference on Human-Robot Interaction (HRI)* (2023).

[23] 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)*. 2019.

[24] 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.

[25] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *ArXiv preprint arXiv:1810.04805* (2019).

[26] Santosh K. Divvala et al. “An empirical study of context in object detection”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 1271–1278.

[27] 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.

[28] 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.

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

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

[31] Xiuye Gu et al. “Open-vocabulary Object Detection via Vision and Language Knowledge Distillation”. In: *International Conference on Learning Representations (ICLR)* (2021).

[32] Ryan Hoque et al. “Learning to Fold Real Garments with One Arm: A Case Study in Cloud-Based Robotics Research”. In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 251–257. DOI: 10.1109/IROS47612.2022.9981253.

[33] Ryan Hoque et al. “VisuoSpatial Foresight for Multi-Step, Multi-Task Fabric Manipulation”. In: *Proc. Robotics: Science and Systems (RSS)*. 2020.

[34] Huang Huang et al. “Mechanical Search on Shelves using a Novel “Bluction” Tool”. In: *IEEE International Conference on Robotics and Automation (ICRA)* (2022).

[35] Huang Huang et al. “Mechanical Search on Shelves using Lateral Access X-RAY”. In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2021, pp. 2045–2052.

[36] Huang Huang et al. “Mechanical Search on Shelves with Efficient Stacking and Destacking of Objects”. In: *International Symposium on Robotics Research (ISRR)* (2022).

[37] 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. DOI: 10.1109/TASE.2023.3233949.

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

[39] 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. DOI: 10.1109/TASE.2017.2726689.

[40] Yiding Jiang et al. “Language as an Abstraction for Hierarchical Deep Reinforcement Learning”. In: *Conference on Neural Information Processing Systems (NeurIPS)* (2019).

[41] Sezer Karaoglu, Jan Gemert, and T. Gevers. “Object Reading: Text Recognition for Object Recognition”. In: vol. 7585. Oct. 2012. ISBN: 978-3-642-33884-7. DOI: 10.1007/978-3-642-33885-4_46.

[42] 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. URL: https://arxiv.org/abs/2201.06775.

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

[44] 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.

[45] Andrey Kurenkov et al. “Semantic and Geometric Modeling with Neural Message Passing in 3D Scene Graphs for Hierarchical Mechanical Search”. In: *International Conference on Robotics and Automation (ICRA)* (2021).

[46] Robert Lee et al. “Learning Arbitrary-Goal Fabric Folding with One Hour of Real Robot Experience”. In: *Conf. on Robot Learning (CoRL)*. 2020.

[47] Jacky Liang et al. “Code as Policies: Language Model Programs for Embodied Control”. In: *ArXiv preprint arXiv:2209.07753* (2022).

[48] J. Lin. “Divergence measures based on the Shannon entropy”. In: *IEEE Transactions on Information Theory* 37.1 (1991), pp. 145–151. DOI: 10.1109/18.61115.

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

[50] 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.

[51] 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.

[52] Corey Lynch and Pierre Sermanet. “Language conditioned imitation learning over unstructured data”. In: *Robotics: Science and Systems (RSS)* (2021).

[53] Corey Lynch et al. “Interactive Language: Talking to Robots in Real Time”. In: *ArXiv preprint arXiv:2210.06407* (2022).

[54] Jan Matas, Stephen James, and Andrew J Davison. “Sim-to-real reinforcement learning for deformable object manipulation”. In: *Conf. on Robot Learning (CoRL)*. 2018.

[55] 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.

[56] 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. URL: https://arxiv.org/abs/2001.11051.

[57] Dipendra Kumar Misra, John Langford, and Yoav Artzi. “Mapping Instructions and Visual Observations to Actions with Reinforcement Learning”. In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2017.

[58] Ashvin Nair et al. “Combining Self-Supervised Learning and Imitation for Vision-Based Rope Manipulation”. In: *CoRR* abs/1703.02018 (2017). arXiv: 1703.02018. URL: http://arxiv.org/abs/1703.02018.

[59] Suraj Nair et al. "Learning Language-Conditioned Robot Behavior from Offline Data and Crowd-Sourced Annotation". In: *Conference on Robot Learning (CoRL)*. 2021.

[60] 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.

[61] 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. ISBN: 1-901725-46-4. DOI: `http://dx.doi.org/10.5244/C.26.5`.

[62] 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. DOI: `10.1109/iciip.2013.6707551`.

[63] Alec Radford et al. "Learning Transferable Visual Models From Natural Language Supervision". In: *International Conference on Machine Learning (ICML)*. 2021.

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

[65] 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.

[66] John Schulman et al. "Tracking deformable objects with point clouds". In: *2013 IEEE International Conference on Robotics and Automation*. 2013, pp. 1130–1137. DOI: `10.1109/ICRA.2013.6630714`.

[67] 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.

[68] 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.

[69] Pratyusha Sharma et al. "Correcting robot plans with natural language feedback". In: *Robotics: Science and Systems (RSS)* (2022).

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

[71] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. "Cliport: What and where pathways for robotic manipulation". In: *Conference on Robot Learning (CoRL)* (2021).

[72] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. "Perceiver-Actor: A Multi-Task Transformer for Robotic Manipulation". In: *Conference on Robot Learning (CoRL)* (2022).

[73] Hwanjun Song et al. "ViDT: An Efficient and Effective Fully Transformer-based Object Detector". In: *International Conference on Learning Representations*. 2022.

[74] Yu Song et al. “Vision Based Topological State Recognition for Deformable Linear Object Untangling Conducted in Unknown Background”. In: *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. 2019, pp. 790–795. DOI: 10.1109/ROBIO49542.2019.8961652.

[75] Priya Sundaresan et al. “Learning Rope Manipulation Policies Using Dense Object Descriptors Trained on Synthetic Depth Data”. In: *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*. 2020.

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

[77] 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. DOI: 10.1177/0278364919841431. eprint: https://doi.org/10.1177/0278364919841431. URL: https://doi.org/10.1177/0278364919841431.

[78] Brijen Thananjeyan et al. *All You Need is LUV: Unsupervised Collection of Labeled Images using Invisible UV Fluorescent Indicators*. 2022. DOI: 10.48550/ARXIV.2203.04566. URL: https://arxiv.org/abs/2203.04566.

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

[80] 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.

[81] Ashish Vaswani et al. “Attention Is All You Need”. In: *Neural Information Processing Systems (NeurIPS)*. 2017.

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

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

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

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

[86] 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.

[87] Yilin Wu et al. “Learning to manipulate deformable objects without demonstrations”. In: *arXiv preprint arXiv:1910.13439* (2019).

[88] 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.

[89] Wilson Yan et al. "Learning Predictive Representations for Deformable Objects Using Contrastive Estimation". In: *Conf. on Robot Learning (CoRL)*. 2020.