2013 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS)
November 3-7, 2013. Tokyo, Japan

Tangled: Learning to Untangle Ropes with RGB-D Perception

Wen Hao Lui and Ashutosh Saxena.
Department of Computer Science, Cornell University, Ithaca, USA.
Email: wl378@cornell.edu, asaxena@cs.cornell.edu

***Abstract*— In this paper, we address the problem of manipulating deformable objects such as ropes. Starting with an RGB-D view of a tangled rope, our goal is to infer its knot structure and then choose appropriate manipulation actions that result in the rope getting untangled. We design appropriate features and present an inference algorithm based on particle filters to infer the rope's structure. Our learning algorithm is based on max-margin learning. We then choose an appropriate manipulation action based on the current knot structure and other properties such as slack in the rope. We then repeatedly perform perception and manipulation until the rope is untangled. We evaluate our algorithm extensively on a dataset having five different types of ropes and 10 different types of knots. We then perform robotic experiments, in which our bimanual manipulator (PR2) untangles ropes successfully 76.9% of the time.**

I. Introduction

The environment around us is full of one-dimensional deformable objects such as pet leashes, cables for electronics, shoelaces and even yarns and threads for the artisan. When sailing, there are lifelines, lanyards and so on. In rescue operations, we need to manipulate wires in improvised explosive devices, power lines, etc. Mobile manipulators working in such environments will encounter such objects and need to effectively work with them. In this paper, we present perception and manipulation algorithms for manipulating such items, specifically for untangling ropes.

While most previous work has focused on manipulating rigid objects (or even kinematically linked objects [13]), there is recent interest in manipulating deformable objects. Schulman et al. [22] have developed an algorithm to track the position and configurations of deformable objects such as ropes and towels. Javdani et al. [7] instead focus on perceiving rope bodies (specifically surgical sutures) in a configuration using an energy function, but accurate execution relied on a consistent initial grasping point. Saha et al. [20] work with a broader class of rope bodies, but focus on the motion-planning aspect and had the same end goal of attaining a specific knot configuration. They also face similar limitations, requiring pre-specified grasp points and knowledge of the current rope's initial configuration. Our work focuses on untangling ropes (of various types and appearance) instead—this involves learning appropriate manipulation moves as well.

One key challenge in untangling ropes is to perceive its knot structure—it includes detecting the crossings and ends, and ultimately its state of tanglement. We first use the 3D point cloud data (obtained from RGB-D sensor) of the tangled rope to generate a representation of the rope state, capturing the main points of interest such as intersection points. We not only need to find where the rope segments are, but also connect them in proper order. We represent the rope as a linear graph and use a score function that scores different possible graph structures. We then use a novel rope representation and an inference algorithm based on particle filters to generate a reliable model of the rope, using an max-margin method to learn the weights. Finally, we design robust actions that help the robot to iteratively make progress towards rope untanglement, prioritizing the actions based on certain features (such as slack) in order to follow a minimum-risk strategy.

Fig. 1. Our robot observes a knotted rope using its RGB-D camera, infers its knot structure and chooses appropriate manipulation actions to untangle the rope.

We perform extensive experiments with five different types of ropes and 10 different knot configurations, achieving an average configuration inference accuracy of 76.7%, while the intersection identification accuracy is 89.2%. Subsequently, we executed a real-time inference and untangling experiment on the PR2 robot platform, obtaining a successful untangling rate of 76.9%.

The rest of the paper is organized as follows. We discuss related work in Section II. We present the features and the learning algorithm for perceiving the rope structure in Section III and Section IV respectively. We then describe our representation for rope knot structure in Section V, followed by explaining our learning algorithm for choosing the manipulation action in Section VI. Finally, we describe our offline and robotic experiments in Section VII, and we conclude in Section VIII.

978-1-4673-6357-0/13/$31.00 ©2013 IEEE

II. Related Work

We consider untangling the knots in a rope, where one has to reason about the rope's topological structure as well as complex manipulation strategies while perceiving the rope from RGB-D sensors. We therefore discuss related works in the following three categories.

Knot Representation. Untangling knots requires reasoning about its knot structure. Matsuno et al. [19] and Matsuno and Fukuda [18] focus on interpreting real-world knots as their mathematical counterparts, primarily using knot polynomials. Although knot polynomials are knot-invariant, they are not useful for real-world manipulations because they do not allow us to identify useful manipulation points—they are mainly used for knot isomorphism between two rope configurations. Another shortcoming is the inability to distinguish between over and under-crossings. In their experiments, they focused on tying knots rather than rope untanglement.

Dowker and Thistlethwaite [3] also came up with a useful knot notation for mathematical knots, involving intersection orderings prefixed by a $+$ or $-$ sign to indicate an over/under-crossing. Each intersection is represented by a pairing of one even and one odd crossing numbering. Although true for mathematical knots, this is often violated in physical knots due to rope ends that are separated via another segment; if the ends were joined, they would have created an additional crossing that makes the even-odd pairing always possible. Goldenstein et al. [5], on the other hand, chose to focus on the physical constraints of the system, and came up with a general acyclic graph model to describe deformable objects. However, this model lacks the necessary information for subsequent analysis of manipulation moves that we need for untangling knots.

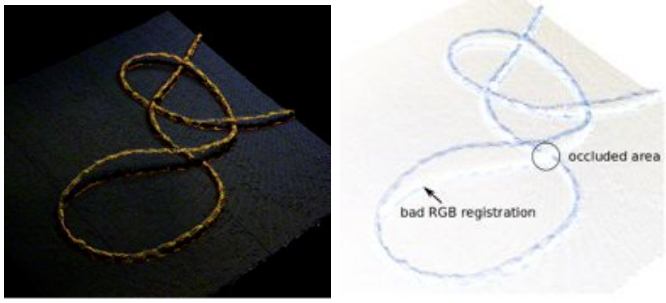
Perception. General deformable-object classifiers [2, 4, 28, 27, 6] are useful for identifying that a given data set contains a rope, but lack the specificity in inferring its configuration and intersections. Javdani et al. [7] instead present an energy model for rope configurations, inferring model parameters from presented ropes. Given multiple possible perception possibilities, the energy model helps to eliminate unlikely high-energy possibilities. The limitation is that the model leverages specific types of suture for perception, and there is a greater challenge to generalize it to a greater class of ropes. Furthermore, our task also uses learning algorithms for inferring the appropriate manipulation moves given the rope structure. Inexpensive RGB-D sensors led to several new applications, including grasping towels and ropes. However, inferring rope configurations is different and challenging. Since we do not start with a known rope configuration, we are unable to make assumptions about rope parameters.

Manipulation. Saha et al. [20] use the Dowker-Thistlethwaite representation for their manipulation planning. This is possible because they start and end the rope with a known state, thus allowing manipulations to be constrained within the set of possible configurations. More interestingly, they make use of rigid objects to guide the structure of the knot, before removing it if necessary for self-knots. This approach is not as suitable when untying knots, since the insertion of guiding rigid objects poses an equally difficult challenge. Both Saha et. al. and Matsuno et. al. [19] explore the manipulation of rope bodies, but their primary application is tying simple knots from a known starting state. Instead, we aim to start from an unknown state and infer the necessary rope configuration and properties to allow us to manipulate it and make progress towards an untangled state.

Interestingly, rope manipulation theory extends to other areas such as fabric manipulation, as demonstrated by Shibata et al. [23] where they approximate the folds in a cloth with ropes of similar build. Both Yamakawa et al. [25] and Vinh et al. [24] deal with the tying of rope knots, which is also similar to our work of untying knots but avoids the difficulty of inferring the initial rope configuration, and identifying manipulation points. Yamakawa et al. [26] goes on to further refine the idea of predicting rope behavior during high-speed manipulation by taking advantage of rope properties that are more stable during quick manipulations.

III. RGB-D Point-Clouds: Features for Learning

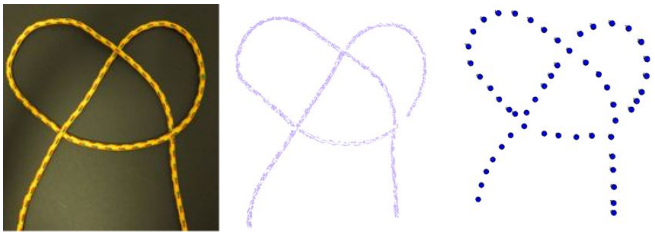
We capture the RGB-D data from the robot's Kinect sensor as a point cloud with color and depth. There are several challenges associated with the RGB-D data from the Kinect sensor (see Fig. 2). First, the resolution of the Kinect limits us to thicker ropes such as power cables and polyester ropes. Second, occlusion occurs when a rope segment overlaps with another segment, making it challenging to infer the rope's structure. Third, the RGB data is misaligned with the depth points and this further limits the accuracy.



(a) Original RGB-D image. (b) Inverted image, with markers.

Fig. 2. Example of the limitations of the Kinect RGB-D sensor.

RGB-D data pre-processing. We start by first removing the underlying table surface via plane fitting. We then over-segment the remaining points using a region-growing method to obtain a collection of small segments, as shown in Fig. 3. (One can potentially improve the performance by using a better RGB-D segmentation method, e.g., [8].) This reduces the size of representation from thousands of points to a few hundred segments. Each segment is then a vertex in the graph $G = (V, E)$, where the vertices correspond to the rope segment and the edges indicate how the segments are ordered relative to each other.



(a) Image of the rope (b) Original point cloud (c) Segments identified.

Fig. 3. Pre-processing of the point-cloud to obtain candidate segments.

We need to obtain an ordering of these segments that represents the correct rope configuration. In a valid graph, all vertices have exactly two neighbors (except for the two end-points). In Section IV, we will define a cost function that scores each possible rope configuration based on certain features we compute next.

Features. We compute several features listed in Table I to form a feature vector ϕ. ϕ_1 and ϕ_8 can be computed directly, while the other features like average distance and angle are computed by aggregating the relevant variable over a traversal of the rope ordering. These features have a higher value in rope orderings that are more likely to be correct. For example, ϕ_1 encourages the inclusion of more vertices in the rope ordering; with more included vertices, we are more likely to have the entire rope configuration instead of a subset of it. ϕ_2 and ϕ_3 reduce the distance between consecutive points in the ordering, while ϕ_4 and ϕ_5 cause the ordering of points to have smaller curvature. This particularly helps in removing outliers. Lastly, ϕ_6 and ϕ_7 cause the ordering to have gentle bends rather than sharp kinks.

TABLE I

LIST OF FEATURES FOR OUR LEARNING ALGORITHM.

Feature	Description
ϕ_1	Binary value to indicate inclusion of vertex in rope configuration
ϕ_2	Length of Edge
ϕ_3	$(\phi_2)^2$
ϕ_4	Cosine of angle made by 2 consecutive edges
ϕ_5	$(\phi_4)^2$
ϕ_6	Difference between consecutive ϕ_4
ϕ_7	$(\phi_6)^2$
ϕ_8	Constant term

(a) Incorrect: ϕ_1 would help here. (b) Incorrect: ϕ_4 and ϕ_5 would help here. (c) Correct graph G^*

Fig. 4. Overlay of various rope configurations over a given set of vertices. The inclusion of various features help in inferring the correct graph G^*.

IV. Learning and Inference

Given a set of segments, our goal is to find the optimal graph G^* that represents the actual structure of the rope. Fig. 4 shows a few graph structures, out of which we want to find the correct graph structure G^*. We define a score function $\Psi(G)$ that indicates how accurately the graph reflects the actual structure of the rope. The problem of finding G^* thus becomes:

$$G^* = \text{argmax}_G \Psi(G) \tag{1}$$

Where $\Psi(G) = w^T \phi(G)$, with w being the weight vector that we will learn from supervised training data. For the graph structure $G = (V, E)$, we can decompose the score function as:

$$\Psi = \frac{1}{|V|} \sum_{v \in V} w_v \phi_v + \frac{1}{|E|} \sum_{e \in E} w_e \phi_e$$

for the node features $\phi_v = (\phi_1, \phi_4, \phi_5)$ and edge features $\phi_e = (\phi_2, \phi_3, \phi_6, \phi_7)$.

A. Inference

We use a particle filter algorithm to find the highest-scoring rope configuration. We start with an initial rope configuration g_0, then perform moves to generate possible graph candidates R. We subsequently take the top scoring candidates (according to Ψ), and repeat the process to generate new candidates R' and so on. (See Fig. 5.) In our experiments, we keep 10 particles. These are the moves we designed (Fig. 6 shows an illustration of these moves):

- m_1 Remove worst segment.
- m_2 Add an excluded segment.
- m_3 Remove worst segment and add an excluded segment.
- m_4 Rotate the segment ordering, so another segment is at the beginning of the rope.
- m_5 Relocate a set of adjacent segments (of any number) to a higher-scoring position in the rope. The set of segments maintain their position relative to each other.
- m_6 Swap the positions of two sets of adjacent segments (of any number). Each set of segments maintains their position relative to each other.
- m_7 Reverse the ordering of a set of adjacent segments (and the points within them).
- m_8 Relocate all segments after a point to their respective highest-scoring rope positions.

Moves m_1, m_2 and m_3 provide opportunities to remove, replace, and include segments as appropriate, thus allowing the number of segments in the rope ordering to shrink or grow correctly. The other moves present permutations over the currently included segments, allowing the ordering to achieve a higher Ψ value. These moves are more complex and cannot be broken down because the intermediate orderings in each move may have a lower Ψ value than the start and end state; if the move was decomposed, this creates local maxima in Ψ for the inference search space, leading to orderings that are sub-optimal.

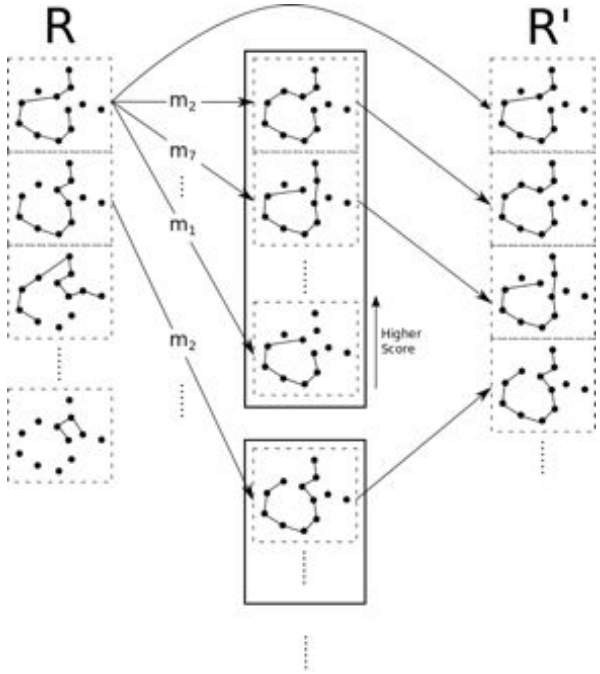
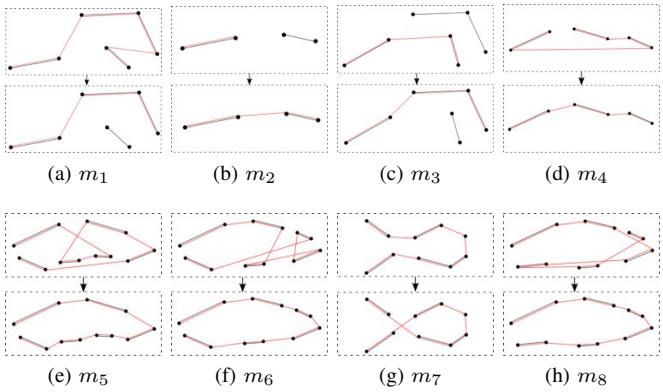


Fig. 5. Illustration of particle filter algorithm. Each iteration updates a set of candidate rope configurations R to R'.



(a) m_1 (b) m_2 (c) m_3 (d) m_4

(e) m_5 (f) m_6 (g) m_7 (h) m_8

Fig. 6. Illustration of different moves for the inference algorithm. The black lines correspond to line segments (invariant over the course of the inference), while the red lines represent the inferred rope ordering.

Moves m_4 and m_8 help ensure that the ends of the rope are correctly identified. m_8 in particular allows the inference algorithm to identify the second rope end, when the first rope end has been found (see Fig. 6h). Moves m_5 and m_6 are just two other actions that help to permute the ordering of segments in the rope. Move m_7 creates a cross-stitching effect, as shown in Fig. 6g. This allows the inference algorithm to switch between possibilities at intersection points.

B. Learning

Our approach is based on large-margin learning [12]. This method for inference has also been used in recent works on RGB-D data [1, 15, 14, 16, 11]. The difference in our work is that we have to reason over different possible graph structures (because of different possible rope configurations).

The label y denotes an ordering of vertices for a given rope configuration (i.e., one possible graph structure in the previous section). We begin by defining a loss function between the ground truth y and an arbitrary label $\hat{y}$:

$$\Delta(y,\hat{y}) = \frac{1}{M(y)}\big(M(y) - D(y,\hat{y})\big) \tag{2}$$

where $D(y,\hat{y}) = \Big(|X(y,\hat{y})| + \sum_{i=1}^{n} |V_i|^{1.5} + \frac{1}{2}\sum_{i=1}^{m} |W_i|^{1.5}\Big)^{\frac{1}{1.5}}$

and $X(y,\hat{y}) = \{x | x \in y \cap \hat{y},$

$\{\text{neighbors of } x \text{ in } y\} \cap \{\text{neighbors of } x \text{ in } \hat{y}\} = \varnothing\}$

M is the number of vertices in the solution y, D is a measure of similarity between two labels, and X is the number of vertices appearing in both $\hat{y}$ and y, but has differing neighbors between y and $\hat{y}$. We further define a streak as a sequence of consecutively matching vertex orderings between y and $\hat{y}$. V_i is the i^{th} streak matching in the same direction, and W_i is the i^{th} streak matching in the opposite direction. Note that when $y = \hat{y}$, we find that $\Delta(y,y) = 0$ since $D(y,y) = M$. We further illustrate it with an example:

$$\begin{aligned} &\text{if } y = (1,3,4,5,7,6,10,9,8) \\ &\quad \hat{y} = (1,2,3,4,5,6,7,8,9,10) \\ &\text{then } X = \{1\} \\ &\quad V = \{(3,4,5)\} \\ &\quad W = \{(10,9,8),(7,6)\} \end{aligned}$$

This loss function was designed with the following in mind:

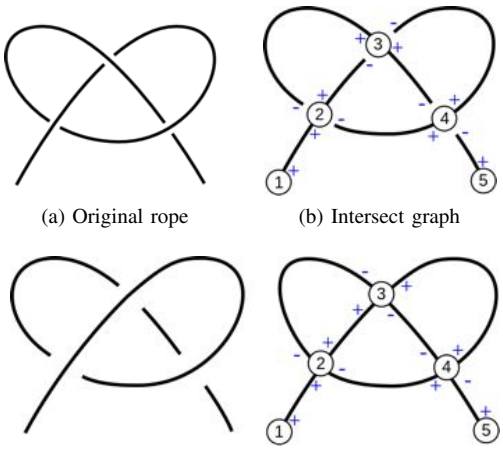
- A L-1.5 norm was chosen to allow a small bonus for longer matching streaks, while not trivializing shorter streaks.
- Streaks that are in the wrong direction are penalized by a discount factor.
- The loss is normalized from 0 to 1, with 0 being the perfect solution.

Since the objective function is convex in the weights w, we use the efficient plane-cutting algorithm for training structural SVMs [12] to find the optimal set of weights. We then formulate the training of the weights as the following convex problem:

$$\begin{aligned} \min_{w,\xi} \quad & \frac{1}{2} w^T w + C\xi \\ s.t. \quad & \forall \hat{y} : w^T \phi(\hat{y}) \geq \Delta(y,\hat{y}) - \xi \end{aligned} \tag{3}$$

where ξ is a slack variable for the SVM learning algorithm. Useful estimates of $\hat{y}$ at each iteration of the SVM are obtained using our inference algorithm discussed in the previous section.

Intersection points are not explicit in the linear ordering of vertices for the earlier inference algorithm, but are necessary for the graph representation. We project all line segments formed by the vertex ordering into a plane, then look at depth values in order to determine the over and under-crossings. Once all intersection points are identified, we then use the vertex ordering to determine the correct edges to add between intersection nodes on the graph representation. An example is shown in Fig. 7.



(a) Original rope (b) Intersect graph

(c) Untangled Rope

Fig. 7. Correspondence from rope to intersect graph $\mathcal{G}$. Intersections and rope ends are labeled according to L, described in Equation (4). The untangled rope differs from the original at intersect 3, allowing it to satisfy the untangled state condition in Section V.

V. Rope Knot Representation

We need to reason about a given rope's knot structure (e.g., Fig. 7) for choosing a suitable manipulation policy. Given the graph structure inferred from the RGB-D image, we present a representation based on the Dowker-Thistlethwaite formulation [3], but modified for physical knots and better intuitive understanding.

Using the earlier segment-based graph $G = (V, E)$, let the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the intersection graph. Each vertex $v \in \mathcal{V}$ represents either an intersection or an end on the rope, along with its position in 3D space. We create a labeling

$$L(v) = n, \quad n \in \{1, 2, ..., |\mathcal{V}|\} \tag{4}$$

such that during a traversal of the rope from the starting end, v is the n-th intersection (or rope end) crossed for the first time. Although there are two possible labelings depending on which rope end is chosen for starting the traversal, this does not affect the correctness of our subsequent manipulations. The labelings are shown in Fig. 7.

Edges $e \in \mathcal{E}$ on the graph correspond to an ordered list of vertexes in V on the rope that connect one intersection $v \in \mathcal{V}$ to another. We define the following functions:

$$X(v_1, v_2) = \begin{cases} 1 & \text{if } e \text{ crosses over the other segment at} \\ & v_2, \text{ or if } v_2 \text{ is a rope end} \\ -1 & \text{otherwise} \end{cases}$$

$$C(e) = \big(X(v_2, v_1), X(v_1, v_2)\big) \quad \text{where } e = (v_1, v_2)$$

Every edge then has a property $C(e)$ that determines if the edge crosses over or under at the intersections on both ends. In Fig. 7b, the ends of the edges are annotated using $+$ and $-$ instead for brevity.

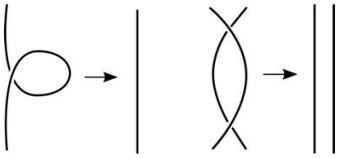
Untangled State. When manipulating a rope, our goal is to restore it to an untangled state. $\mathcal{G}$ is very useful in making statements about entanglement. In the simplest case, a length of rope with no intersections (i.e. $|\mathcal{V}| = 2$) can be considered untangled. However, ropes with one or more intersections can also be considered untangled if they do not form a knot. If we pick up one exposed end of the rope and lift it up until no part of it touches the original surface it was lying on, it would form a straight line from end to end. This is true for the rope shown in Fig. 7c, which has a slight but important variation from Fig. 7b.

We have a stronger test for untanglement: traversing along the rope from end to end and checking if the rope performs an under or over-crossing at each intersection, there is at most one over-crossing to under-crossing transition (or under-crossing to over-crossing). This is sufficient but not necessary for untanglement.

VI. Rope Manipulation

Having obtained the graph mapping of the intersection points from RGB-D data, we now present our approach to choose the manipulation actions (or rope untangling moves). While many manipulation moves may be possible, some are preferred because of criterion such as empty space, slack in the rope, etc. Our manipulation moves belong to the following categories:

- **Reidemeister moves** are actions that remove intersections from a rope anywhere along its length. The type I move in Figure 8a is the simple untwisting of a loop in the rope, originating and ending at the same intersection. The type II move in Fig. 8b separates rope segments that overlap, but are not entangled with each other.
- **The node deletion move** is executed only at the ends of a rope. We pull the rope out from the first under-crossing intersection, on the side opposite the rope end. By repeating this move alone we can satisfy the earlier criteria for untanglement, where over-crossings all precede or follow all under-crossings.



(a) Type I (b) Type II

Fig. 8. Reidemeister Moves.

Fig. 9 shows an example of the node deletion move being applied, causing a transition from the leftmost rope to the middle rope. Next, the Reidemeister type II move can be applied, causing the rope to reach an untangled state.

Fig. 9. Process of untangling a rope using a node deletion followed by a Reidemeister move.

A. Choosing the optimal Manipulation Action

A single rope configuration may have several different kinds of manipulation moves available. Each move could also have multiple methods of execution, because the robot can move the rope segment to different possible places. Motivated by Jiang et al. [10], we frame manipulation as a learning problem. We define the score of a manipulation action a as:

$$\Gamma(a) = \omega^T \chi(a)$$

where $\chi(a)$ refers to the features for action a and ω is the weight vector.

As Fig. 10 indicates, choosing which manipulation action to use is tricky. One needs to not only take into account which segment of rope is over another, but also other aspects such as the amount of slack in the rope, quality of configuration at the end of the action, and area available around the release point (so that the robot maintains slack in the rope). In order to capture these effects, we design the following features:

- χ_1 Distance to nearest rope segment that is not manipulated.
- χ_2 $\frac{1}{\chi_1}$
- χ_3 $cos(\alpha)$, where α is the angle made by the drop point and the rope after u.
- χ_4 χ_3/χ_1
- χ_5 Number of segments crossed when a line is drawn from the drop point to u.

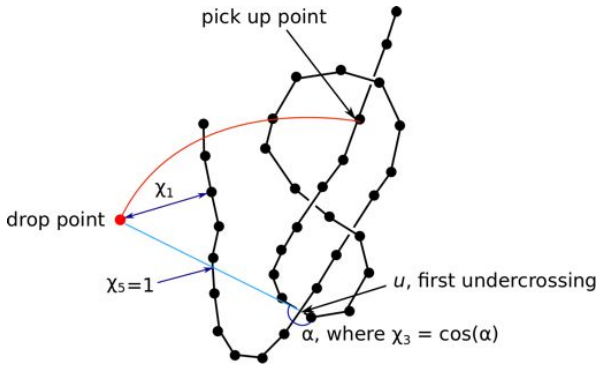


Fig. 10. Illustration of features for a move. We grab the rope at the pick up point node, then move it along the red line to the drop point. The new rope configuration determines the feature scores for χ_1, χ_3, χ_5.

Since the number of moves in a given configuration is small, we simply calculate Γ for all moves, searching the entire execution-space for each move, and choose the highest-scoring candidate for actual execution on the PR2. For each move, in addition to moving the selected segment, the other segment at the identified intersection needs to be held in place as well to prevent it from being dragged along and rendering the relative position of the segments unchanged. We do so by using both the hands of our robot. See some snapshots in Fig. 13. After taking one manipulation action, our robot infers the new state of the rope from new RGB-D data and repeats the process until the rope is untangled.

VII. Experiments

A. Inference and Learning Results

Data. In order to evaluate our learning algorithm for estimating the rope structure, we collected an RGB-D data set of 50 examples, with five different types of rope (Fig. 11), and 10 different knot configurations (Fig. 12).

Evaluation Metrics. We evaluated our learning algorithm for perception using 5-fold cross-validation, using a 40-10 split. Although the loss function is a good metric for the accuracy of the inference algorithm, we report results on the following three metrics:

1) Loss: The loss function, as described in Eq. (2).

2) Intersection-Graph: 1/0 classification of the correctness of the inference ordering. An inferred ordering is correct if it generates the same intersection graph as the correct ordering solution.

3) Nodes: We generate an intersection graph for both the inference and solution ordering, and check the percentage of intersection nodes that are inferred correctly.

$$Nodes = \frac{|V_{inferred} \cap V_{solution}|}{\max(|V_{inferred}|, |V_{solution}|)}$$

Results. Table II shows the results, where we obtain an average success-rate of 76.7% for intersection graph inference and 89.2% for node inference. The main source of inference failure was poor segmentation of the rope from the surface it was resting on, causing the rope's shadow to be interpreted as part of the rope. As anticipated, the harder ropes (containing more intersections) had a higher average loss. However, the inference of the intersection positions and overall intersection graph was largely consistent across all rope configurations, indicating that our inference algorithm would scale well with complex rope configurations that have even more intersections.

B. Robotic Experiments and Results

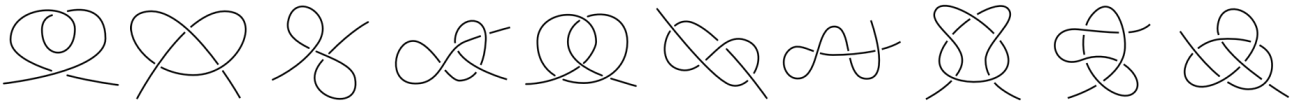
In order to ensure that our rope structure inference and rope manipulation algorithms are robust given the physical constraints of the rope, we performed robotic experiments on the ten different rope configurations using the PR2 robot platform mounted with a Kinect sensor.

Given the RGB-D input, we use our rope configuration inference algorithm to convert it into the rope representation. Our move inference algorithm then generates a move, indicating the point in space where the rope segment is to be grasped, and the point in space that it should be moved to. The PR2 executes these commands, creating a new rope state that needs to be re-inferred from a new RGB-D image. This process is repeated until the untangled state is reached. Because the PR2's hands are relatively large, we need to completely release the rope before taking the RGB-D input to avoid the hands occluding the rope. This creates uncertainty in manipulation.

We consider it a success if the rope is completely untangled after 5 moves. Experimental results are shown in Table III. The table reports the following metrics:

(a) Nylon rope - Knot 8 (b) Manila rope - Knot 3 (c) Power cable - Knot 5 (d) Hose - Knot 6 (e) Polyester rope - Knot 10

Fig. 11. Pictures of different types of ropes in various configurations in our dataset.



(a) Knot 1 (b) Knot 2 (c) Knot 3 (d) Knot 4 (e) Knot 5 (f) Knot 6 (g) Knot 7 (h) Knot 8 (i) Knot 9 (j) Knot 10

Fig. 12. Different configurations of knots in our dataset.

TABLE II

Offline learning results for rope perception, split across rope types and difficulty of rope configuration. **Loss:** Loss against ground truth, computed as in Eq. (2). **Int. Graph:** % intersection graphs that are correctly inferred. **Nodes:** % nodes in the intersection graph that are correctly inferred.

	Easy (Knots 1, 2, 3)			Medium (Knots 4, 5, 6, 7)			Hard (Knots 8, 9, 10)			Average		
	Loss	Int. Graph	Nodes	Loss	Int. Graph	Nodes	Loss	Int. Graph	Nodes	Loss	Int. Graph	Nodes
Nylon	0.209	0.667	0.667	0.193	0.750	0.938	0.376	0.333	0.821	0.259	0.583	0.809
Hose	0.120	0.667	0.800	0.187	0.750	0.854	0.342	0.333	0.689	0.216	0.583	0.781
Manila	0.281	1.000	1.000	0.307	0.500	0.830	0.361	1.000	0.944	0.316	0.833	0.925
Poly	0.213	1.000	1.000	0.312	1.000	1.000	0.375	1.000	1.000	0.300	1.000	1.000
Power	0.236	1.000	1.000	0.263	0.500	0.838	0.226	1.000	1.000	0.242	0.833	0.946
Average	0.212	0.867	0.893	0.252	0.700	0.892	0.336	0.733	0.891	0.267	0.767	0.892

TABLE III

Robot experiment results. The average optimal number of steps to untangle each rope is also shown.

	Node Inference	Rope Inference	Move Inference	Move Execution	Untangled Rope	# Steps to Untangle	Optimal #Steps to Untangle
Nylon	82.7%	43.5%	95.7%	69.6%	80.0%	4.50	2.75
Hose	76.1%	34.8%	78.3%	95.7%	100.0%	4.60	2.80
Manila	70.2%	28.6%	60.7%	64.3%	40.0%	4.00	2.50
Poly	78.5%	33.3%	87.5%	62.5%	80.0%	4.75	2.75
Power	60.7%	12.1%	78.8%	66.7%	83.3%	5.40	2.60
Average	72.6%	29.0%	79.4%	71.0%	**76.9%**	**4.75**	2.70

1) Node Inference: Percentage of nodes that have a correct XYZ co-ordinate and over/under crossing allocation.

2) Rope Inference: Percentage of cases where all nodes are correctly inferred.

3) Move Inference: A correct and feasible move is chosen for the given rope configuration.

4) Untangled Rope: The rope meets the criteria for untanglement (see Section V).

We observe that node inference and rope inference successes are significantly lower in the robot experiment compared to the offline learning experiments. This is largely due to the rope often being manipulated by the robot into configurations that are difficult to perceive. Some parts of the rope fell outside the field of vision of the RGB-D sensor, and other segments occluded important intersections. The correctness of the inferred moves is also largely dependent on the accuracy of the inferred rope structure, and is expected to increase with better isolation of the rope RGB-D data from its environment.

Note that despite the low accuracy in perception, our manipulation choices are robust enough to ensure a higher degree of success. This is because we do not need to necessarily infer the complete rope configuration—even if part of the rope configuration is incorrect, manipulation on the other parts could still be successful. This allows us to untangle 76.9% of the presented knots, spread over various difficulty levels and rope types. Fig. 14 shows various perspectives during manipulation experiments.

In our experiments, most of the failures were caused by motion planning failures of the OMPL library (71% of failures), and self-collisions with the other robot arm that anchors part of the rope during manipulation (24% of failures). We try to mitigate the occurrence of arm-to-arm collisions by preferentially selecting moves that allow a wide berth between both arms, and do not cause arms to cross over each other. However, a more principled approach that takes into account arm path planning while choosing manipulation policies would improve performance in future work. Given the available Reidemeister and node deletion moves, we calculated the minimum number of moves needed to manipulate each knot into an untangled state. We see that the PR2 took an average of 4.75 moves, more than the ideal average of 2.70—this is largely due to the failures in move execution and poor grasping. If we employ learning in our grasping approach (e.g., [17, 21, 9]) and use improved arm motion planning, the performance should improve.

VIII. Conclusions and Future Work

We presented a learning algorithm for inferring and manipulating rope structures. Starting with a point-cloud obtained from an RGB-D camera, we designed appropriate features

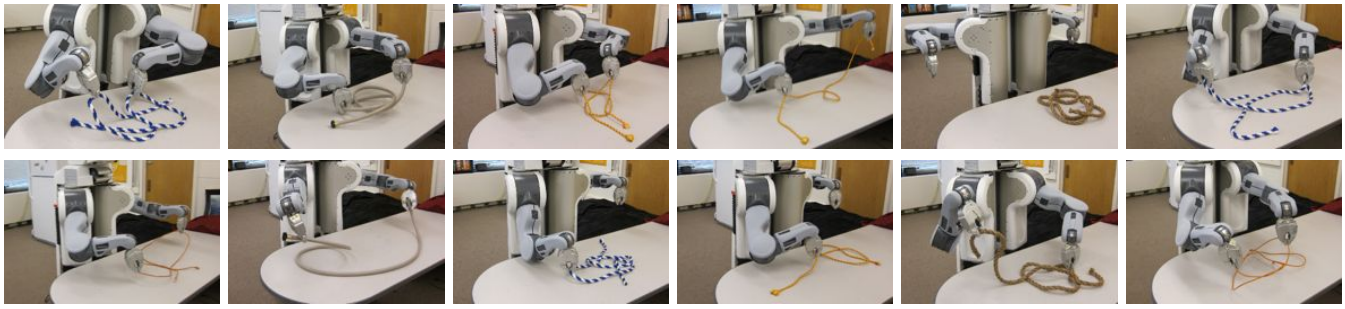


Fig. 13. Snapshots of our robot untangling various ropes.

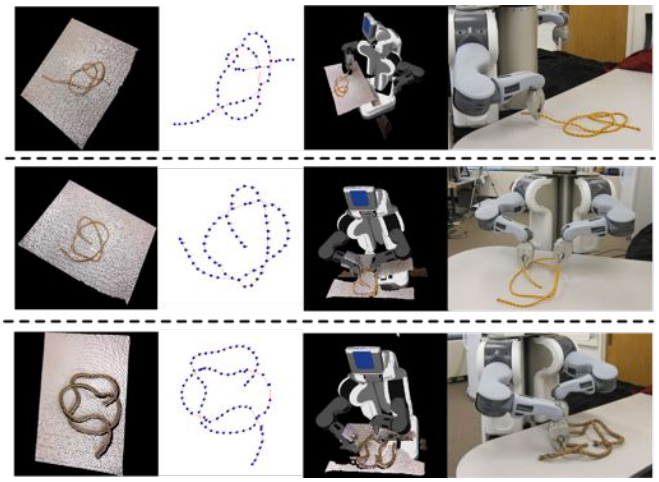


Fig. 14. Example of various perspectives during an experiment. From left to right: RGB-D data view, the inferred graph view, the scene in the RViz motion planner, and a snapshot of the PR2 in action.

that our learning algorithm uses to first infer the rope's knot structure and then choose an appropriate manipulation action to untangle the knots in the ropes. The algorithm used a particle filtering approach for inference and a max-margin approach for learning the parameters. Our extensive experiments first verified the performance on an offline dataset, and then our robotic experiments showed that our robot can untangle ropes 76.9% of the time.

There are a few directions for future work. Currently, we rely solely on visual perception for identifying rope's knot structure, which may not be enough in very tight knots where multiple intersections overlap each other. One possibility is to take an active perception approach, e.g., turn the rope over or introduce additional slack for perceiving it better. Tactile sensing could also help in better perception while manipulation. Finally, better planning algorithms that consider the limitations of the arms as well as uncertainty in perception would also improve the performance.

Acknowledgments. We thank Hema Koppula and Sanford Johnson for useful discussions. This work was supported by ARO award W911NF-12-1-0267, the Microsoft Faculty Fellowship and the NSF Career Award to one of us (Saxena).

REFERENCES

[1] A. Anand, H. Koppula, T. Joachims, and A. Saxena. Contextually guided semantic labeling and search for 3d point clouds. *IJRR*, 32(1): 19–34, 2013.

[2] C.C. Chang. Deformable shape finding with models based on kernel methods. *TIP*, 15(9):2743–2754, 2006.

[3] C. Dowker and M. B. Thistlethwaite. Classification of knot projections. *Topology and its Applications*, 16(1):19–31, 1983.

[4] P.F. Felzenszwalb. Representation and detection of deformable shapes. *TPAMI*, 27(2):208–220, 2005.

[5] S. Goldenstein, C. Vogler, and D. Metaxas. Directed acyclic graph representation of deformable models. In *WMVC*, 2002.

[6] S. Goldenstein, C. Vogler, and L. Velho. Adaptive deformable models. In *SIBGRAPI*, 2004.

[7] S. Javdani, S. Tandon, Jie Tang, J.F. O'Brien, and P. Abbeel. Modeling and perception of deformable one-dimensional objects. In *ICRA*, 2011.

[8] Zhaoyin Jia, Andy Gallagher, Ashutosh Saxena, and Tsuhan Chen. 3d-based reasoning with blocks, support, and stability. In *CVPR*, 2013.

[9] Y. Jiang, S. Moseson, and A. Saxena. Efficient grasping from rgbd images: Learning using a new rectangle representation. In *ICRA*, 2011.

[10] Y. Jiang, M. Lim, C. Zheng, and A. Saxena. Learning to place new objects in a scene. *IJRR*, 31(9), 2012.

[11] Y. Jiang, H. Koppula, and A. Saxena. Hallucinated humans as the hidden context for labeling 3d scenes. In *CVPR*, 2013.

[12] T. Joachims, T. Finley, and C. Yu. Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59, 2009.

[13] D. Katz, M. Kazemi, A. Bagnell, and A. Stentz. Interactive segmentation, tracking, and kinematic modeling of unknown 3d articulated objects. In *ICRA*, 2013.

[14] H. Koppula, A. Anand, T. Joachims, and A. Saxena. Semantic labeling of 3d point clouds for indoor scenes. In *NIPS*, 2011.

[15] H. Koppula, R. Gupta, and A. Saxena. Learning human activities and object affordances from rgb-d videos. *IJRR*, 32(8):951–970, 2013.

[16] H.S. Koppula and A. Saxena. Anticipating human activities using object affordances for reactive robotic response. In *RSS*, 2013.

[17] I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. In *RSS*, 2013.

[18] T. Matsuno and T. Fukuda. Manipulation of flexible rope using topological model based on sensor information. In *IROS*, 2006.

[19] T. Matsuno, D. Tamaki, F. Arai, and T. Fukuda. Manipulation of deformable linear objects using knot invariants to classify the object condition based on image sensor information. *IEEE/ASME Transactions on Mechatronics*, 11(4):401–408, 2006.

[20] M. Saha, P. Isto, and J.-C. Latombe. Motion planning for robotic manipulation of deformable linear objects. In *ICRA*, 2006.

[21] A. Saxena, J. Driemeyer, J. Kearns, and A.Y. Ng. Robotic grasping of novel objects. In *NIPS*, 2006.

[22] J. Schulman, A. Lee, J. Ho, and P. Abbeel. Tracking deformable objects with point clouds. In *ICRA*, 2013.

[23] M. Shibata, T. Ota, and S. Hirai. Virtual rope theory for fabric manipulation. In *ISAM*, 2009.

[24] T.V. Vinh, T. Tomizawa, S. Kudoh, and T. Suehiro. A new strategy for making a knot with a general-purpose arm. In *ICRA*, 2012.

[25] Y. Yamakawa, A. Namiki, M. Ishikawa, and M. Shimojo. Knotting manipulation of a flexible rope by a multifingered hand system based on skill synthesis. In *IROS*, 2008.

[26] Y. Yamakawa, A. Namiki, and M. Ishikawa. Simple model and deformation control of a flexible rope using constant, high-speed motion of a robot arm. In *ICRA*, 2012.

[27] S.Y. Yeo, X. Xie, I. Sazonov, and P. Nithiarasu. Geometrically induced force interaction for three-dimensional deformable models. *TIP*, 20(5): 1373–1387, 2011.

[28] Q. Zhou, L. Ma, M. Celenk, and D. Chelberg. Object detection and recognition via deformable illumination and deformable shape. In *ICIP*, 2006.