COMS 6998 007 Report: Crossing Detection with Assistance of Robot Interaction

Ruoyu Xue, Peter N. Belhumeur

Department of Computer Science
Columbia University
rx2194@columbia.edu, belhumeur@cs.columbia.edu

Abstract

We proposed an innovate method to find the crossing configuration based on the combination of vision-based crossing detection and robot interaction, which attempts to answer an open question on whether the robot could assist the vision-based methods to make a better perception. We proposed a method to generate a simulation dataset, applied non-learning based and learning-based model to get the preliminary crossing prediction, and created robot interaction to optimize the prediction and to learn a detailed crossing configuration.

1 Introduction

Robot manipulation on linear deformable objects(LDO) has made significant progress in recent years, however, compared with the manipulation on rigid-bodies, there still exists some limitations: First, as the simulation of LDO is much harder than rigid-bodies, it is hard to generate a simulation dataset. Second, it is hard for deep learning networks to predict the crossing configuration(at the crossing, which rope segment is on the top of the other). Considering these challenges, we proposed to combine the vision-based detection and robot interaction to predict the location of crossings and their configuration. Our **contribution** is: created an algorithm to make the simulation dataset on softGym[3], and proved that the robot interaction could improve the performance of vision-based detection.

2 Relate Works

Learning based detection: Crossing detection is essential for rope untangling, and most of the state-of-the-art models could be categorized as two categories: keypoint-based and image-based. Keypoint-based detection requires the pre-processing to obtain detailed information of the rope, such as its graph, to make the prediction. [4] construct a graph stored the keypoints and their order, and predict the crossing based on the relation of vertices and edges. Image-based detection directly applied supervised learning to predict the location of crossings based on raw RGB images. For example, [8] proposed the topological state recognition based on YOLO[6] to predict the location and order of crossings. Instead of getting the exact location of crossings, [1] directly predict the pick-up and holding point on the rope for untangling planning. Most of learning based methods are based on supervised learning, and required much effort for labeling, such as the bounding box of crossings, order of pixels from one endpoint to the other endpoint.

Non-learning base detection: Non-learning based methods are usually applied on complex rope configurations, for example, a messy headphone cable, compared with learning-based methods. [5] used fitting curves to represent the configuration of ropes and [2] applied energy functions and linear interpolation planner to get the crossing. However, their generalization ability is limited compared with learning based methods.

3 Approach

We divided the whole structure as two tasks: perception and interaction. This part will illustrate the learning based and non-learning based algorithms for perception, and the robot interaction algorithm to get the detailed crossing configuration and optimize the predicted crossing location with non-maximum suppression.

3.1 Perception

Branch-based detection: This is the non-learning based detection. We sampled the feature-points shown in Fig. 1(c) by computing the mean coordinate of all skeleton pixels in sliding windows. for each feature-point, it records 4 nearest neighbors as

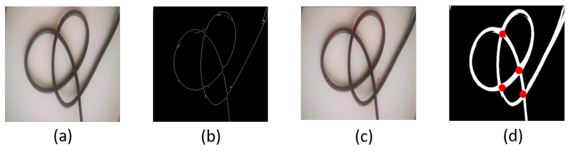

Figure 1: Branch-based detection. It extracts the skeleton of the rope as (b), sampled by sliding windows to get the feature-point image. Red points in (d) is the result of preliminary crossings.

its branches. Four constrains are designed to filter the feature-points to get the preliminary crossings. The thresholds are obtained adaptively to improve its generalization ability.

- Windows with crossing always have more skeleton pixels compared with windows without crossings. Only 20% feature-points are selected as they have larger corresponding skeleton area.
- Branches must belong to the rope. One example of the branch that does not satisfy this constrain is shown in Fig. 2(f).
- Crossings usually have 3 or more different branches. Branches are classified based on their angles, and those with similarly angles will be counted as the same branch.
- Filter fake branches. Each branch has an α as:

$$\alpha = arccos \left\langle (p_{bran} - p_{cent}), (p_{bran} - p_{neigh}) \right\rangle \quad (1)$$

Here p_{bran} is the branch point, p_{cent} is the center of branches, p_{neigh} is the nearest point of p_{bran}. Branches with small α values will be regarded as fake. For example, the red branch in Fig. 2(d) is fake, and the blue is real.

Learning based detection: As our model is applied on real world data, considering the effort to label the ground truth location of crossings, we simplified this task as a bi-classification. We still sampled the feature-points on the 512*512 image with the same method mentioned in branch-based detection, and generated 40*40 sub-images with their centers to be the feature-points. We manually took 100 RGB images and generated 5000 sub-images, with 2500 positive sub-images(contain crossings) and 2500 negative sub-images(do not contain crossings). Some examples are shown as Fig. 3.
We constructed a convolution neural network with

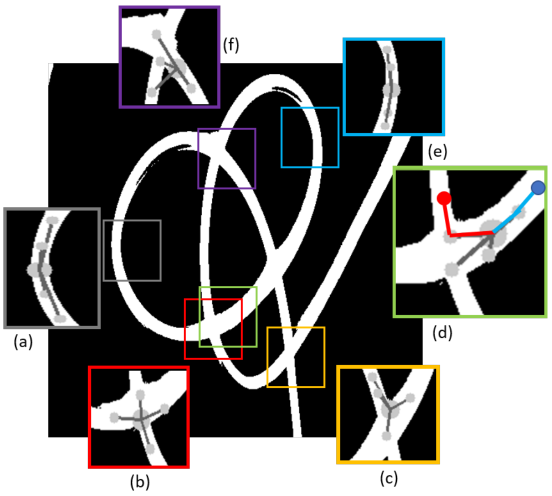

Figure 2: (a)-(f) are sub-figures represent different shape of branches. (b), (c) are supposed to be the preliminary crossings. In each sub-figure, the bigger circle is the center of the branches.

5 convolution layers and 3 fully connected layers to predict if the sub-image contains crossings or not. Inspired by [7], we assigned each sub-iamge an confidence value based on the output of softmax layer, and applied non-maximum suppressing on those positive predictions to get the preliminary crossings. We choose four nearest neighbors of the preliminary crossing as its branches.
We are still working on this part as this learning based method does not utilized the state-of-the-art keypoint detection. We are applying MTCNN[9], which is used on face keypoint detection, to directly predict the location of preliminary crossings. The desired structure is shown as Fig. 4. The result of interaction should be fed back to the perception network to optimized its prediction.
There are some limitations of both perception algorithms: some of the preliminary crossings are not perfectly aligned with the ground truth, and they do not have knowledge of the crossing configuration. We believe that the deep learning network would predict such crossing configuration when ground truth is available. However, obtaining such ground truth requires much effort, which is against our purpose to simplify the LDO labeling. Therefore, the robot interaction is introduced to optimize the crossing locations and to get crossing configurations.

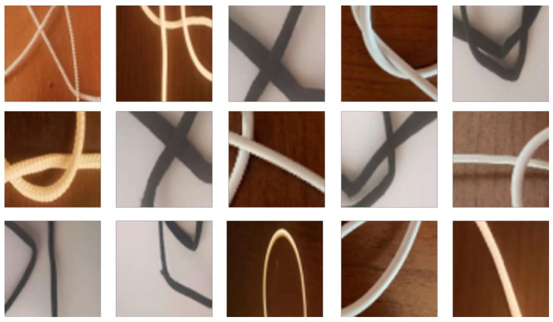

Figure 3: Examples of training data. The first and second row are positive examples, the third row is the negative examples.

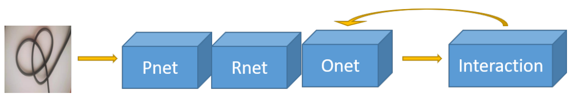

Figure 4: Desired workflow of learning based detection.

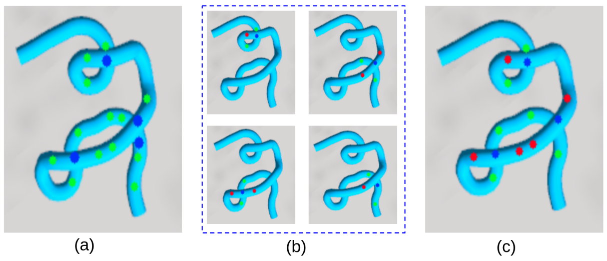

Figure 5: (a) is the output of perception. Blue points are the predicted preliminary crossings, green points are their corresponding branches. (b) is the result of robot interaction, (c) is the result of crossing optimization. Red points are from the its top segment, and green points are from its bottom segment.

3.2 Interaction

Interaction is separated into two parts: robot interaction and crossing optimization. Robot interaction learns the knowledge of rope configuration, and crossing optimization defines the optimal crossings with their configuration for manipulation. The workflow is shown in Fig. 5.

Robot interaction: As mentioned in [3], rope is modeled as a sequence of particles, where each pair of neighbouring particles are connected by a spring. Based on the distance of two particles on the rope, two terms are defined as *absolute distance* and *relative distance*. For a rope with an arbitrary shape, relative distance is their euclidean distance of two particles in the 3D space, and absolute distance is the euclidean distance when the rope is straightened. Based on the experience of rope manipulation in the real world, we have the stretch principle:

- *When stretching a rope with a suitable distance by holding two particles, if their relative distance is small, those with a short absolute distance cannot be stretched, and those with a long absolute distance are easy to be stretched.*

Based on it, the robot with two pickers will interact with the rope to predict the desired crossing configuration. The action of two pickers is defined as:

$$A = \{a^{pick} = (0, 0, height), a^{hold} = (0, 0, 0)\} \tag{2}$$

The output of perception is C containing n preliminary crossings. c_i is the coordinate of i^{th} crossing, b_i^j is the j^{th} branch of i^{th} crossing.

$$C = \{c_0 : b_0^0, b_0^1, ...; \quad c_1 : b_1^0, b_1^1, ...; \quad ...; c_n\} \tag{3}$$

The action A is applied on each pair of $\{c_i, b_i^j\}$. SoftGym provides a $threshold$ of the maximum distance of each two neighbouring particles. Based on the stretch principle, the robot will make the prediction:

$$\text{label of } b_i^j = \begin{cases} \text{top} & max\,\|p_k - p_{k+1}\|_2 > threshold \\ \text{bottom} & max\,\|p_k - p_{k+1}\|_2 < threshold \end{cases} \tag{4}$$

Here p_k is the k^{th} particle of the rope. If the branch is predicted as top, it belongs to the top segment of the crossing, otherwise it belongs to the bottom segment. A visualization is shown in Fig. 5(b), the red points are branches labeled as top and green points are labeled as bottom.

Crossing optimization: In some cases, preliminary crossings may contain duplication and false prediction. For example, in Fig. 5(a), there are four preliminary crossings but only 3 ground truth crossings, which is regarded as duplication. After interaction, each crossing is assigned a score based on the number and shape of its top and bottom branches. As shown in Table 1, Top and Bottom is the number of top and bottom branches. TA is the included angle of two top branches, and BA is the included angle of two bottom branches.

Crossings will be grouped if they are close, and the

Score	Top	Bottom	Included Angle
1.0	2	2	TA and BA $> 0.8\pi$
0.9	2	2	TA or BA $> 0.8\pi$
0.8	2	1	-
0.7	1	2	-
0	other values		

Table 1: Score table

one with the highest score will be selected as an optimal crossing. If the group have the same score, their mean coordinate will be selected.

4 Experiments

4.1 Environment Setup

The simulation environment is softGym, a benchmark environment based on Nvidia FleX and openAI Gym. The experiment is conducted on a personal computer, CPU 3.20GHz with 16GB RAM and GPU Nvidia RTX 1650, using Ubuntu 18.04, cuda 11.1 and nvidia driver 460.32.

4.2 Data Generation

We proposed two methods to generate the simulation data. The random method is used to test our algorithms, and we are still working on the graph method for generating knots.

Random method: The robot randomly picks up two particles and randomly chooses an action. The perception algorithm is applied every step to find the number of generated crossings, and the robot will continue this random process until it generates enough crossings.

Graph method: As crossing detection is utilized by rope untangling, we tried to generate knots rather than crossings to ensure the extension of our work. As shown in Fig. 6(a), we manually drew a rope image and sampled feature-points of it. The we applied the traverse algorithm to find the order of those feature-points. The key idea is to minimize the orientation offset every time a new point is passed during the traverse.

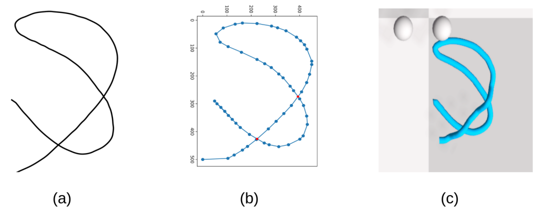

Figure 6: The process of generating crossings. (a) is an randomly drawn image, (b) is the constructed graph encoding the trajectory of particles. (c) is the simulation result.

Algorithm 1 Traverse algorithm

```
Input:
$Z = \{(x_1, y_1), (x_2, y_2), ...(x_n, y_n)\}$. Coordinates of feature-points from $N$ sliding windows
Output:
$D = \{(x_1, y_1), (x_2, y_2), ...(x_n, y_n)\}$ follows the order from one endpoint to the other endpoint
Initialize:
$D = \{\}$
$Q = \{(p_0, p_1)\}$
$p_0$: endpoint, $p_1$: nearest point of $p_0$
 1: while $Z$ is not empty do
 2:     KDTree to find $k$ nearest points of $p_1$:
 3:     $P = \{p_1^0, p_1^1, ...p_1^k\}$
 4:     find $p_1^\kappa \in P$ that maximum the angle of
 5:     vector $p_1 - p_0$ and $p_1 - p_1^\kappa$
 6:     $D \leftarrow D \bigcup \{(p_1, p_1^\kappa)\}$
 7:     $Z \leftarrow Z \setminus \{p_1^\kappa\}$
 8:     $p_0 \leftarrow p_1, p_1 \leftarrow p_1^\kappa$
 9: end while
10: return $D$
```

The order of feature-points encodes the trajectory from one endpoint to the other one. As there is a spring constrain of rope particles in softGym, which means the euclidean distance of two neighboring particles should not exceed 0.013, we only use the orientation of feature-points when we map them to the simulation environment:

$$\text{Slope}\{f_i, f_{i+1}\} \rightarrow \text{Slope}\{p_i, p_{i+1}\} \tag{5}$$

Here f_i is the i^{th} feature-point, p_i is the i^{th} particle in softGym. The crossing has two particles (x_0, y_0, z_0) and (x_1, y_1, z_1). Due to the overlap, $x_0 = x_1$ and $y_0 = y_1$. Therefore, for each input image, we could generate 2^k simulated ropes, k is the number of ground truth crossings.

	Perception				Interaction			
Perception algorithm	Success	Duplication	All cases	Accuracy	Success	Duplication	All cases	Accuracy
Branch-based	486	372	600	81.0%	92	0	100	92%
Learning-based	523	204	600	87.2%	100	0	100	100%

Table 2: Main results

	Train	Test
Accuracy	98.2%	85.4%

Table 3: Score table

4.3 Main Results

The result of our bi-classification CNN is shown in Table 3, we use 4000 real-world sub-images as training and validation data, and 1000 real-world sub-images as test data. The main result is shown in Table 2. The perception algorithms are tested on 600 simulation images containing 1-4 ground truth crossings. Success is all crossings are detected regardless duplication, and if some preliminary crossings belong to the same ground truth crossing, it will be counted as Duplication. The interaction part is tested on 100 simulation cases. Success means the branches are correctly labeled and all optimized crossings are aligned with the ground truth.
The learning based method has higher accuracy and less duplication in both perception and interaction tasks. We believe that after we applied MTCNN for keypoint detection, the performance will be much better. The both improvement of accuracy after interaction suggests that robot interaction is able to assist vision methods to make a better perception.

4.4 Future Work

There are still limitation in our work: the rope only contains crossing but rather knots. Although the perception algorithm has been tested on real world images, due to the limit number of test examples, we still cannot guarantee its generalization ability. Therefore, future works will focus on realizing Fig. 4: training deep networks to encode the perception and interaction together. The networks should be able to predict the coordinates of crossings and their configurations with the feedback from interaction. After getting a desired result, we are able to start the manipulation process by using some state-of-the-art models such as DQN and actor-critic for knot untangling planning.

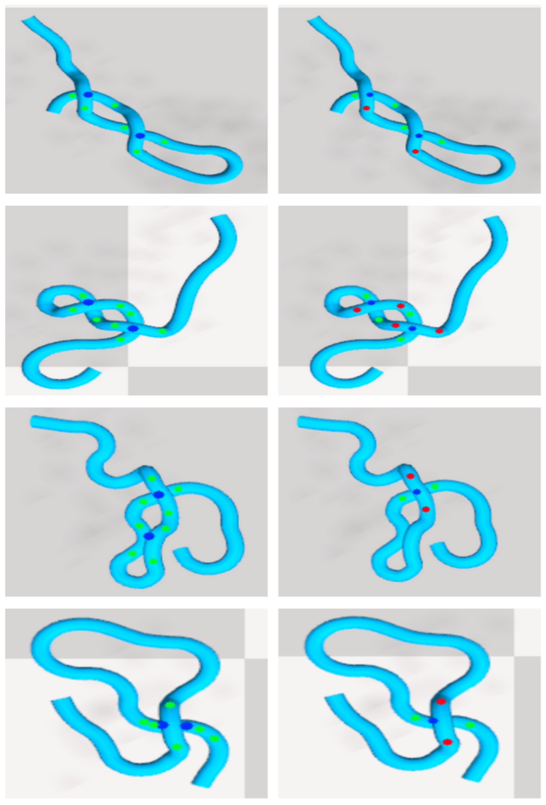

Figure 7: The first column are results of perception(learning based algorithm), and the second column are results of interaction.

5 Appendix

The idea to combine crossing detection and robot interaction is proposed by Professor Shuran Song, and this is one of the candidates of my thesis project. The code in **my_test** folder is all written by myself, and code in other folders is provided by softGym.

References

[1] J. Grannen, Priya Sundaresan, Brijen Thananjeyan, Jeff Ichnowski, A. Balakrishna, Vainavi Viswanath, Michael Laskey, J. Gonzalez, and Ken Goldberg. Learning robot policies for untangling dense knots in linear deformable structures. 2020.

[2] Andrew M. Ladd and Lydia E. Kavraki. Using motion planning for knot untangling. *The Interna-*

tional Journal of Robotics Research, 23(7-8):797–808, 2004.

[3] Xingyu Lin, Yufei Wang, Jake Olkin, and David Held. Softgym: Benchmarking deep reinforcement learning for deformable object manipulation. In *Conference on Robot Learning*, 2020.

[4] Wen Hao Lui and Ashutosh Saxena. Tangled: Learning to untangle ropes with rgb-d perception. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 837–844, 2013.

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

[6] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018.

[7] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.

[8] Yu Song, Kang Yang, Xin Jiang, and Yunhui Liu. Vision based topological state recognition for deformable linear object untangling conducted in unknown background. In *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 790–795, 2019.

[9] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, Oct 2016.