Supplementary Materials for Learning Foresightful Dense Visual Affordance for Deformable Object Manipulation

Ruihai Wu[1,3,4*] **Chuanruo Ning**[2,1*] **Hao Dong**[1,3,4†]
[1]CFCS, School of CS, PKU [2]School of EECS, PKU [3]BAAI
[4]National Key Laboratory for Multimedia Information Processing, School of CS, PKU

Contents

A. Additional Details of Tasks, Settings and Metrics

A.1. Tasks

We select 2 representative tasks from *DeformableRavens* benchmark: **cable-ring** and **cable-ring-notarget**, as well as 2 harder tasks from *SoftGym*: **SpreadCloth** and **RopeConfiguration** (we use the shape 'S' as the target).

- (1) For **cable-ring**, it has a ring-shaped cable with 32 beads. The goal of the robot is to manipulate the cable towards a target zone denoted by a green circular ring in the observation. The maximum convex hull area of the cable-ring and the target cable-ring are the same.
- (2) For **cable-ring-notarget**, the setting is the same as **cable-ring**, except that there is no visible target zone in the observation, so that the goal is to manipulate the cable to a circular ring anywhere on the workbench.
- (3) For **SpreadCloth**, we use a square cloth. The cloth is randomly perturbed to a crumpled state. The goal of the robot is to manipulate the crumpled cloth into flat state.
- (4) For **RopeConfiguration**, the rope is randomly perturbed to a crumpled state. The goal of the robot is to manipulate the rope into the shape of letter 'S'.

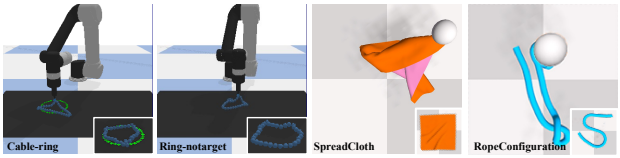

Figure 1. **Demonstrations of the selected tasks.** Each image shows an observation and a successful state (the cropped sub-images to the bottom right).

A.2. Simulation

For *DeformableRavens* benchmark, we use the suite of simulated manipulation tasks using PyBullet [2] physics engine and OpenAI GYM [1] interfaces. For *SoftGym* benchmark, we use Nvidia FleX physics simulator and the Python interface for a better simulation of cloth and rope.

A.3. Metrics

For **cable-ring** and **cable-ring-notarget**, we follow *DeformableRavens* benchmark, when the convex hull area of the ring beads exceeds a thresh β, the manipulation is

*Equal contribution, order determined by coin flip.
†Corresponding author

judged as a success. We set β to be 0.75 for these two tasks, as established in *DeformableRavens*. We use the success rate as manipulation score, which is number of successful manipulation trajectories divided by total number of manipulation trajectories.

For the **SpreadCloth** and **RopeConfiguration**, following *SoftGym* benchmark, we choose the normalized score as manipulation score, which is $\frac{metric_{final}-metric_{initial}}{metric_{goal}-metric_{initial}}$, where $metric_{final}$ means the score of final state, $metric_{initial}$ means the score of initial state and $metric_{goal}$ means the score of target state. Specifically, for **SpreadCloth**, we use the coverage area of cloth as the measurement, and $metric_{goal}$ is 1.00. For **RopeConfiguration**, we use the negative bipartite graph matching distance as the metric, and $metric_{goal}$ is -0.04.

During **testing**, we randomly select 60 (the same number of trials as in *DeformableRavens*) random seeds representing different initial configurations of the objects, conduct experiments and report manipulation score on these different initial states.

B. Additional Details of Experiments

B.1. Data Collection

We collect 5,000 interactions in cable-related tasks, and 40000 interactions in **SpreadCloth** and **RopeConfiguration** for each step. The training of the **SpreadCloth** and **RopeConfiguration** needs more data, for the reason that, the states, kinematics and dynamics of these objects are much more complex.

From a starting state, we collect both successful interaction data using the proposed *Fold to Unfold* data collection method, and failure interaction data using a random policy. Therefore, the trained dense affordance could represent the distribution of diverse results of diverse actions. Each interaction data contains the actions (picking point and placing point) and results after the action (*e.g.* cloth coverage area for **SpreadCloth**).

B.2. Hyper-parameters

We set batch size to be 20, and use Adam Optimizer [3] with 0.0001 as the initial learning rate. During **Integrated Systematic Training (IST)** procedure, we set learning rate to be 0.00005, as the affordance modules have been trained before, and are only adapted and integrated into a system in this procedure.

B.3. Computing Resources

We use TensorFlow as our Deep Learning framework. Each experiment is conducted on an RTX 3090 GPU, and consumes about 20 GB GPU Memory for training. It takes about 12 hours and 6 hours to respectively train the Placing Module and the Picking Module for one step. Besides, the Integrated Systematic Training procedure consumes 6 hours.

C. Additional Details of Network Architectures

The Picking Module and the Placing Module both employ Fully Convolutional Networks (FCNs) with the same structure to extract point-level features. Through the FCNs, the feature of the $W \times H \times C$ dimension input sequentially transforms to $W \times H \times 64$, $W \times H \times 64$, $W/2 \times H/2 \times 128$, $W/4 \times H/4 \times 256$, $W/8 \times H/8 \times 512$, $W/16 \times H/16 \times 512$ (bottleneck of the net work, where global feature is extracted), $W/8 \times H8 \times 512$, $W/8 \times H/8 \times 256$, $W/4 \times H/4 \times 256$, $W/4 \times H/4 \times 128$, $W/2 \times H/2 \times 128$, $W/2 \times H/2 \times 256$, $W \times H \times 256$, $W \times H \times 256$.

Afterwards, the Picking Module uses MLPs with hidden sizes to be (256→256, 256→1) to predict picking affordance, and the Picking Module uses MLPs with hidden sizes to be (1024→256, 256→1) to predict placing affordance. Here, 256 denotes the feature dimension of each (picking or placing) point, and 1024 denotes the dimension of the concatenation of the picking point feature (256), the placing point feature (256), and the global feature (512).

D. Additional Details of Real-world Experiments

D.1. Real-robot Settings

For real-robot experiments, we set up one Franka Panda robot on the workbench, with a RealSense camera mounted on the robot gripper to take observations. We use Robot Operating System (ROS) [4] to control the robot to execute actions.

Additionally, as shown in Figure 2, as the original fingers of Franka Panda is wide and coarse, to ensure that the gripper can pick only one layer of cloth instead of two layers at a time, we design two fine-grained fingers mounted on the fingertips of the original fingers.

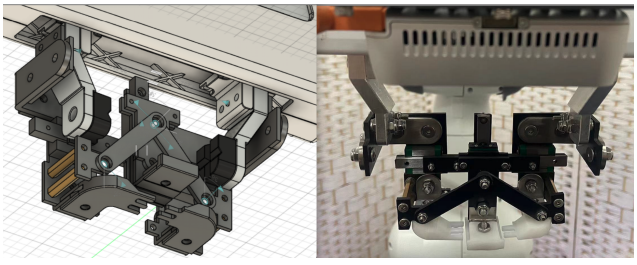

Figure 2. **Our Designed Fine-grained Fingers** to better pick deformable objects.

D.2. Real-world Data Collection and Fine-tuning

As the configurations, kinematics and dynamics of deformable objects (like cloth and ropes) in the real world are different from those in simulation, we fine-tune the trained-in-simulation affordance using real-world collected interactions.

Specifically, following Section 4.5 (***Fold to Unfold*: Efficient Multi-stage Data Collection for Learning Foresightful Affordance**) in the main paper, we collect real-world interactions using the *Fold-to-Unfold* method in different stages (demonstrations shown in Figure 3). For fine-tuning, we tune the learned picking and placing affordance stage-by-stage using the above real-world collected data.

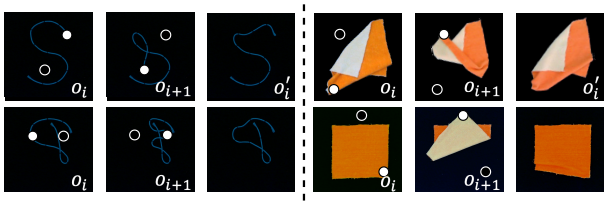

Figure 3. **Demonstrations of real-world collected data.** State o_{i+1} shows the starting state and state o_i' show the ending state of interaction data for training.

D.3. Metrics

For **SpreadCloth**, we use the same metric as in A.3, as it is easy to compute the coverage area of the cloth in the real world. For **RopeConfiguration**, we mark a black dot on the rope every 10cm, and compute the distances between these black dots and their ideal locations as the bipartite graph matching distance for further evaluation.

D.4. Video Records of Real-world Manipulations

Please see the **experiment part of the supplementary video** for video records of real-world manipulations for both **SpreadCloth** and **RopeConfiguration** tasks.

E. Assets and Code

E.1. Assets

We use the cable assets in *DeformableRavens* benchmark as well as cloth and rope assets in *SoftGym* benchmark, following their licenses. Our proposed assets with novel configurations can be generated using our code.

E.2. Code

Code for **SpreadCloth** and **RopeConfiguration** experiments is attached with this supplementary. The *README.md* file illustrates the usage of our code.

References

[1] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016. 1

[2] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. 2016. 1

[3] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *The 3rd International Conference for Learning Representations*, 2015. 2

[4] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, number 3.2, page 5. Kobe, Japan, 2009. 2