Supplementary Material for

Perceiving Unseen 3D Objects by Poking the Objects

I. Poking policy

Algorithm 1: Poking algorithm.

Input: Object point cloud P_o; start distance d_s; poking distance d_p; poking iterations N_p

Output: start point s; end point e

1 Fit oriented bounding box $B = (b_1, b_2, b_3, b_4)$ in bird-eye-view for P_o

2 **for** n_p = *1 to* N_p **do**

3 sort B by $\{b_{ix}\}$ increasingly

4 **if** $|b_1 - b_2| < |b_1 - b_3|$ **then**

5 $p_1, p_2 \leftarrow b_1, b_3$

6 **else**

7 $p_1, p_2 \leftarrow b_1, b_2$

8 $r \leftarrow 75°$

9 **if** $p_{1y} < p_{2y}$ **then**

10 $p_1, p_2 \leftarrow p_2, p_1$

11 $r \leftarrow 105°$

12 $d \leftarrow \begin{bmatrix} \cos r & -\sin r \\ \sin r & \cos r \end{bmatrix} \begin{bmatrix} p_{2x} - p_{1x} \\ p_{2y} - p_{1y} \end{bmatrix}$

13 $p \leftarrow p_1 + (p_2 - p_1)/3$

14 $s \leftarrow p - d \cdot d_s$

15 $e \leftarrow p + d \cdot d_p$

16 Poke from s to e

17 Update B

Fig. 1. **A scene with several unseen objects.** A fixed RGB-D camera is used to capture data, and a UR5 robot with a cylinder is used to poke the objects.

The coordinate systems are defined as follows: the z axes of the world and the robot are perpendicular to the plane and the y axes are perpendicular to the line connecting the camera and the center of the world coordinate system. The line p_1p_2 represents one of the longer sides of the bounding box B that rotates the object in a clockwise direction upon poking. The angle r between the poking direction and the longer side of the bounding box is defined in Algorithm 1 (lines 8 and 11). Instead of poking in a perpendicular direction ($r = 90°$), we empirically add or subtract $15°$ to achieve more translation of the object. The z values of the start and end points are determined by keeping the distance between the end of the gripper and the plane constant. Before execution, the computed start and end points are evaluated to avoid any unexpected collisions between the gripper and the environment. After each iteration of poking, the oriented bounding box B is updated through point cloud registration on the foreground, which is defined as the set of pixels with optical flow norms greater than a pre-defined threshold. The point cloud registration is achieved through scene flow estimation and refined by the Iterative Closest Point (ICP) method. The scene flow is obtained from the back-projected optical flow, which is estimated using a trained optical flow estimation network [1]. The order of poking is determined by sorting the distances between the center of the object proposal and the right-down corner of the plane in increasing order.

II. Data collection

The real-world setup is composed of a UR5 robot with a cylinder pusher tool and an RGB-D Kinect camera visualized in Fig. 1. The relative pose between the camera and the base of the robot arm T_{ca} is computed as follows:

$$T_{ca} = T_{gc}^{-1} T_{ga}, \tag{1}$$

where T_{gc} is the relative pose between the gripper and the camera and T_{ga} is the relative pose between the gripper and the base of the robot arm. In practice, T_{ga} is read from the Robot Operating System (ROS). T_{gc} is obtained by aligning the model of the gripper and an RGB-D image. The alignment is achieved via differentiable rendering. Representing the 3D model of the gripper as G and an RGB-D image with the manually-annotated segmentation mask of the gripper as M, T_{gc} is obtained by optimizing the following objective function:

$$\mathcal{L}(R, t) = \sum (\pi(RG + t) - M)^2, \tag{2}$$

where π is a differentiable neural renderer [2] that produce the mask of the gripper.

III. Supplementary video

The supplementary video provides a demonstration of the real-world robotics grasping task. It starts by describing the task, followed by the recorded video of the poking process, and ends with the application of object pose estimation and robotics grasping to demonstrate the practicality of our method in real-world scenarios.

References

[1] Z. Teed and J. Deng, "Raft: Recurrent all-pairs field transforms for optical flow," in *European conference on computer vision*. Springer, 2020, pp. 402–419.

[2] H. Kato, Y. Ushiku, and T. Harada, "Neural 3d mesh renderer," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3907–3916.