APPENDIX

A. Additional Details of Experimental Setup

Experimental setup. We design experiments to fully test the efficacy of our method by providing the robot with videos captured in everyday scenarios, which naturally encompass visual backgrounds and camera setups that are different from the one for the robot. Specifically, we record an RGB-D video of a person performing each of the seven tasks in everyday scenarios, such as an office or a kitchen. We use an iPad for recording, which comes with a TrueDepth Camera, and we fix it on a camera stand. The videos can be found in the supplementary materials. During test time, the robot receives visual data through a single RGB-D camera, Intel Realsense435, and performs manipulation in its workstation to evaluate policies. We use the 7DoF Franka Emika Panda robot for all the experiments.

Evaluation protocol. As we describe in the experimental setup, the videos naturally include various visual backgrounds and camera perspectives that are significantly different from the robot workspace. Therefore, we only intentionally vary two dimensions before evaluating each trial of robot execution, namely the spatial layouts and the new object instances. Furthermore, the new object generalizations are included in the tasks `Mug-on-coaster` and `Chips-on-plate` as mugs and chip bags have many similar instances. As for the other three tasks, there are no novel objects involved, but we extensively vary the spatial layouts of task-relevant objects for evaluation. The policy performance of a task is the averaged success rates over 15 real-world trials. Aside from the success rates, we also group the failed executions into three types: *Missed tracking* of objects due to failure of the vision models, *Missed grasping* of objects during execution, and *Unsatisifed contacts* where the target object configurations are not achieved for reasons other than the previous two failure types.

B. Additional Technical Details

Data Structure of an OOG. For easy reproducibility of the proposed method, we present a table that explains the data structure of an OOG.

Changepoint detections. We use changepoint detection to identify changes in velocity statistics of TAP keypoints. Specifically, we use a kernel-based changepoint detection method and choose radial basis function [23]. The implementation of this function is directly based on an existing library Ruptures [69].

Plane estimation. In Section III-B, we mentioned using the prior knowledge of tabletop manipulation scenarios and transforming the point clouds by estimating the table plane. Here, we explain how the plane estimation is computed. Concretely, we rely on the plane estimation function from Open3D [70], which gives an equation in the form of $ax + by + cz = d$. From this estimated plane equation, we can infer a normal vector of the estimated table plane, (a, b, c), in the camera coordinate frame. Then, we align this plane with xy plane in the world coordinate frame, where we compute a transformation matrix that displaces the normal vector (a, b, c) to the normalized vector $(0, 0, 1)$ along the z-axis of the world coordinate frame. This transformation matrix is used to transform point clouds in every frame so that the plane of the table always aligns with the xy plane of the world coordinate.

Object localization at test time. When we localize objects at test time, there could be some false positive segmentation of distracting objects. Such vision failures will prevent the robot policy from successfully executing actions. To exclude such false positive object segmentaiton, we use Segmentation Correspondence Model (SCM) from GROOT [11], where SCM filters out the false positive segmentation of the objects by computing the affinity scores between masks using DINOv2 features.

Global registration. In this paper, we use global registration to compute the transformation between observed object point clouds from videos and those from rollout settings. We implement this part using a RANSAC-based registration function from Open3D [70]. Specifically, given two object point clouds, we first compute their features using Fast-Point Feature Histograms (FPFH) [71], and then perform a global RANSAC registration on the FPFH features of the point clouds [27].

Implementation of SE(3) optimization. We parameterize each homogeneous matrix T_i into a translation variable and a rotation variable and randomly initialize each variable using the normal distribution. We choose quaternions as the representation for rotation variables, and we normalize the randomly initialized vectors for rotation so that they remain unit quaternions. With such parameterization, we optimize the SE(3) end-effector trajectories $T_0, T_1, \ldots, T_{t_{l+1}-t_l}$ over the Objective (1). However, jointly optimizing both translation and rotation from scratch typically results in trivial solutions, where the rotation variables do not change much from the initialization due to the vanishing gradients. To avoid trivial solutions, we implement a two-stage process. In the first stage, we only optimize the rotation variables with 200 gradient steps. Then, the optimization proceeds to the second stage, where we optimize both the rotation and translation variables for another 200 gradient steps. In this case, we prevent the optimization process from getting stuck in trivial solutions for rotation variables. We implement the optimization process using Lietorch [72].

C. System Setup

Details of camera observations. As mentioned in Section IV, we use an iPad with a TrueDepth camera for collecting human video demonstrations. We use an iOS app, Record3D, that allows us to access the depth images from the TrueDepth camera. We record RGB and depth image frames in sizes 1920×1080 and 640×480, respectively. To align the RGB images with the depth data, we resize the RGB frames to the size 640×480. The app also automatically records the camera intrinsics of the iPhone camera so that the back-projection of point clouds is made possible.

Node/Edge	Type	Attributes
$\mathcal{G}.vo_i$	Object Node	3D point cloud of an object.
$\mathcal{G}.vh$	Hand Node	Hand mesh and locations of the thumb and index finger.
$\mathcal{G}.vp_{ij}$	Point Node	A trajectory of a TAP keypoint between two keyframes, recorded in xyz positions.
$\mathcal{G}.eo_{ik}$	Object-Object Edge	A binary value of contact or not.
$\mathcal{G}.eh_i$	Object-Hand Edge	A binary value of contact or not.
$\mathcal{G}.ep_{ij}$	Object-Point Edge	The presence of an edge represents the belonging relation, and no specific feature is attached.

TABLE I: Data Structure of an OOG. For a given OOG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, it has $\mathcal{V} = \{\mathcal{G}.vo_i\} \cup \{\mathcal{G}.vh\} \cup \{\mathcal{G}.vp_{ij}\}$, and $\mathcal{E} = \{\mathcal{G}.eo_{ik}\} \cup \{\mathcal{G}.eh_i\} \cup \{\mathcal{G}.ep_{ij}\}$.

To stream images at test time, we use an Intel Realsense D435i. In our robot experiments, we use RGB and depth images in the size 640×480 or 1280×720 in varied scenarios, all covered in our evaluations. Evaluating on different image sizes showcases that our method is not tailored to specific camera configurations, supporting the wide applicability of constructed policy.

Implementation of real robot control. In our evaluation, we reset the robot to a default joint position before object interaction every time. Then we use a reaching primitive for the robot to reach the interaction points. Resetting to the default joint position enables an unoccluded observation of task-relevant objects at the start of each decision-making step. Note that the execution of object interaction does not necessarily require resetting. To command the robot to interact with objects, we convert the optimized SE(3) action sequence to a sequence of joint configurations using inverse kinematics and control the robot using joint impedance control. We use the implementation of Deoxys [5] for the joint impedance controller that operates at 500 Hz. To avoid abrupt motion and make sure the actions are smooth, we further interpolate the joint sequence from the result of inverse kinematics. Specifically, we choose the interpolation so that the maximal displacement for each joint does not exceed 0.5 radian between two adjacent waypoints.

D. Task descriptions and Success conditions

Task descriptions. 1) `Mug-on-coaster`: placing a mug on the coaster; 2) `Simple-boat-assembly`: putting a small red block on a toy boat; 3) `Chips-on-plate`: placing a bag of chips on the plate; 4) `Succulents-in-llama-vase`: inserting succulents into the llama vase; 5) `Rearrange-mug-box`: placing a mug on a coaster and placing a cream cheese box on a plate consecutively; 6) `Complex-boat-assembly`: placing both a small red block and a chimney-like part on top of a boat. 7) `Prepare-breakfast`: placing a mug on a coaster and putting a food box and can on the plate.

Success conditions. We describe the success conditions for each of the tasks in detail:

- `Mug-on-coaster`: A mug is placed upright on the coaster.
- `Simple-boat-assembly`: A red block is placed in the slot closest to the back of the boat. The block needs to be upright in the slot.
- `Chips-on-plate`: A bag of chips is placed on the plate, and the bag does not touch the table.
- `Succulents-in-llama-vase`: A pot of succulents is inserted into a white vase in the shape of a llama.
- `Rearrange-mug-box`: The mug is placed upright on the coaster, and the cream cheese box is placed on the plate.
- `Complex-boat-assembly`: The chimney-like part is placed in the slot closest to the front of the boat. The red block is placed in the slot closest to the back of the boat. Both blocks need to be upright in the slots.
- `Prepare-breakfast`: The mug is placed on top of a coaster, the cream cheese box is placed in the large area of the plate, and the food can is placed on the small area as shown in the video demonstration.

In practice, we record the success and failure of a rollout as follows: If the program in ORION policy returns true when matching the observed state with the final OOG from a plan, we mark a trial as success as long as we observe that the object state indeed satisfies the success condition of a task as described above. Otherwise, if the robot generates dangerous actions (bumping into the table) or does not achieve the desired subgoal after executing the computed trajectory, we consider the rollout as a failure and we manually record the failure.

E. Additional Details on Experiments

Diverse video recordings used in the ablation study. Figure 6 shows the three videos taken in very different scenarios: kitchen, office, and outdoor. The video taken in kitchen scenario is used in the major quantitative evaluation, termed "Original setting". The other two settings are termed "Diverse setting 1" and "Diverse setting 2." We conduct an ablation study where we compare policies imitated from these three videos, which inherently involve varied visual scenes, camera perspectives. The result of the ablation study is shown in Figure 5.

F. Limitations

We consider the task goals to be described by contact states so that we naturally avoid the ambiguities introduced when considering spatial relations, such as placing items next

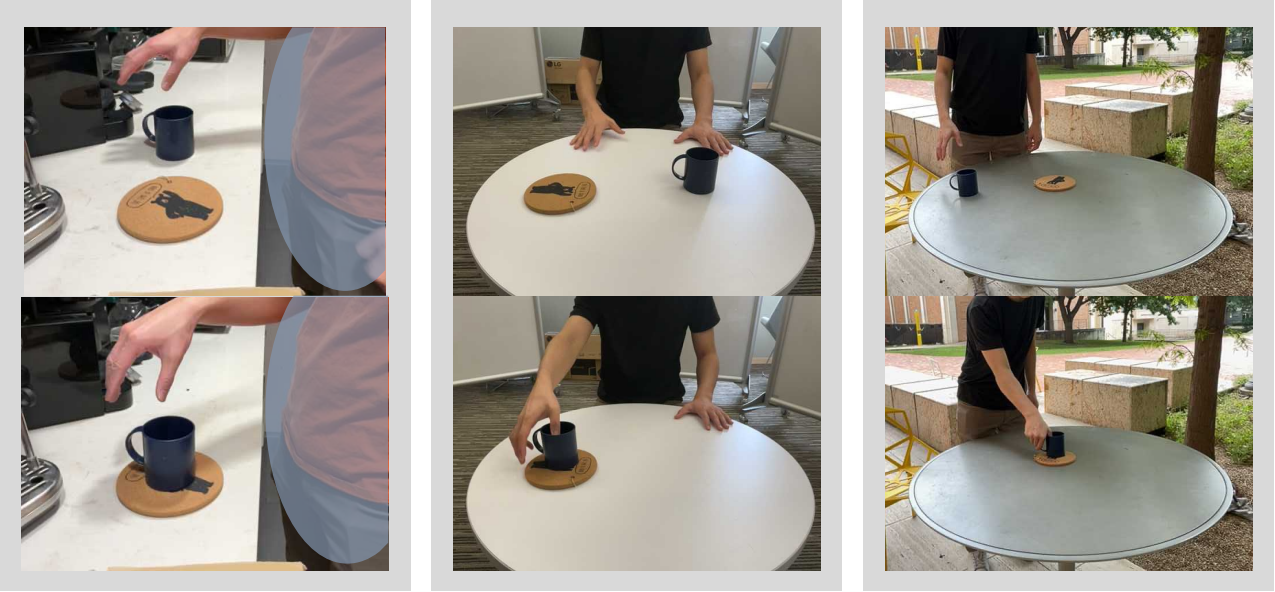

Fig. 6: This figure visualizes the initial and final frames of the three videos of the same task `Mug-on-coaster`.

to an object. How to infer human intentions while clearing the inherent ambiguities in videos is a future direction to explore. We have also assumed the videos are captured in RGB-D using a stationary camera. In reality, most videos in-the-wild are taken in RGB images with moving camera views. A future direction is to build a model that reconstruct the dynamic scenes from in-the-wild video data, where the model needs to reconstruct the geometry of both static and dynamic objects.

This work uses the robot with a parallel-jaw gripper, therefore we have focused on translating the thumb and index finger positions of the human hand to the robot gripper and assume the thumb and index finger are involved in hand-object interaction. However, OOG representation is general enough to capture full hand poses and can be used in dexterous manipulation, which is shown by our follow-up work [68].

While our OOG representation is general enough to capture full hand poses, we only have hardware access to the Panda robot, which prevents us from showcasing dexterous manipulation tasks. A future promising direction is to extend ORION to a robot hardware with a dexterous hand so that the full hand pose can be exploited and unlock the dexterous manipulation tasks.

In ORION, a robot imitates the human actions at the kinematic level to tackle the problem of "open-world imitation from observation." Consequently, ORION does not consider tasks that require force sensing such as screwing. Another future direction that ORION unlocks is how to allow robot to imitate forceful manipulation behaviors [67]. Another future extension is to extend the action synthesis in ORION to handle real-time motion adjustment, an improvement that will unlock non-prehensile behaviors such as planar pushing.

Furthermore, ORION establishes the correspondence between objects from demonstration and rollout using global registration of the point clouds. Such correspondence relies solely on the geometry of objects, which may suffer from ambiguities when the object shapes are symmetric and the correspondence relies on the texture information. A future direction is to incorporate both semantic and geometric information of the objects to establish object correspondence.