FUNCTO: Function-Centric One-Shot Imitation Learning for Tool Manipulation

Chao Tang[12], Anxing Xiao[2], Yuhong Deng[2], Tianrun Hu[3], Wenlong Dong[1],
Hanbo Zhang[2], David Hsu[23], and Hong Zhang[1]

[1]Southern University of Science and Technology [2]School of Computing, National University of Singapore
[3]Smart Systems Institute, National University of Singapore

Abstract—Learning tool use from a single human demonstration video offers a highly intuitive and efficient approach to robot teaching. While humans can effortlessly generalize a demonstrated tool manipulation skill to diverse tools that support the same function (e.g., pouring with a mug versus a teapot), current one-shot imitation learning (OSIL) methods struggle to achieve this. A key challenge lies in establishing functional correspondences between demonstration and test tools, considering significant geometric variations among tools with the same function (i.e., intra-function variations). To address this challenge, we propose FUNCTO (Function-Centric OSIL for Tool Manipulation), an OSIL method that establishes function-centric correspondences with a 3D functional keypoint representation, enabling robots to generalize tool manipulation skills from a single human demonstration video to novel tools with the same function despite significant intra-function variations. With this formulation, we factorize FUNCTO into three stages: (1) functional keypoint extraction, (2) function-centric correspondence establishment, and (3) functional keypoint-based action planning. We evaluate FUNCTO against exiting modular OSIL methods and end-to-end behavioral cloning methods through real-robot experiments on diverse tool manipulation tasks. The results demonstrate the superiority of FUNCTO when generalizing to novel tools with intra-function geometric variations. More details are available at https://sites.google.com/view/functo.

I. INTRODUCTION

The ability to use tools has long been recognized as a hallmark of human intelligence [1]. Endowing robots with the same capability holds the promise of unlocking a wide range of downstream tasks and applications [2, 3, 4]. As a step towards this goal, we tackle the problem of one-shot imitation learning (OSIL) for tool manipulation, which involves teaching robots a tool manipulation skill with a single human demonstration video. The objective is to develop an OSIL method capable of *generalizing the demonstrated tool manipulation skill to novel tools with the same function*. Here, "same function" refers to the robot imitating the demonstrated tool manipulation behavior to accomplish functionally equivalent tasks.

While humans can effortlessly achieve the objective described above, it remains a non-trivial challenge for robots due to significant geometric variations (e.g., shape, size, topology) among tools supporting the same function (i.e., intra-function variations). As shown in Figure 1, although both the mug and the teapot support the same function of pouring, their geometries differ significantly (e.g., the teapot features a long neck and a handle positioned on top of its body). Apparently, to successfully apply OSIL in this case, a key challenge lies in
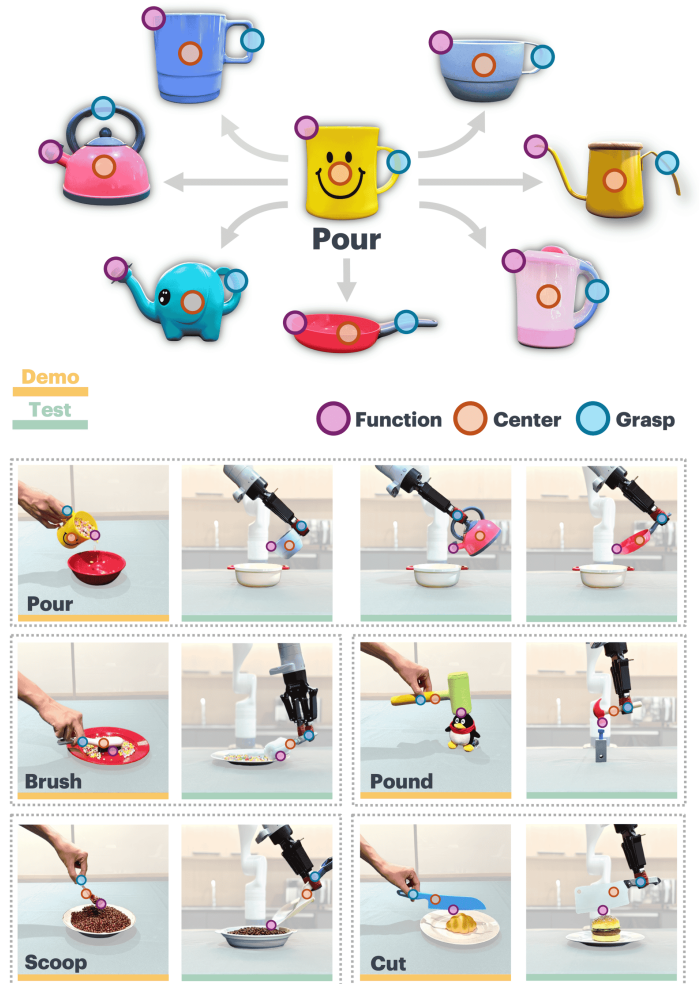

Fig. 1. FUNCTO establishes functional correspondences between demonstration and test tools using 3D functional keypoints. With a single human demonstration video, FUNCTO generalizes the demonstrated tool manipulation skill to novel tools, even with significant intra-function geometric variations.

establishing functional correspondences between demonstration and test tools. Previous OSIL methods [4, 5, 6, 7, 8, 9, 10] assume that tools supporting the same function share highly similar shapes or appearances. Based on this assumption, they establish "shallow" correspondences through techniques such as keypoint-based pose estimation [4, 5, 6], global point set registration [7, 8], shape warping [9], and invariant region matching [10] to align geometrically or visually similar tools. However, this assumption often fails in practice due to large intra-function variations. As a result, previous OSIL methods

exhibit limited generalization to novel tools. This limitation motivates us to ask: What remains invariant among tools with the same function despite significant intra-function variations? Pioneering studies in cognitive anthropology [1] reveal that humans exhibit highly consistent *behavioral patterns* when using different tools serving the same purpose. For instance, the behavioral pattern of pouring involves approaching the tool (e.g., mug), grasping it, and directing its spout towards the target object (e.g., bowl). This spatiotemporal pattern remains invariant across tools (e.g., mug, teapot, saucepan) with the same function of pouring.

Inspired by this observation, we propose FUNCTO (Function-Centric OSIL for Tool Manipulation), which emphasizes the functional aspects of tool correspondences over geometric or visual similarities as in previous works. FUNCTO achieves this by establishing function-centric correspondences between demonstration and test tools using a 3D functional keypoint representation. The 3D functional keypoint representation consists of a function point, where the tool interacts with the target (e.g., the spout of a mug); a grasp point, where the tool is held (e.g., the handle of a mug); and a center point, which is the tool's 3D center. By focusing on these three functional keypoints, FUNCTO captures the invariant spatiotemporal pattern among tools supporting the same function while ignoring function-irrelevant geometric details. Specifically, FUNCTO is factorized into three stages: (1) Functional keypoint extraction, which detects functional keypoints and tracks their motions from the human demonstration video; (2) Function-centric correspondence establishment, which transfers functional keypoints from the demonstration tool to the test tool and establishes function-centric correspondences using geometric constraints on the functional keypoints; (3) Functional keypoint-based action planning, which uses the demonstration and test functional keypoints to generate a robot end-effector trajectory for task execution.

We evaluate FUNCTO against existing OSIL methods and behavioral cloning (BC) methods through extensive real-robot experiments on diverse tool manipulation tasks. Leveraging the proposed function-centric approach with 3D functional keypoints, FUNCTO addresses the limitations of previous works that rely solely on geometric or visual similarities and achieves better generalization to novel tools with the same function despite significant intra-function variations.

Contribution. The main contribution of this work is a novel formulation of function-centric correspondence using a 3D functional keypoint representation for tool manipulation. This enables robots to generalize tool manipulation skills from a single human demonstration video to novel tools with the same function despite significant intra-function variations.

II. RELATED WORK

A. One-Shot Imitation Learning

OSIL has been explored in various domains, such as image recognition [11, 12], generative models [13], and reinforcement learning [14]. The objective is to generalize the demon-

strated behavior to novel instances or variations of the task with minimal prior knowledge or additional training.

In robotics, early OSIL works [3, 15, 16] propose to leverage prior knowledge through meta-learning across a diverse set of robot tasks or skills. Following a "pre-training and adapting" strategy, Wen et al. [17, 18] learn a category-level canonical representation during the pre-training stage and adapt it to new instances at inference. Similarly, Zhang et al. [10] conduct in-domain pre-training of a graph-based invariant region matching network and generalize to geometrically similar tools with a single demonstration. However, these methods face two major limitations: (1) they require in-domain pre-training, and (2) their generalization is restricted to geometrically or visually similar tools, struggling to handle out-of-domain tools with significant intra-function variations. Meanwhile, there has been a growing trend of using behavioral cloning (BC) models [2, 19, 20], pre-trained on massive expert demonstrations, to generalize to unseen tools and configurations. We will later experimentally show that by clearly articulating the functional correspondences between tools, FUNCTO achieves better generalization performance compared to state-of-the-art BC methods, even with significantly less data.

Similar to our setup, [5] utilizes a transformer-based local feature matching model to compute the relative pose transformations between demonstration and test tools. Meanwhile, [4] and [6] leverage the off-the-shelf vision foundation model DINO [21] to establish semantic correspondences between visually similar tools. In parallel, [7] and [8] employ global point set registration [22] to align demonstration and test tools. Similarly, [9] adopts shape warping [23] to correspond instances within the same category. Nevertheless, these methods assume that demonstration and test tools share highly similar shapes or appearances, limiting their generalization to novel tools with large geometric variations. In contrast, FUNCTO can handle significant intra-function variations to enable generalization to novel tools.

B. Keypoint Representation for Tool Manipulation

Keypoint representation has been extensively studied in tool manipulation [24, 25, 26, 27, 28, 29, 30, 31], as it provides a compact and expressive way of encoding object information in terms of both semantics and actionability. For instance, KETO [26] introduces a task-specific keypoint generator trained with self-supervision for planar tool manipulation. KPAM [24] uses 3D semantic keypoints as the tool representation to accomplish category-level manipulation tasks. Similarly, K-VIL [30, 31] leverages a categorical correspondence model [32] to extract keypoint-based geometric constraints from one or few-shot human demonstrations.

More recent works leverage foundation models to predict semantic keypoints in zero-shot and generate corresponding tool manipulation motions. MOKA [28] generates planar manipulation motions via mark-based visual prompting [33]. While FUNCTO also employs a similar technique for functional keypoint detection, it can predict tool trajectories in 3D, enabling it to handle more complex tool manipulation

tasks. ReKep [29] proposes to represent a manipulation task as a list of task-specific keypoint constraints and predict semantic keypoints in a zero-shot manner. However, these constraints require significant manual effort, and the zero-shot keypoint extraction strategy is highly error-prone. In contrast, FUNCTO effortlessly extracts tool manipulation constraints and functional keypoints from human demonstration videos and does not require any object/task-specific knowledge to define the constraints. Furthermore, compared to the zero-shot keypoint proposal as in [28, 29], human demonstrations provide valuable cues for keypoint localization. Leveraging these cues enhances task performance, as demonstrated in the experiment section.

C. Visual Correspondence in Robotics

The ability to establish correspondences [34] between seen and unseen scenarios is essential for robots to generalize. Techniques from the computer vision community, such as pose estimation [35], optical flow estimation [36], and point tracking [37], have been widely adopted in robotic tasks and applications. In robotic manipulation, DON [32] learns dense visual correspondences with self-supervision for transferring grasps across visually similar object instances. Building on this concept, TransGrasp [38] adopts Deformed Implicit Field [39] to build shape correspondences within the same object category for grasp transfer. More recently, NDF [40] proposes a neural implicit representation to learn categorical descriptors from few-shot demonstrations. While these approaches focus on building visual correspondences within the same category, FUNCTO extends this capability to establishing functional correspondences, even in the presence of significant intra-function variations.

FUNCTO is also closely related to affordance theory [41] and the functional correspondence problem [42]. A common principle shared by FUNCTO and these works is that correspondences should extend beyond geometric or visual similarity to incorporate functional relevance.

III. Problem Formulation

We consider the problem of enabling the robot to imitate the tool manipulation behavior demonstrated in a single human video to accomplish functionally equivalent tasks using novel tools with the same function. Specifically, each task involves a robot grasping a tool (object) with a parallel-jaw gripper to interact with a target (object) in a tabletop environment. The task is defined by a list of spatiotemporal constraints between the tool and the target.

During the demonstration phase, a human performs a tool manipulation task, recording a sequence of RGB-D images, $\mathcal{V}_H = \{I_t\}_{t=0}^{N-1}$, with a stationary camera, where N denotes a finite task horizon. The sequence $\mathcal{V}_H$ is paired with a natural language task description l_H (e.g., *"use the mug to pour contents into the bowl"*) that specifies three elements: a tool (e.g., *mug*), a target (e.g., *bowl*), and a function (e.g., *pour*). During inference, given a robot observation o_R and a corresponding task description l_R, the objective is to develop

an OSIL policy π, mapping o_R and l_R to a robot end-effector trajectory $\tau_R = \{a_t\}_{t=0}^{N-1}$ that maximizes the likelihood of task success. Here, $a_t = (R_t, T_t) \in \text{SE}(3)$ represents the 6-DoF end-effector pose at timestep t, where $R_t \in \text{SO}(3)$ and $T_t \in \mathbb{R}^3$ denote 3D orientation and translation, respectively.

Assumptions. During the implementation, we have made the following assumptions: (1) Visual observations are single-view and do not contain any action annotations. (2) The robot has no object/task-specific prior knowledge, such as 3D object models or manual task constraints, but has access to commonsense knowledge embedded in foundation models. (3) No in-domain pre-training is conducted. (4) Tools are modeled as rigid objects and can be manipulated by the designated gripper.

IV. FUNCTO

A. Overview

In this section, we describe FUNCTO, a function-centric OSIL framework for tool manipulation. FUNCTO consists of three stages: (1) functional keypoint extraction, (2) function-centric correspondence establishment, and (3) functional keypoint-based action planning. An overview of the proposed framework is presented in Figure 2. Specifically, in the first stage (Section IV-B), FUNCTO detects 3D functional keypoints K_H and track their motions $\{K_H^t\}_{t=0}^{N-1}$ from $\mathcal{V}_H$. In the second stage (Section IV-C), K_H are transferred from the demonstration tool to the test tool to obtain test functional keypoints K_R. Subsequently, FUNCTO establishes function-centric correspondences with K_H and K_R. In the final stage (Section IV-D), FUNCTO generates a robot end-effector trajectory τ_R, using the reference $\{K_H^t\}_{t=0}^{N-1}$ and the established function-centric correspondences, for execution.

B. Functional Keypoint Extraction

Demonstration Tool Tracking. FUNCTO starts by localizing and segmenting the tool and the target in the first frame of $\mathcal{V}_H$ (i.e., I_0) using Grounding-SAM [43]. Subsequently, N_k keypoints are uniformly sampled within the tool mask. We use Cotracker [37] to capture their 3D motions (using the depth information) for the rest of $\mathcal{V}_H$, yielding their 3D trajectories in the camera frame. To ensure the extracted motion is independent of the absolute locations of the tool and target, FUNCTO transforms the 3D keypoint trajectories from the camera frame to the target (object) frame by estimating the target object pose in the camera frame. The origin of the target frame, denoted as the target point p_{target}, is defined as the 3D center of the target. Finally, FUNCTO utilizes rigid body transformation [44] to calculate the relative transformations of the tool between consecutive timesteps based on the 3D keypoint trajectories. Throughout this section, all 3D elements are represented in the target frame unless otherwise specified.

Keyframe Discovery. Due to the partial observability of functional keypoints in the human demonstration video, FUNCTO discovers keyframes where functional keypoints on the demonstration tool can be effectively identified.
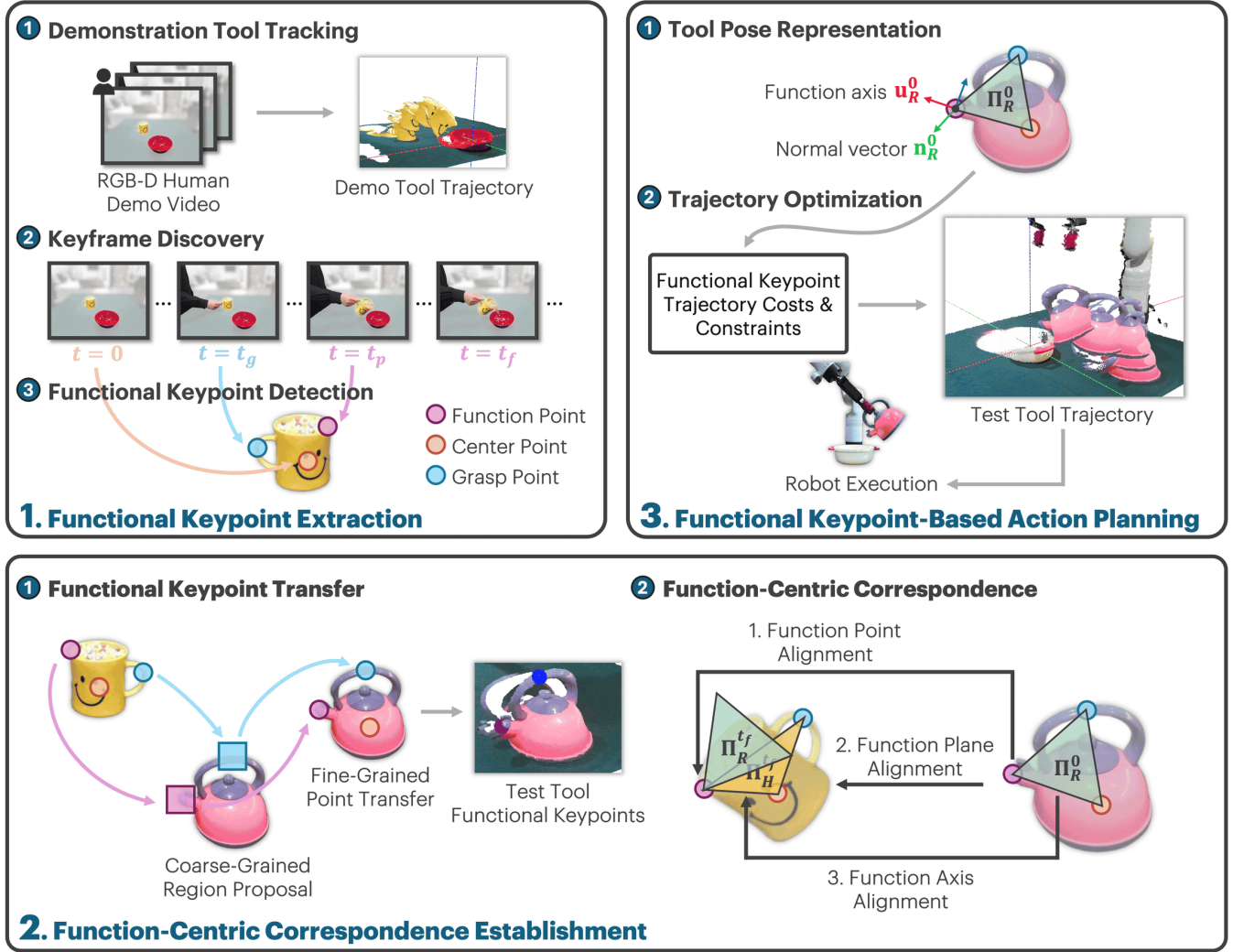
Fig. 2. An overview of the FUNCTO framework. The pipeline consists of three stages: (1) Functional keypoint extraction, where functional keypoints and their trajectories are extracted from the human demonstration video; (2) Function-centric correspondence establishment, where function-centric correspondences between demonstration and test tools are established using geometric constraints on the functional keypoints; and (3) Functional keypoint-based action planning, where the test tool trajectory is synthesized and executed to accomplish a functionally equivalent task.

Specifically, FUNCTO discovers four keyframes: (1) the initial keyframe I_0 ($t = 0$), where the tool is in its initial state; (2) the grasping keyframe I_g ($t = t_g$), where the hand grasps the tool; (3) the function keyframe I_f ($t = t_f$), where the interaction between the tool and target starts; (4) the pre-function keyframe I_p ($t = t_p$), where the interaction is about to start while both the tool and target remain clearly visible. These keyframes satisfy $0 < t_g < t_p < t_f < N - 1$. Note that I_p is essential for function point detection before heavy occlusion associated with the interaction occurs. For detecting I_g, we use an off-the-shelf hand-object detector from [45]. Following [7], an unsupervised change point detection method [46] is employed to identify I_f based on velocity statistics derived from the 3D keypoint trajectories. Lastly, we backtrack through $\mathcal{V}_H$ to locate a preceding frame (i.e., I_p) where the occlusion between the tool and the target, measured by IoU, is below a pre-defined threshold. Examples of discovered keyframes are presented in Figure 2 (stage 1).

Functional Keypoint Detection. Once the keyframes are

identified, FUNCTO proceeds to detect 3D functional keypoints and track their motions, denoted as $\{K_H^t\}_{t=0}^{N-1} = \{[p_{\text{func}}^t, p_{\text{grasp}}^t, p_{\text{center}}^t]\}_{t=0}^{N-1}$. p_{func}^t, p_{grasp}^t, and p_{center}^t represent the 3D locations of the function point, grasp point, and center point at timestep t, respectively, with $p \in \mathbb{R}^3$.

Three functional keypoints are detected in the respective keyframes. The grasp point is determined by computing the intersection between the hand mask and the tool mask in I_g, while the center point is computed as the 3D center of the tool in I_0. Detecting the function point is non-trivial, as there may be no physical contact between the tool and the target (e.g., pouring), and it also requires commonsense knowledge about tool usage. Therefore, FUNCTO leverages the multi-modal reasoning capabilities of VLMs and employs mark-based visual prompting [33]. The visual prompting process involves two steps: (1) task-agnostic meta-prompt definition and (2) task-specific prompting. In the first step, we provide a meta-prompt with task-agnostic context information to the VLM, including the definition of the function point, the expected behavior of the VLM, and the desired response format. In the
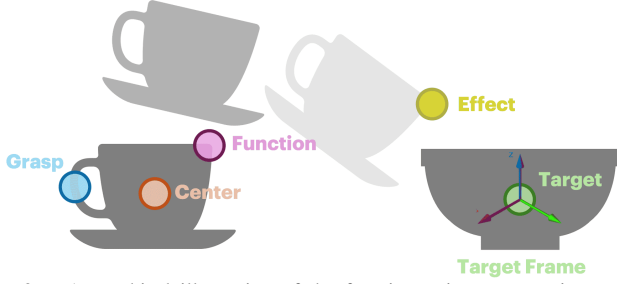
Fig. 3. A graphical illustration of the function point, grasp point, center point, effect point, target point, and target frame.

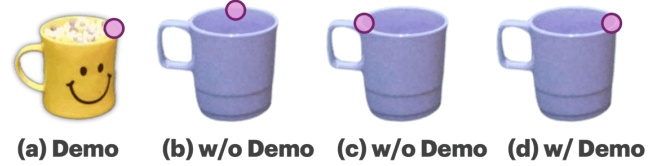(a) Demo   (b) w/o Demo   (c) w/o Demo   (d) w/ Demo

Fig. 4. Qualitative results of function point transfer. (a) shows the function point extracted from the human demonstration. Function points in (b) and (c) are proposed by the VLM in a zero-shot manner. (d) shows the transferred function point using (a) as a reference.

second step, we sample and annotate N_c candidate points on the tool's boundary in I_p with Farthest Point Sampling [47], assigning each point a unique index. Guided by visual cues from the human demonstration indicating where interaction occurs, I_p and l_H are then used to prompt the VLM with a multi-choice problem among N_c candidates to determine the function point.

After identifying three functional keypoints in their respective keyframes, we track their motions using the previously computed sequence of relative transformations, resulting in $\{K_H^t\}_{t=0}^{N-1}$. Additionally, $p_{\text{func}}^{t_f}$ is attached to the target, referred to as the effect point p_{eff}, to represent the 3D location where the interaction occurs on the target object. Figure 3 provides a graphical illustration of the function point, grasp point, center point, effect point, target point, and target frame.

C. Function-Centric Correspondence Establishment

Functional Keypoint Transfer. At the core of FUNCTO lies the function-centric correspondence establishment using functional keypoints. FUNCTO achieves this by first transferring the functional keypoints from the demonstration tool to the test tool, obtaining $K_R^0 = [q_{\text{func}}^0, q_{\text{grasp}}^0, q_{\text{center}}^0]$, where $q \in \mathbb{R}^3$.

The functional keypoint transfer process, illustrated in Figure 2 (stage 2), consists of a coarse-grained region proposal and a fine-grained point transfer. In the first step, we begin by projecting p_{func}^0 and p_{grasp}^0 from the 3D space onto the image plane I_0. The marked I_0 serves as a reference for identifying test tool functional keypoints. Providing such a reference is essential, as the functional keypoint location is coupled with the demonstrated human action. When the test tool has multiple possible functional keypoints, selecting a functional keypoint not matching the intended action can lead to failure. Figure 4 provides a qualitative comparison of function point transfer with and without using the reference.

Similar to Section IV-B, FUNCTO first utilizes mark-based visual prompting to propose coarse candidate regions on the test tool for 2D function and grasp points. Compared to functional keypoint detection in the previous stage, two key differences are: (1) the marked I_0 is additionally provided to the VLM as a reference, and (2) the selected candidate point is expanded into a candidate region, with its size adaptively adjusted based on the 2D dimensions of the test tool. In the second step, we employ a pre-trained dense semantic correspondence model [34] to precisely transfer 2D function and grasp points from I_0 to candidate regions in o_R, resulting in q_{func}^0 and q_{grasp}^0. The dense semantic correspondence model

provides finer point-level correspondences compared to using the VLM alone. q_{center}^0 is computed as the 3D center of the test tool. For the target object, we scale p_{eff} based on the 3D dimension ratio between the demonstration and test targets to obtain q_{eff}. K_R and q_{eff} are expressed in the test target frame.

Function-Centric Correspondence. FUNCTO formulates the function-centric correspondence as geometric constraints on 3D functional keypoints, inspired by [24]. Specifically, this step specifies the function keyframe constraint (i.e., the desired test tool state at t_f) for trajectory generation.

Formally, the function keyframe constraint can be represented by a rigid transformation $\mathbf{T}_{\text{func}} \in \text{SE}(3)$ that aligns $K_H^{t_f}$ and K_R^0. This process, illustrated in Figure 2 (stage 2), can be divided into the following steps:

0. **Function Plane Construction**: Given $K_H^{t_f}$ and K_R^0, function planes $\Pi_H^{t_f}$ and Π_R^0 are constructed as follows:
   - $\mathbf{u}_H^{t_f}$ (function axis): A normalized vector pointing from the center to the function point at t_f.
   - $\mathbf{v}_H^{t_f}$: A normalized vector pointing from the function to the grasp point at t_f.
   - $\mathbf{n}_H^{t_f}$: The unit normal vector at t_f.

   Similarly, $\mathbf{u}_R^0$, $\mathbf{v}_R^0$, and $\mathbf{n}_R^0$ are defined for Π_R^0.

1. **Function Point Alignment**: The function points should be aligned to ensure that the interaction occurs at the desired location of the test tool. The function point alignment is defined by the following constraint:

$$\left\| \mathbf{T}_{\text{point}} \begin{bmatrix} q_{\text{func}}^0 \\ 1 \end{bmatrix} - \begin{bmatrix} p_{\text{func}}^{t_f} \\ 1 \end{bmatrix} \right\| = 0$$

2. **Function Plane Alignment**: The normal vectors should be aligned to ensure that function planes have the same orientation. The function plane alignment is defined by the following constraint:

$$\mathbf{n}_H^{t_f} \cdot (\text{rot}(\mathbf{T}_{\text{plane}}) \mathbf{n}_R^0) = 1$$

where $\text{rot}(\mathbf{T})$ denotes the rotation component of a rigid transformation $\mathbf{T}$.

3. **Function Axis Alignment**: The function axes, which are function-relevant operational vectors, should be aligned to ensure that the test tool is properly tilted relative to the target (e.g., pitch angle for pouring). The function
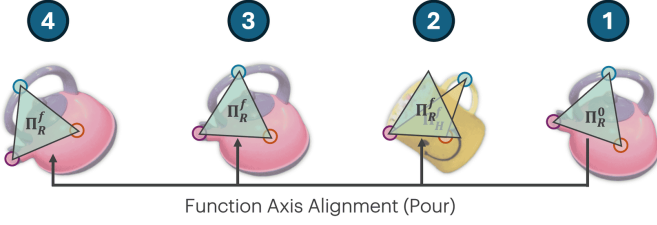
Fig. 5. An illustration of function axis alignment process: (1) test function plane Π_R^0, (2) demonstration function plane $\Pi_H^{t_f}$, (3) initially aligned test function plane $\Pi_R^{t_f}$, and (4) VLM refined test function plane $\Pi_R^{t_f}$.

axis alignment is defined by the following constraint:

$$\mathbf{u}_H^{t_f} \cdot (\mathrm{rot}(\mathbf{T}_{\mathrm{axis}})\mathbf{u}_R^0) = 1$$

However, due to structural differences between the demonstration and test tools (i.e., differences in relative locations of functional keypoints), directly applying $\mathbf{T}_{\mathrm{axis}}$ to the test tool may not result in successful task execution. For instance, a mug and a teapot may require different pouring angles. To address this issue, $\mathbf{T}_{\mathrm{axis}}$ is further refined using the VLM. Specifically, a pre-defined set of angle offsets, ranging from $[-45°, -45°]$, is applied to $\mathbf{T}_{\mathrm{axis}}$. For each offset, the combined point cloud of the test tool and target is back-projected onto the camera plane. The VLM is then prompted to identify the rendered image that represents the optimal state conducive to the task success, given the demonstration function keyframe I_f as a reference. The transformation corresponding to the optimal state is recorded as $\mathbf{T}_{\mathrm{axis}}^*$. Qualitative results of function axis alignment are illustrated in Figure 5.

Finally, the geometric constraints from each step are combined to compute $\mathbf{T}_{\mathrm{func}}$ and $K_R^{t_f}$:

$$\mathbf{T}_{\mathrm{func}} = \mathbf{T}_{\mathrm{point}} \cdot \mathbf{T}_{\mathrm{plane}} \cdot \mathbf{T}_{\mathrm{axis}}^*, \quad K_R^{t_f} = \mathbf{T}_{\mathrm{func}} K_R^0$$

To ensure that the function point interacts with the effect point at t_f, $q_{\mathrm{func}}^{t_f}$ is further adjusted to align with q_{eff}. Meanwhile, the same adjustment is applied to $q_{\mathrm{grasp}}^{t_f}$ and $q_{\mathrm{center}}^{t_f}$. The resulting $K_R^{t_f}$ represents the predicted test tool state at t_f.

D. Functional Keypoint-Based Action Planning

Tool Pose Representation. In the final stage, FUNCTO computes the test tool pose at each timestep and generates a robot end-effector trajectory τ_R for task execution. Specifically, the demonstration tool pose at timestep t can be represented, using K_H^t, as:

$$\mathbf{T}_H^t = \begin{bmatrix} \mathbf{R}_H^t & p_{\mathrm{func}}^t \\ \mathbf{0} & 1 \end{bmatrix}$$

where $\mathbf{R}_H^t$ is the rotation matrix derived from the function axis $\mathbf{u}_H^t$ and the normal vector $\mathbf{n}_H^t$. With such a pose representation, demonstration functional keypoint trajectory $\{K_H^t\}_{t=0}^{N-1}$ can be transformed to a sequence of SE(3) poses $\{\mathbf{T}_H^t\}_{t=0}^{N-1}$. Similarly, the test tool pose at t is represented as:

$$\mathbf{T}_R^t = \begin{bmatrix} \mathbf{R}_R^t & q_{\mathrm{func}}^t \\ \mathbf{0} & 1 \end{bmatrix}$$

where $\mathbf{R}_R^t$ is similarly defined. An example of the tool pose representation is illustrated in Figure 2 (stage 3). The function keyframe state $K_R^{t_f}$ and the initial keyframe state K_R^0 are transformed to the pose representations $\mathbf{T}_{\mathrm{func}}$ and $\mathbf{T}_{\mathrm{init}}$, respectively.

Tool Trajectory Optimization. Given the function keyframe pose $\mathbf{T}_{\mathrm{func}}$, the initial keyframe pose $\mathbf{T}_{\mathrm{init}}$, and the reference pose trajectory $\{\mathbf{T}_H^t\}_{t=0}^{N-1}$, the optimization problem for solving the test tool trajectory $\{\mathbf{T}_R^t\}_{t=0}^{N-1}$ can be formulated as:

$$\min_{\{\mathbf{T}_R^t\}_{t=0}^{N-1}} \sum_{t=0}^{N-1} \left(\|q_{\mathrm{func}}^t - p_{\mathrm{func}}^t\|_2^2 + \|\mathrm{Log}(\mathbf{R}_R^t(\mathbf{R}_H^t)^\top)\|^2 \right)$$

$$\text{s.t.} \quad \mathbf{T}_R^0 = \mathbf{T}_{\mathrm{init}}$$
$$\mathbf{T}_R^{t_f} = \mathbf{T}_{\mathrm{func}}$$

where $\mathrm{Log} : \mathrm{SO}(3) \to \mathbb{R}^3$[48]. This formulation can flexibly incorporate additional costs and constraints, such as smoothness costs and collision avoidance constraints. Implementation details can be found in Appendix E.

Tool Trajectory Execution. The test tool trajectory in test target frame $\{\mathbf{T}_R^t\}_{t=0}^{N-1}$ is first transformed into the robot base frame $\{\mathbf{T}_{R_{\mathrm{base}}}^t\}_{t=0}^{N-1}$. Then, we use GraspGPT [49] to sample a 6-DoF task-oriented grasp pose around q_{grasp}^0 on the test tool. Assuming the gripper is rigidly attached to the test tool after grasping, the robot end-effector trajectory τ_R can be computed with the sampled grasp pose and $\{\mathbf{T}_{R_{\mathrm{base}}}^t\}_{t=0}^{N-1}$. This trajectory is tracked and executed using operational space control.

V. EXPERIMENTS

In this section, we conduct real-robot experiments to validate FUNCTO's effectiveness and analyze key design choices. Specifically, we answer the following questions: (1) How well does FUNCTO generalize from a single human demonstration to novel tools? (2) How does FUNCTO perform compared to existing OSIL methods under the same setup? (3) How does FUNCTO compare to state-of-the-art BC methods? (4) How does each design choice in function-centric correspondence establishment (Section IV-C) affect the overall performance?

A. Experimental Setup

Baselines. We compare FUNCTO against the following OSIL baselines: (1) **DINOBOT** [4, 6], which leverages the visual correspondence capability of the vision foundation model DINO to perform semantic feature extraction and correspondence. (2) **DITTO** [5], which employs a pre-trained visual correspondence model LOFTR to estimate the relative pose transformations between demonstration and test tools. (3) **ORION** [7, 8], which extracts geometric features with Fast-Point Feature Histograms and performs a global-local registration. We adopt the original correspondence implementations of these baselines. The low-level execution components remain consistent with FUNCTO. OSIL methods with different setups, such as those requiring in-domain pre-training or object/task-specific prior knowledge, are excluded for a fair comparison.
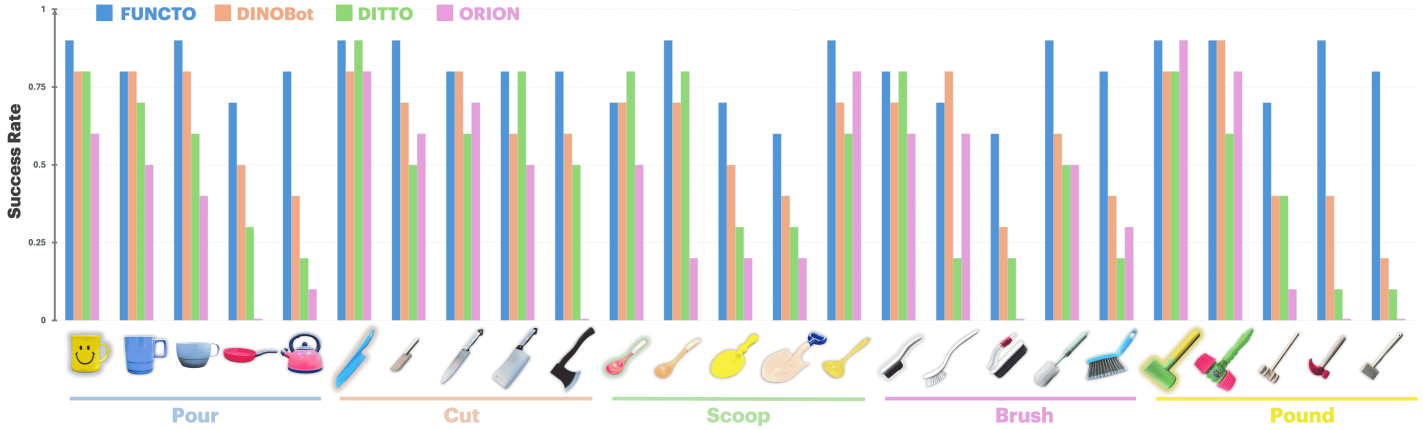
Fig. 6. Quantitative comparison to one-shot imitation learning baselines. The first tool of each function (highlighted) is used for demonstration.

TABLE I
QUANTITATIVE COMPARISON TO BEHAVIORAL CLONING BASELINES

Method	Pour		Cut		Scoop		Brush		Pound		Overall	
	Seen	Unseen	Seen	Unseen	Seen	Unseen	Seen	Unseen	Seen	Unseen	Seen	Unseen
ACT (50 Demos)	60.0%	17.5%	70.0%	40.0%	**70.0%**	47.5%	50.0%	32.5%	50.0%	25.0%	60.0%	32.5%
DP (50 Demos)	50.0%	20.0%	50.0%	22.5%	40.0%	27.5%	50.0%	35.0%	40.0%	27.5%	57.50%	26.50%
DP3 (50 Demos)	40.0%	20.0%	40.0%	25.0%	20.0%	15.0%	20.0%	12.5%	40.0%	15.0%	32.00%	17.50%
FUNCTO (1 Demo)	**90.0%**	**80.0%**	**90.0%**	**82.5%**	70.0%	**77.5%**	**80.0%**	**75.0%**	**90.0%**	**82.5%**	**84.00%**	**79.50%**

In addition to OSIL baselines, we also compare FUNCTO with BC baselines, which represent more typical imitation learning approaches in recent works. Specifically, the following BC baselines are compared: (1) ACTION CHUNKING TRANSFORMER (ACT) [19], a transformer-based BC method with action chunking and temporal ensemble. (2) DIFFUSION POLICY (DP) [2], a diffusion-based BC method that models a visuomotor policy as a conditional denoising diffusion process. For both ACT and DP, we use the pre-trained DINO-ViT as the backbone for visual feature extraction. (3) 3D DIFFUSION POLICY (DP3) [20], a more recent diffusion-based BC method that operates on 3D visual representations extracted from sparse point clouds with a lightweight point encoder.

Task Description. We evaluate FUNCTO and baselines on five tool manipulation functions: pour, cut, scoop, brush, and pound. A tool manipulation task is defined by pairing a function with a tool and a target (e.g., mug-pour-bowl). In this work, we primarily focus on addressing intra-function variations between tools, placing less emphasis on target variations. For each function, we design five tasks using different tools, divided into three levels of generalization: (1) spatial generalization (seen), where the demonstration tool is randomly positioned in the workspace; (2) instance generalization (unseen), where the demonstration and test tools are different instances from the same category; (3) category generalization (unseen), where demonstration and test tools are from different categories with the same function. In the

context of this paper, "same function" refers to imitating the tool manipulation behavior demonstrated by the human to accomplish a functionally equivalent task.

Experimental Protocol. During the demonstration phase, a single-view, actionless video of a human performing a tool manipulation task is recorded with a stationary RGB-D camera, accompanied by a task description, for each function. During the testing phase, an RGB-D image of the workspace is captured using a similar camera setup and sent to the robot for action planning and execution. For training the BC baselines, we collect 50 human demonstrations with teleportation for each function. For testing the BC baselines, we use the same targets as those in the demonstrations to emphasize the impact of variations in the test tools. In terms of performance evaluation, each method is tested with 10 trials per task, resulting in a total of 50 trials per function and 250 trials across all five functions. The detailed task success conditions are described in Appendix B. The average success rate is used as the evaluation metric.

B. Experimental Results

Quantitative Comparison to OSIL Baselines. The detailed quantitative evaluation results are reported in Figure 6. Each function is evaluated with five tasks. The first task tests spatial generalization, the next two tasks evaluate instance generalization, and the final two tasks assess category generalization.

All methods perform well in spatial generalization with the seen demonstration tool, achieving success rates above 70%. However, all baselines exhibit significant performance

Functional Keypoints | **Test Tool Trajectory** | **Real Robot Execution**

"Use the **teapot** to **pour** the contents into the **pot**"

"Use a **knife** to **cut** the **burger** in half"

"**Scoop** the **plate** with a plastic **scoop**"

"**Pound** the **nail** using the **hammer**"

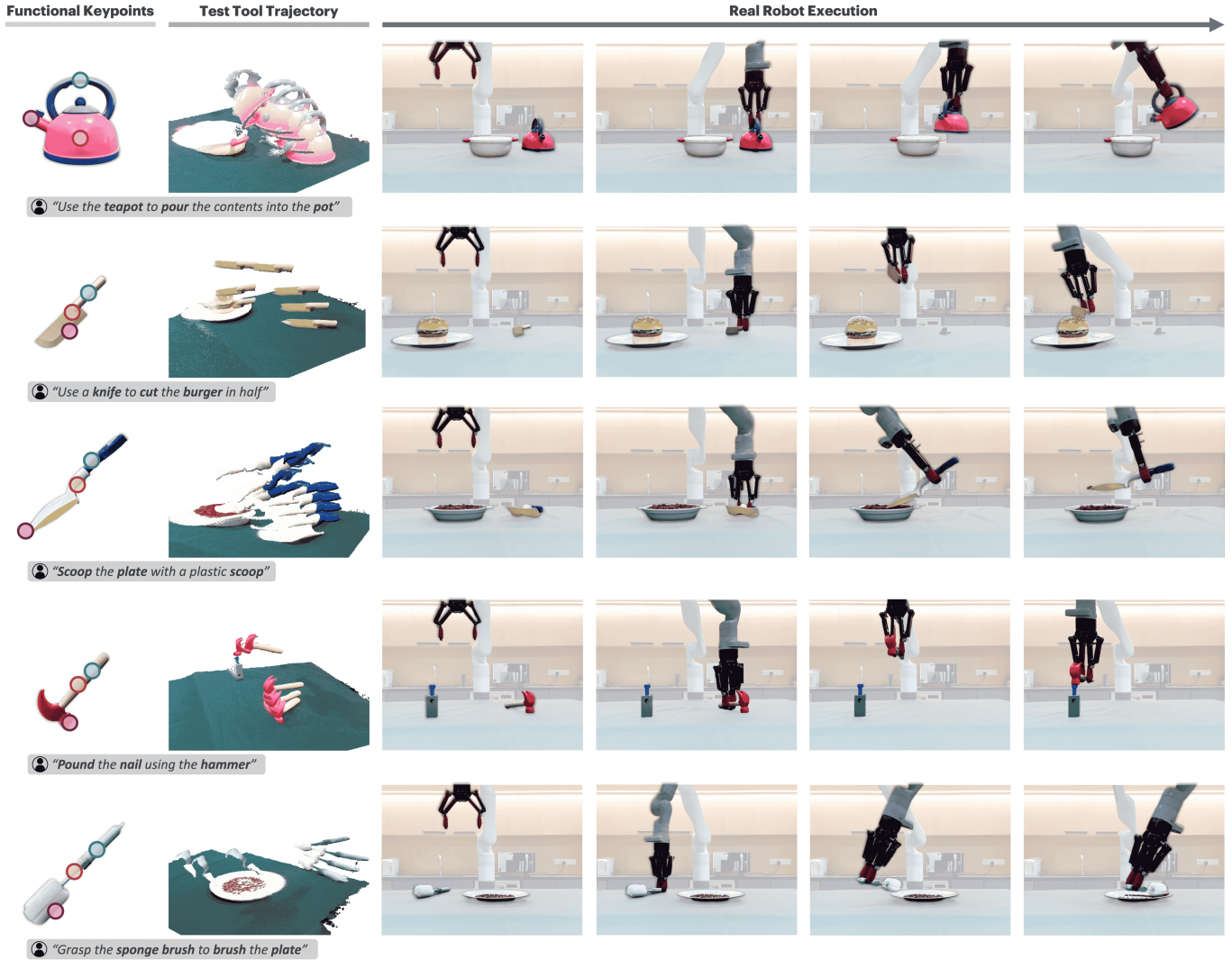"Grasp the **sponge brush** to **brush** the **plate**"

Fig. 7. Qualitative results of predicted functional keypoints, test tool trajectories, and real-robot executions across five functions.

drops (from 20% to 40%) when generalizing to novel tool instances and categories, especially for those with substantial differences in shape, size, or topology. For instance, in `teapot-pour-pot`, the teapot and the demonstrated mug differ in both shape and part topology. The teapot has a conical body, a long neck, and a handle positioned on top, whereas the mug features a cylindrical body with a side-mounted handle. We also observe that variations in size and scale negatively affect the performance of the baselines. In `hammer-pound-nail`, the red hammer is much smaller than the demonstrated mallet, causing inaccurate pounding point alignment in 3D. This results in infeasible contact between the tool and the target, highlighting the significance of the proposed function-centric correspondence.

Among all baselines, ORION relies solely on geometric features, rendering it ineffective at handling large geometric variations. DINOBot outperforms both DITTO and ORION, achieving an average success rate of 57.5% when generalizing to novel tools. This performance can be attributed to DINO's strong visual correspondence capability. However, DINOBot still struggles to establish correspondences between visually

distinct tools due to intra-function variations. In contrast, FUNCTO significantly outperforms the OSIL baselines, achieving a high success rate of 79.5% across five functions for novel tool generalization. Figure 7 visualizes the qualitative results of real-robot executions across five functions.

Quantitative Comparison to BC Baselines. The quantitative evaluation results are summarized in Table I. We divide the results into Seen and Unseen categories to emphasize the performance gap between demonstration and test tools. For seen tools, BC baselines trained on 50 demonstrations exhibit some level of generalization to different spatial layouts, with the two leading baselines, ACT and DP, achieving success rates ranging from 50% to 60%. By modeling the relative spatial relationship between the tool and the target, FUNCTO inherently supports spatial generalization.

However, all BC baselines struggle with unseen tool generalization, primarily due to intra-function variations, with success rates approximately half those of the Seen category. In contrast, FUNCTO achieves a significantly higher success rate of 79.5%, attributed to the proposed function-centric
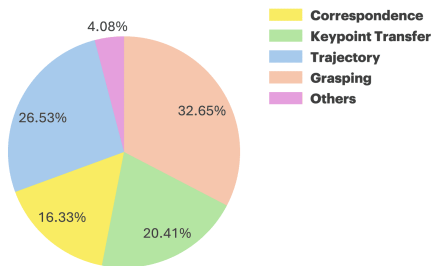
Fig. 8. Failure analysis of system components

correspondence. We also evaluate BC baselines trained with 10 and 1 demonstration(s). However, they fail to produce meaningful results and are therefore excluded from the report.

Failure Analysis. The modular design of FUNCTO facilitates the interpretation and in-depth analysis of failure cases. The result of the failure analysis is reported in Figure 8. The identified failure sources are categorized into: (1) function-centric correspondence, (2) functional keypoint transfer, (3) trajectory planning, (4) grasping, and (5) others (e.g., segmentation, detection).

The primary failures arise from (4) and (3). Specifically, failures in grasping often occur when the tool flips or slips due to unstable contact between the tool and the gripper, preventing the robot from completing the task. Incorporating an online state tracking module (probably using multiple calibrated cameras) and closed-loop execution could potentially mitigate this issue. For trajectory planning, failures primarily result from unexpected collisions between the tool and the target, particularly in contact-rich tasks (e.g., `scrubber-brush-plate`). Providing visual-tactile feedback is essential for successfully accomplishing such tasks. Functional keypoint transfer errors are mainly caused by incorrect candidate region proposals for function points but contribute less significantly to overall failures. These errors may be mitigated as VLMs continue to improve. Correspondence errors are mainly attributed to inaccurate depth information of the functional keypoints. Empirically, the function-centric correspondence works well with accurate 3D functional keypoint locations.

C. Ablation study

To gain further insights into the design choices behind the core component of FUNCTO, function-centric correspondence establishment, we conduct two sets of ablation studies: one on functional keypoint transfer and another on function-centric correspondence. Performance is evaluated on five tool manipulation tasks: `teapot-pour-pot`, `knife-cut-burger`, `hammer-pound-nail`, `scoop-scoop-bowl`, `scrubber-brush-plate`.

Ablation on Functional Keypoint Transfer. We evaluate four functional keypoint transfer strategies: (1) Demo+VLM+DSC (proposed), which utilizes demonstration functional keypoints as references to prompt the VLM for region proposal, followed by point transfer through a dense semantic correspondence model; (2) Demo+VLM, which removes the

dense semantic correspondence model from the proposed implementation; (3) Demo+DSC, which relies solely on a dense semantic correspondence model for functional keypoint transfer; (4) VLM (zero-shot), which directly prompts the VLM to propose functional keypoints in a zero-shot manner, as in MOKA [28] and ReKep [29]. The quantitative results are reported in Figure 9 (left). The proposed Demo+VLM+DSC consistently outperforms ablated versions. Demo+VLM performs reasonably well, benefiting from the rich commonsense knowledge embedded in VLMs. However, as indicated in [50], VLMs struggle to provide precise point-level correspondences, particularly for tasks requiring high precision (e.g., `hammer-pound-nail`). On the other hand, solely relying on the dense semantic correspondence model often fails when dealing with large intra-function variations. The performance gap between Demo+VLM and VLM (zero-shot) justifies the point that demonstration functional keypoints provide valuable references for test functional keypoint proposal. Additional quantitative and qualitative evaluations are provided in Appendix C.

Ablation on Function-Centric Correspondence. In this ablation study, we investigate two questions: (1) Is aligning the function point more effective than aligning other functional keypoints? (2) Is VLM refinement necessary for function axis alignment? As shown in Figure 9 (right), function point alignment achieves the optimal performance in most cases by ensuring interactions occur at the desired location of the tool, regardless of variations in shape, topology, or size. When comparing strategies with and without VLM refinement, the latter rigidly aligns the function axes, ignoring changes in the relative locations of the three functional keypoints. This strategy may produce infeasible function keyframe poses. Incorporating commonsense knowledge from VLMs for function axis refinement statistically improves the performance.

VI. CONCLUSION AND LIMITATIONS

Conclusion. In this work, we present FUNCTO, a function-centric one-shot imitation learning framework for tool manipulation. At the core of FUNCTO is the idea of functional correspondence using a 3D functional keypoint representation. With such a formulation, FUNCTO generalizes the tool manipulation skill from a single human demonstration video to novel tools with the same function despite significant intra-function variations. Extensive real-robot experiments validate the effectiveness of FUNCTO, outperforming both modular one-shot imitation learning methods and end-to-end behavioral cloning methods.

Limitations. Despite the promising results, several limitations remain: (1) Functional keypoint visibility and collinearity. Our current implementation assumes that the function point is clearly visible from the camera view. However, this assumption may not always hold, especially when learning from egocentric videos. Additionally, FUNCTO fails when the three functional
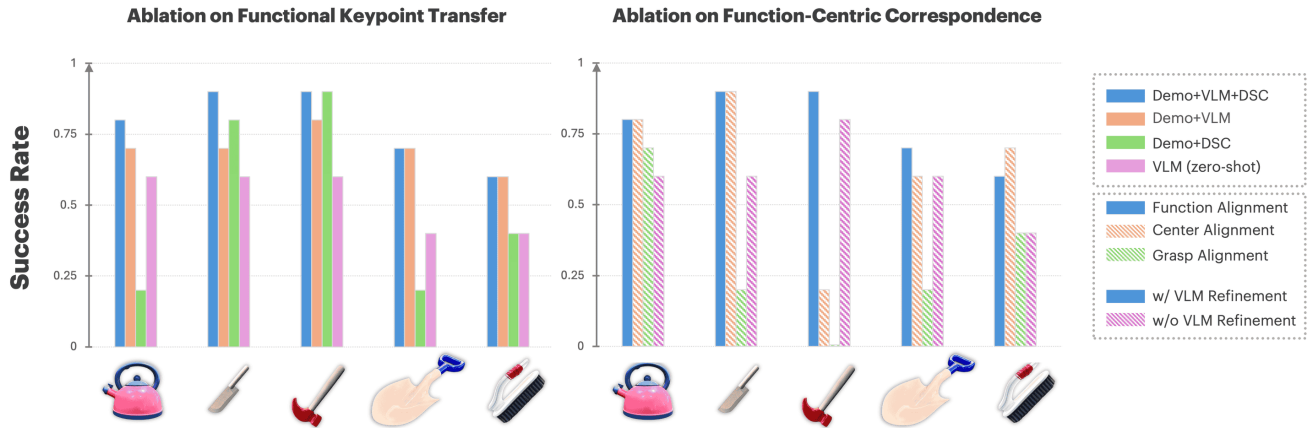
Fig. 9. Ablation studies on functional keypoint transfer (left) and function-centric correspondence (right).

keypoints are collinear in 3D. That said, such cases are uncommon for everyday tools. (2) State tracking and closed-loop execution. The current pipeline operates in an open-loop manner, which is sensitive to unexpected state changes or external disturbances. Integrating a state tracking module (probably using multiple calibrated cameras) and enabling closed-loop execution would further improve the robustness. (3) Multimodal function modeling. Functions inherently exhibit multimodality. For example, the function point on the mug shown in Figure 1 is used for forward pouring, while points positioned on the left or right sides of the rim can facilitate side pouring. Although FUNCTO is currently limited to imitating a single usage of the function based on a single demonstration, future work will focus on capturing the multi-modality of a function from few-shot human demonstrations with a human-robot interaction system [51]. Further discussions can be found in Appendix H.

REFERENCES

[1] Sherwood L Washburn. Tools and human evolution. *Scientific American*, 203(3):62–75, 1960.

[2] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.

[3] Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. In *Conference on robot learning*, pages 357–368. PMLR, 2017.

[4] Pietro Vitiello, Kamil Dreczkowski, and Edward Johns. One-shot imitation learning: A pose estimation perspective. In *Conference on Robot Learning*, pages 943–970. PMLR, 2023.

[5] Nick Heppert, Max Argus, Tim Welschehold, Thomas Brox, and Abhinav Valada. Ditto: Demonstration imitation by trajectory transformation. *arXiv preprint arXiv:2403.15203*, 2024.

[6] Norman Di Palo and Edward Johns. Dinobot: Robot manipulation via retrieval and alignment with vision foundation models. *arXiv preprint arXiv:2402.13181*, 2024.

[7] Yifeng Zhu, Arisrei Lim, Peter Stone, and Yuke Zhu. Vision-based manipulation from single human video with open-world object graphs. *arXiv preprint arXiv:2405.20321*, 2024.

[8] Jinhan Li, Yifeng Zhu, Yuqi Xie, Zhenyu Jiang, Mingyo Seo, Georgios Pavlakos, and Yuke Zhu. Okami: Teaching humanoid robots manipulation skills through single video imitation. In *8th Annual Conference on Robot Learning*.

[9] Ondrej Biza, Skye Thompson, Kishore Reddy Pagidi, Abhinav Kumar, Elise van der Pol, Robin Walters, Thomas Kipf, Jan-Willem van de Meent, Lawson LS Wong, and Robert Platt. One-shot imitation learning via interaction warping. In *Conference on Robot Learning*, pages 2519–2536. PMLR, 2023.

[10] Xinyu Zhang and Abdeslam Boularias. One-shot imitation learning with invariance matching for robotic manipulation. *arXiv preprint arXiv:2405.13178*, 2024.

[11] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016.

[12] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850. PMLR, 2016.

[13] Harrison Edwards and Amos Storkey. Towards a neural statistician. In *5th International Conference on Learning Representations*, pages 1–13, 2017.

[14] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

[15] Yan Duan, Marcin Andrychowicz, Bradly Stadie, OpenAI Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. *Advances in neural information processing systems*, 30, 2017.

[16] Tianhe Yu, Chelsea Finn, Sudeep Dasari, Annie Xie, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. *Robotics: Science and Systems XIV*, 2018.

[17] Bowen Wen, Wenzhao Lian, Kostas Bekris, and Stefan Schaal. You only demonstrate once: Category-level manipulation from single visual demonstration. *Robotics: Science and Systems 2022*, 2022.

[18] Bowen Wen, Wenzhao Lian, Kostas Bekris, and Stefan Schaal. Catgrasp: Learning category-level task-relevant grasping in clutter from simulation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6401–6408. IEEE, 2022.

[19] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[20] Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy. *arXiv preprint arXiv:2403.03954*, 2024.

[21] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*.

[22] Sungjoon Choi, Qian-Yi Zhou, and Vladlen Koltun. Robust reconstruction of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5556–5565, 2015.

[23] Diego Rodriguez, Corbin Cogswell, Seongyong Koo, and Sven Behnke. Transferring grasping skills to novel instances by latent space non-rigid registration. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4229–4236. IEEE, 2018.

[24] Lucas Manuelli, Wei Gao, Peter Florence, and Russ Tedrake. kpam: Keypoint affordances for category-level robotic manipulation. In *The International Symposium of Robotics Research*, pages 132–157. Springer, 2019.

[25] Lucas Manuelli, Yunzhu Li, Pete Florence, and Russ Tedrake. Keypoints into the future: Self-supervised correspondence in model-based reinforcement learning. In *Conference on Robot Learning*, pages 693–710. PMLR, 2021.

[26] Zengyi Qin, Kuan Fang, Yuke Zhu, Li Fei-Fei, and Silvio Savarese. Keto: Learning keypoint representations for tool manipulation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7278–7285. IEEE, 2020.

[27] Ruinian Xu, Fu-Jen Chu, Chao Tang, Weiyu Liu, and Patricio A Vela. An affordance keypoint detection network for robot manipulation. *IEEE Robotics and Automation Letters*, 6(2):2870–2877, 2021.

[28] Fangchen Liu, Kuan Fang, Pieter Abbeel, and Sergey Levine. Moka: Open-vocabulary robotic manipulation through mark-based visual prompting. In *First Workshop on Vision-Language Models for Navigation and Manipulation at ICRA 2024*.

[29] Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. In *8th Annual Conference on Robot Learning*.

[30] Jianfeng Gao, Zhi Tao, Noémie Jaquier, and Tamim Asfour. K-vil: Keypoints-based visual imitation learning. *IEEE Transactions on Robotics*, 2023.

[31] Jianfeng Gao, Xiaoshu Jin, Franziska Krebs, Noémie Jaquier, and Tamim Asfour. Bi-kvil: Keypoints-based visual imitation learning of bimanual manipulation tasks. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 16850–16857. IEEE, 2024.

[32] Peter R Florence, Lucas Manuelli, and Russ Tedrake. Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. In *Conference on Robot Learning*, pages 373–385. PMLR, 2018.

[33] Soroush Nasiriany, Fei Xia, Wenhao Yu, Ted Xiao, Jacky Liang, Ishita Dasgupta, Annie Xie, Danny Driess, Ayzaan Wahid, Zhuo Xu, et al. Pivot: Iterative visual prompting elicits actionable knowledge for vlms. In *Forty-first International Conference on Machine Learning*.

[34] Junyi Zhang, Charles Herrmann, Junhwa Hur, Luisa Polania Cabrera, Varun Jampani, Deqing Sun, and Ming-Hsuan Yang. A tale of two features: Stable diffusion complements dino for zero-shot semantic correspondence. *Advances in Neural Information Processing Systems*, 36, 2024.

[35] Bowen Wen, Wei Yang, Jan Kautz, and Stan Birchfield. Foundationpose: Unified 6d pose estimation and tracking of novel objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17868–17879, 2024.

[36] Christian Bailer, Kiran Varanasi, and Didier Stricker. Cnn-based patch matching for optical flow with thresholded hinge embedding loss. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3250–3259, 2017.

[37] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker: It is better to track together. In *European Conference on Computer Vision*, pages 18–35. Springer, 2025.

[38] Hongtao Wen, Jianhang Yan, Wanli Peng, and Yi Sun. Transgrasp: Grasp pose estimation of a category of objects by transferring grasps from only one labeled instance. In *European Conference on Computer Vision*, pages 445–461. Springer, 2022.

[39] Yu Deng, Jiaolong Yang, and Xin Tong. Deformed implicit field: Modeling 3d shapes with learned dense correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10286–10296, 2021.

[40] Anthony Simeonov, Yilun Du, Andrea Tagliasac-

chi, Joshua B Tenenbaum, Alberto Rodriguez, Pulkit Agrawal, and Vincent Sitzmann. Neural descriptor fields: Se (3)-equivariant object representations for manipulation. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6394–6400. IEEE, 2022.

[41] JJ Gibson. The theory of affordances. *Perceiving, acting and knowing: Towards an ecological psychology/Erlbaum*, 1(2):67–82, 1977.

[42] Zihang Lai, Senthil Purushwalkam, and Abhinav Gupta. The functional correspondence problem. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15772–15781, 2021.

[43] Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, et al. Grounded sam: Assembling openworld models for diverse visual tasks. *arXiv preprint arXiv:2401.14159*, 2024.

[44] Oene Bottema and Bernard Roth. *Theoretical kinematics*, volume 24. Courier Corporation, 1990.

[45] Dandan Shan, Jiaqi Geng, Michelle Shu, and David F Fouhey. Understanding human hands in contact at internet scale. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9869–9878, 2020.

[46] Rebecca Killick, Paul Fearnhead, and Idris A Eckley. Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association*, 107(500):1590–1598, 2012.

[47] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[48] Joan Sola, Jeremie Deray, and Dinesh Atchuthan. A micro lie theory for state estimation in robotics. *arXiv preprint arXiv:1812.01537*, 2018.

[49] Chao Tang, Dehao Huang, Wenqi Ge, Weiyu Liu, and Hong Zhang. Graspgpt: Leveraging semantic knowledge from a large language model for task-oriented grasping. *IEEE Robotics and Automation Letters*, 2023.

[50] Pooyan Rahmanzadehgervi, Logan Bolton, Mohammad Reza Taesiri, and Anh Totti Nguyen. Vision language models are blind. In *Proceedings of the Asian Conference on Computer Vision*, pages 18–34, 2024.

[51] Anxing Xiao, Nuwan Janaka, Tianrun Hu, Anshul Gupta, Kaixin Li, Cunjun Yu, and David Hsu. Robi butler: Remote multimodal interactions with household robot assistant. *arXiv preprint arXiv:2409.20548*, 2024.

VII. Appendix

A. FUNCTO

In this section, we provide additional qualitative results of the function-centric correspondences established by FUNCTO across five functions.

Fig. 10. Function-centric correspondences established by FUNCTO across five functions.

B. Real-Robot Experiment

Experimental Setup. All experiments are conducted on the platform depicted in Figure 11. The platform consists of a Kinova Gen3 7-DoF robotic arm and an Azure Kinect RGB-D camera. During each trial, a tool object and a target object are placed within the robot's workspace, which is a 50cm × 30cm region.
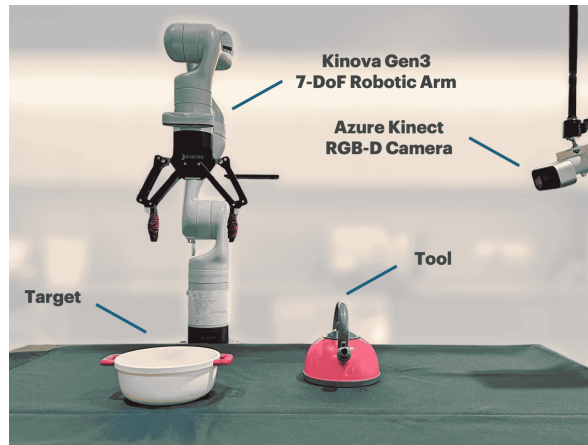

Fig. 11.   Experimental platform

The tool and target objects used in the experiments are shown in Figure 12 and Figure 13, respectively. The leftmost tool of each function is used for human demonstration.

Fig. 12.   Tool objects used in the real-robot experiments.

Fig. 13.   Target objects used in the real-robot experiments.

Task Success Conditions.

- **Pour**: The particles within the tool are transferred into the target container.
- **Cut**: The blade of the tool makes contact with the target from above.
- **Scoop**: The tool collects and securely holds particles from the target container.
- **Pound**: The bottom of the tool head strikes the nail head.
- **Brush**: The tool moves across the target's surface, displacing particles with its bristle.

Qualitative Results.

Fig. 14. Qualitative results of predicted functional keypoints, trajectories, and real-robot executions (pour, cut, scoop).

Functional Keypoints   **Test Tool Trajectory**   **Real Robot Execution**

"Brush the plate with the scrubber"

"Use the long handle scrubber to brush the plate"

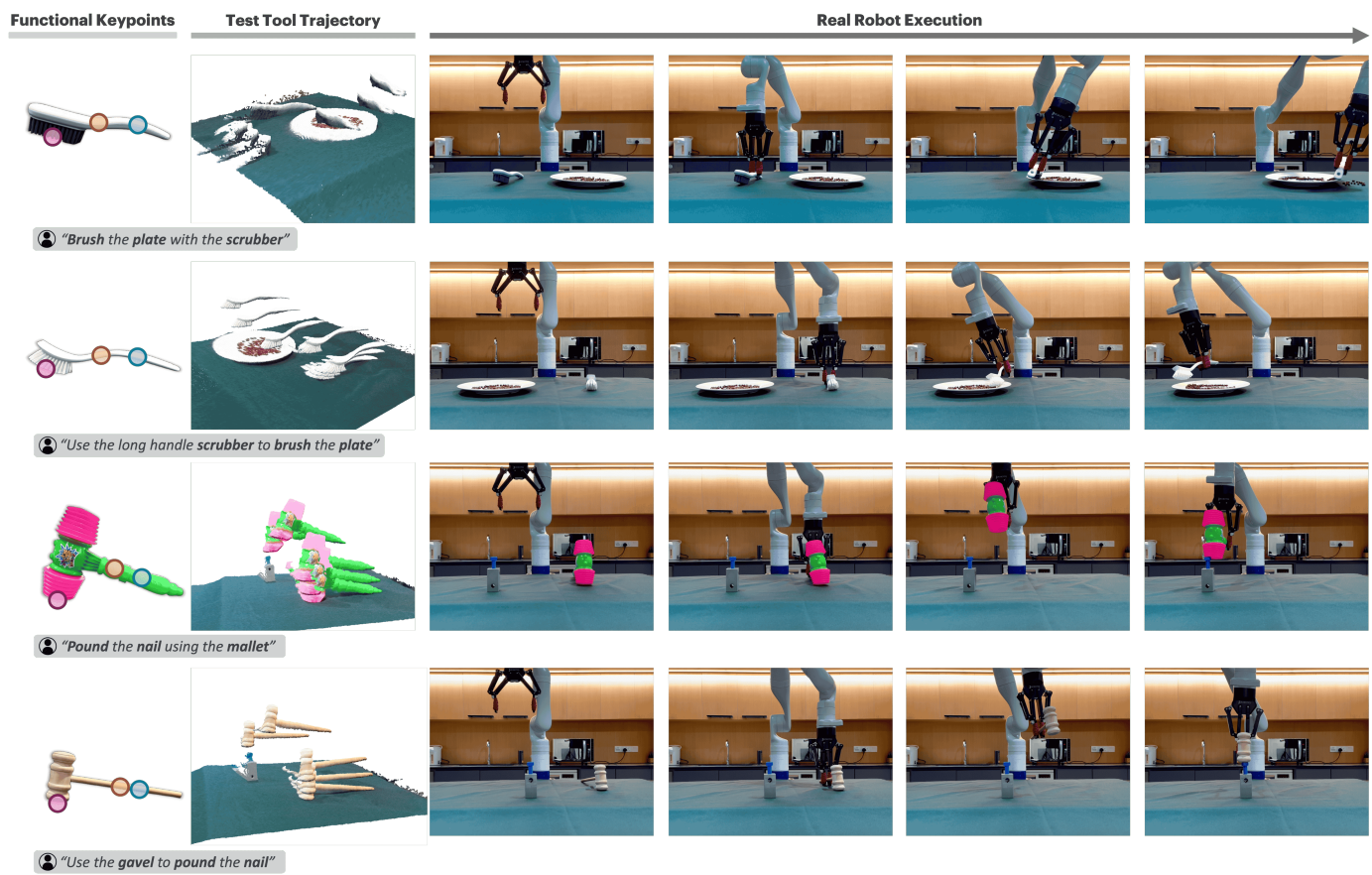*"Pound the nail using the mallet"*

"Use the gavel to pound the nail"

Fig. 15.   Qualitative results of predicted functional keypoints, trajectories, and real-robot executions (brush, pound).

C. Functional Keypoint Transfer Experiment

In addition to the real-robot experiments, we compare the performance of different functional keypoint transfer strategies, with a focus on the function point transfer.

Baselines. We evaluate four function point transfer strategies:

- Demo+VLM+DSC (proposed), which utilizes demonstration functional keypoints as references to prompt the VLM for region proposal, followed by point transfer through a dense semantic correspondence model;
- Demo+VLM, which removes the dense semantic correspondence model from the proposed implementation;
- Demo+DSC, which relies solely on a dense semantic correspondence model for functional keypoint transfer;
- VLM (zero-shot), which directly prompts the VLM to propose functional keypoints in a zero-shot manner.

Experimental Setup. For each test tool used in the real-robot experiment, we capture RGB images from 6 different views, covering various positions and orientations within the workspace. Each image has a resolution of 1280*720. A total of 150 images are used for evaluation. A set of examples is shown in Figure 16.
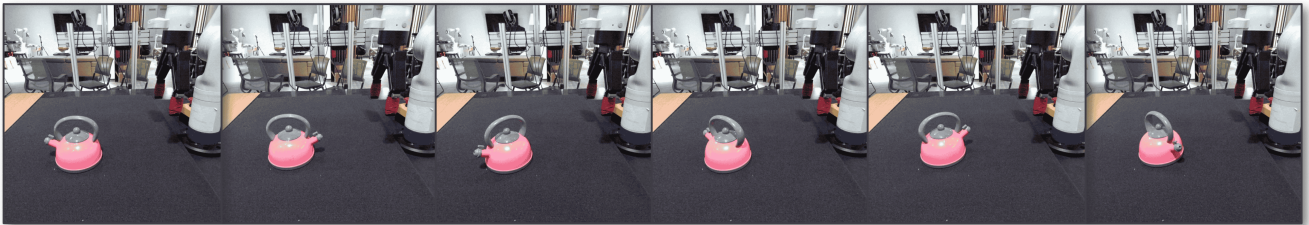

Fig. 16. Examples of collected images for function point transfer evaluation.

Evaluation Protocol. To collect ground truth for function point transfer evaluation, five volunteers were asked to annotate keypoints on test images using demonstration function points as references. Two evaluation metrics are used: (1) Average Keypoint Distance (AKD), which measures the average pixel distance between ground truth and detected keypoints. (2) Average Precision (AP), which represents the proportion of correctly detected keypoints under various thresholds. AP is evaluated under three thresholds: 15, 30, and 45 pixels.

TABLE II
QUANTITATIVE RESULTS OF FUNCTION POINT TRANSFER

Method	AKD (pixel) ↓	AP@15 (%) ↑	AP@30 (%)↑	AP@45 (%)↑
Demo+VLM	26.42	38.89	68.44	83.56
Demo+DSC	33.54	47.11	68.67	78.67
VLM (zero-shot)	56.09	15.56	36.22	52.67
Demo+VLM+DSC (proposed)	**18.54**	**51.33**	**85.78**	**94.44**

Quantitative results. The quantitative results of function point transfer are presented in Table II. The proposed Demo+VLM+DCS consistently outperforms the ablated strategies in both AKD and AP metrics. Demo+VLM achieves reasonable performance by leveraging the rich commonsense knowledge embedded in VLMs. However, VLMs alone struggle to provide precise point-level correspondences, which limits the effectiveness of Demo+VLM compared to the proposed strategy. Meanwhile, relying solely on the dense semantic correspondence model (i.e., Demo+DSC) often fails when faced with large intra-function variations. The performance gap between Demo+VLM and VLM (zero-shot) highlights the importance of demonstration functional keypoints, which serve as valuable references for proposing test functional keypoints.
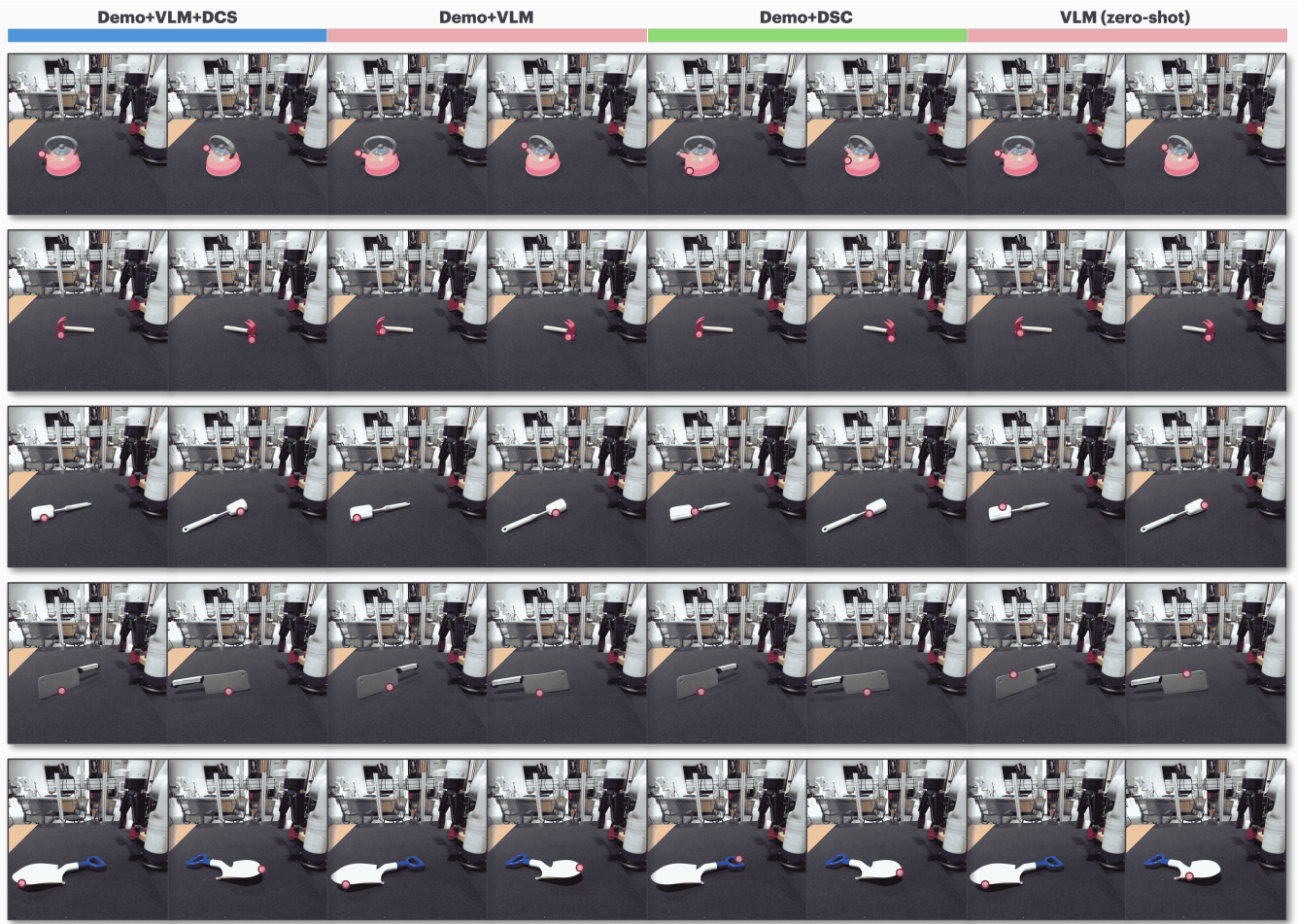
Qualitative results.

Fig. 17.   Qualitative results of function point transfer.

D. Function-Centric Correspondence Implementation Detail

This section provides the implementation details for functional keypoint transfer and function-centric correspondence.

Function Keypoint Transfer. The pseudo-code for functional keypoint transfer is illustrated in Algorithm 1.

Algorithm 1 Functional Keypoint Transfer.

Input:
Demo functional keypoints $K_H^0 = [p_{\text{func}}^0, p_{\text{grasp}}^0, p_{\text{center}}^0]$, Initial keyframe I_0, Robot observation o_R, Test tool mask M,
Vision-language model (VLM), Dense semantic correspondence model Φ,
3D-2D projection $P_{\text{3D-2D}}$, 2D-3D projection $P_{\text{2D-3D}}$, 3D center computation F_{center}
Output: Test functional keypoints $K_R^0 = [q_{\text{func}}^0, q_{\text{grasp}}^0, q_{\text{center}}^0]$

1:   $K_R \leftarrow \emptyset$
2:   **1. Coarse-Grained Region Proposal:**
3:   **for** each $k \in \{\text{func}, \text{grasp}\}$ **do**
4:       $p_k^{2D} \leftarrow P_{\text{3D-2D}}(p_k^0, I_0)$
5:       $r_k \leftarrow \text{VLM}(p_k^{2D}, I_0, o_R, M)$ ▷ Region proposal
6:   **end for**
7:   **2. Fine-Grained Point Transfer:**
8:   **for** each $k \in \{\text{func}, \text{grasp}\}$ **do**
9:       $q_k^{2D} \leftarrow \Phi(p_k^{2D}, r_k, I_0, o_R)$ ▷ Point transfer
10:      $q_k^0 \leftarrow P_{\text{2D-3D}}(q_k^{2D}, o_R)$
11: **end for**
12: **3. 3D Center Computation:**
13: $q_{\text{center}}^0 \leftarrow F_{\text{center}}(M, o_R)$
14: **4. Functional Keypoint Transfer Output:**
15: $K_R^0 \leftarrow [q_{\text{func}}^0, q_{\text{grasp}}^0, q_{\text{center}}^0]$

Function Plane Construction. We aim to construct function planes $\Pi_H^{t_f}$ and Π_R^0 based on the functional keypoints $K_H^{t_f} = [p_{\text{func}}^{t_f}, p_{\text{grasp}}^{t_f}, p_{\text{center}}^{t_f}]$ and $K_R^0 = [q_{\text{func}}^0, q_{\text{grasp}}^0, q_{\text{center}}^0]$. $\Pi_H^{t_f}$ are defined by the following vectors:

1) **Function Axis**
   - Definition:
$$\mathbf{u}_H^{t_f} = \frac{p_{\text{func}}^{t_f} - p_{\text{center}}^{t_f}}{\|p_{\text{func}}^{t_f} - p_{\text{center}}^{t_f}\|}$$

   - Description: $\mathbf{u}_H^{t_f}$ is a normalized vector that defines the function axis. It points from the center point $p_{\text{center}}^{t_f}$ to the function point $p_{\text{func}}^{t_f}$ at t_f. This axis represents the primary direction along which the function operates.

2) **Grasp Vector**
   - Definition:
$$\mathbf{v}_H^{t_f} = \frac{p_{\text{grasp}}^{t_f} - p_{\text{func}}^{t_f}}{\|p_{\text{grasp}}^{t_f} - p_{\text{func}}^{t_f}\|}$$

   - Description: $\mathbf{v}_H^{t_f}$ is a normalized vector that points from the function point $p_{\text{func}}^{t_f}$ to the grasp point $p_{\text{grasp}}^{t_f}$ at t_f.

3) **Normalized Normal Vector**
   - Definition:
$$\mathbf{n}_H^{t_f} = \frac{\mathbf{u}_H^{t_f} \times \mathbf{v}_H^{t_f}}{\|\mathbf{u}_H^{t_f} \times \mathbf{v}_H^{t_f}\|}$$

   - Description: $\mathbf{n}_H^{t_f}$ is the unit normal vector of the function plane $\Pi_H^{t_f}$.

4) **Function Plane**
   - Definition:
$$\Pi_H^{t_f} : (\mathbf{p} - p_{\text{func}}^{t_f}) \cdot \mathbf{n}_H^{t_f} = 0$$

   - Description: The function plane is defined by the function point and its normal vector, describing the tool's orientation and spatial configuration at t_f.

Similarly, $\mathbf{u}_R^0$, $\mathbf{v}_R^0$, and $\mathbf{n}_R^0$ are defined for Π_R^0.

Function Axis Alignment. Simply aligning the function axes of the demonstration and test tools may not yield a feasible function keyframe pose for the test tool. This is due to substantially different relative locations of the three functional keypoints (particularly for cross-category generalization). As is shown in Figure 18, the two function keyframe poses in Step 3 may fail to achieve successful task executions, as the teapot and axe are not tilted sufficiently.
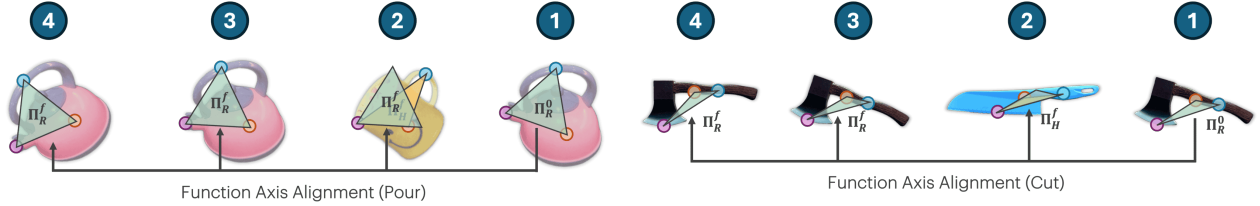

Fig. 18.   Examples of function axis alignment.

To address this issue, we refine the function axis alignment using the VLM. Specifically, in Step 3, we rotate the function plane with the function point as the origin and the normal vector as the rotation axis. Seven angle offsets ranging from $[-45°, -45°]$ are applied , including $-45°$, $-30°$, $-10°$, $0°$, $10°$, $30°$, $45°$. Next, the combined point cloud of each rotated test tool and the target are back-projected onto the camera planes using the camera intrinsic matrix, rendering seven synthetic function keyframes. Examples of rendered synthetic function keyframes are presented in Figure 19. Then, we prompt the VLM, using the demonstration function keyframe as the reference, to identify the image that represents the optimal state conducive to the task success. The detailed prompt is given in Appendix G. Finally, the rotation transformation corresponding to the optimal function keyframe state is recorded for function axis alignment (Step 4).

Fig. 19.   Examples of rendered synthetic function keyframes. The demonstration function keyframe is highlighted in yellow, and the selected test function keyframe is highlighted in green.

E. Trajectory Optimization Implementation Detail

In the section, we provide implementation details for trajectory optimization, complementing the constrained optimization problem formulated in the manuscript.

Demonstration Trajectory and Pose Wrapping. As shown in Figure 20, Step 1, the demonstration and test tools are positioned on opposite sides of the target. While the demonstration trajectory requires the test tool to approach from the left side for pouring, the target object's rotational symmetry about the z-axis allows approaching from the right side as well. This symmetry, common among target objects in the experiment, enables us to warp the demonstration trajectory and function keyframe pose to generate shorter and easier-to-execute test tool trajectories.
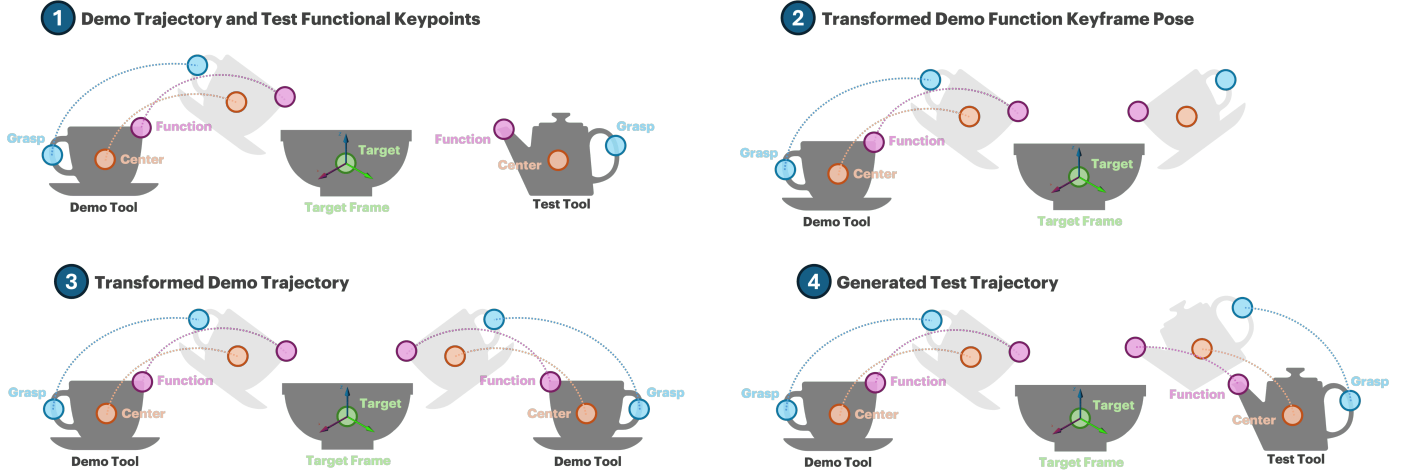

Fig. 20. Demonstration trajectory and function keyframe pose wrapping.

In Figure 20, Step 2, we align the demonstration function frame pose with the test tool by rotating the demonstration functional keypoints around the z-axis. The rotation angle is computed based on the angular difference between the demonstration and test function points. Similarly, in Step 3, we wrap the demonstration trajectory through two operations: (1) aligning it with the test tool by rotating around the z-axis and (2) scaling its translational component based on the test tool's function point. Finally, the wrapped demonstration trajectory and function keyframe pose serve as the reference and constraint for trajectory optimization (Step 4).

Optimization Constraints and Costs. In addition to the trajectory cost and the keyframe pose constraints described in the manuscript, we incorporate the following adjustments:

- Early Trajectory Cost Relaxation. The trajectory cost is omitted for the initial 30% of the trajectory, as the interaction primarily occurs during the later phases. This approach also allows the optimizer to explore more feasible paths and ensure smoother transitions to the interaction phase, particularly when the initial states of the demonstration and test tools differ significantly.
- Velocity Constraint. We impose limits on the translational and angular velocities of the test tool to ensure smooth and physically feasible trajectory generation.
- Collision Avoidance Constraint. This constraint enforces a minimum Euclidean distance between the test tool and the 3D bounding box of the obstacle to prevent collisions during trajectory execution.

We employ CasADi as the optimization framework for symbolic modeling and automatic differentiation. IPOPT is used as the solver to efficiently handle the nonlinear programming problem with constraints.

F. PD Controller Implementation Detail

This section details the practical implementation, control architecture, and parameter selection for the implemented velocity-based PD controller.

Controller Architecture. The controller implements joint velocity control with null space optimization. The input commands are received at 10 Hz while the controller operates at a higher frequency of 200 Hz. To ensure smooth motion, we implement trajectory interpolation between commanded poses.

The PD control law is formulated separately for translation and rotation.

  1) For the translational motion:

$$\mathbf{v}_{\text{lin}} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} = \underbrace{\mathbf{K}_p}_{\text{proportional}} \underbrace{\begin{bmatrix} e_{p,x} \\ e_{p,y} \\ e_{p,z} \end{bmatrix}}_{\text{position error}} + \underbrace{\mathbf{K}_d}_{\text{derivative}} \underbrace{\begin{bmatrix} \dot{e}_{p,x} \\ \dot{e}_{p,y} \\ \dot{e}_{p,z} \end{bmatrix}}_{\text{velocity error}}$$

  2) For the rotational motion:

$$\boldsymbol{\omega} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix} = \underbrace{\mathbf{K}_{p,\text{rot}}}_{\text{proportional}} \underbrace{\begin{bmatrix} e_{\theta,x} \\ e_{\theta,y} \\ e_{\theta,z} \end{bmatrix}}_{\text{orientation error}} + \underbrace{\mathbf{K}_{d,\text{rot}}}_{\text{derivative}} \underbrace{\begin{bmatrix} \dot{e}_{\theta,x} \\ \dot{e}_{\theta,y} \\ \dot{e}_{\theta,z} \end{bmatrix}}_{\text{angular velocity error}}$$

where the control gains are represented by the following matrices:

$$\mathbf{K}_p = \begin{bmatrix} 3.0 & 0 & 0 \\ 0 & 3.0 & 0 \\ 0 & 0 & 3.0 \end{bmatrix} \in \mathbb{R}^{3\times3}, \quad \mathbf{K}_d = \begin{bmatrix} 0.001 & 0 & 0 \\ 0 & 0.001 & 0 \\ 0 & 0 & 0.001 \end{bmatrix} \in \mathbb{R}^{3\times3}$$

$$\mathbf{K}_{p,\text{rot}} = \begin{bmatrix} 3.0 & 0 & 0 \\ 0 & 3.0 & 0 \\ 0 & 0 & 3.0 \end{bmatrix} \in \mathbb{R}^{3\times3}, \quad \mathbf{K}_{d,\text{rot}} = \begin{bmatrix} 0.01 & 0 & 0 \\ 0 & 0.01 & 0 \\ 0 & 0 & 0.01 \end{bmatrix} \in \mathbb{R}^{3\times3}$$

Trajectory Interpolation. To address the frequency mismatch between command inputs and controller execution, we implement a trajectory interpolation scheme. Given two consecutive desired poses at times t_k and t_{k+1}:

$$\mathbf{X}_k = \begin{bmatrix} \mathbf{p}_k \\ \mathbf{q}_k \end{bmatrix}, \quad \mathbf{X}_{k+1} = \begin{bmatrix} \mathbf{p}_{k+1} \\ \mathbf{q}_{k+1} \end{bmatrix}$$

where $\mathbf{p}$ represents position and $\mathbf{q}$ represents orientation in quaternion form.
The number of interpolation points is determined by:

$$N = \max(1, \lfloor (t_{k+1} - t_k) f_c \rfloor)$$

For position interpolation, we use linear interpolation:

$$\mathbf{p}(s) = (1 - s)\mathbf{p}_k + s\mathbf{p}_{k+1}, \quad s \in [0, 1]$$

For orientation interpolation, we use spherical linear interpolation (SLERP):

$$\mathbf{q}(s) = \frac{\sin((1-s)\Omega)}{\sin(\Omega)}\mathbf{q}_k + \frac{\sin(s\Omega)}{\sin(\Omega)}\mathbf{q}_{k+1}$$

where $\Omega = \arccos(\mathbf{q}_k \cdot \mathbf{q}_{k+1})$ is the angle between quaternions.
The interpolation parameter s is discretized as:

$$s_i = \frac{i}{N-1}, \quad i = 0, \ldots, N-1$$

Error Computation. The translational error is computed directly in Cartesian space:

$$\mathbf{e}_p = \mathbf{x}_{\text{des}} - \mathbf{x}_{\text{cur}}$$

For safety, the controller implements a position error threshold:

$$\|\mathbf{e}_p\| \leq 0.5 \text{ m}$$

The rotational error is computed using rotation matrices:

$$\mathbf{R}_{\text{error}} = \mathbf{R}_{\text{des}} \mathbf{R}_{\text{cur}}^{-1}$$

The error is converted to axis-angle representation and normalized to ensure the rotation angle remains within $[-\pi, \pi]$:

$$\mathbf{e}_\theta = \begin{cases} \mathbf{e}_{\text{axis-angle}} & \text{if } \|\mathbf{e}_{\text{axis-angle}}\| \leq \pi \\ \mathbf{e}_{\text{axis-angle}} \frac{\|\mathbf{e}_{\text{axis-angle}}\| - 2\pi}{\|\mathbf{e}_{\text{axis-angle}}\|} & \text{otherwise} \end{cases}$$

Joint Space Control. For the joint velocity control mode, the Cartesian velocities are mapped to joint space using the manipulator Jacobian:

$$\dot{\mathbf{q}} = \mathbf{J}^\dagger \begin{bmatrix} \mathbf{v}_{\text{lin}} \\ \boldsymbol{\omega} \end{bmatrix} + \mathbf{N}\dot{\mathbf{q}}_0$$

where:

- $\mathbf{J}^\dagger$ is the Moore-Penrose pseudoinverse of the Jacobian
- $\mathbf{N} = \mathbf{I} - \mathbf{J}^\dagger\mathbf{J}$ is the null space projector
- $\dot{\mathbf{q}}_0$ is the null space velocity

Null Space Optimization. The null space velocity combines two objectives:

$$\dot{\mathbf{q}}_0 = K_{\text{home}}\left(\mathbf{q}_{\text{home}} - \mathbf{q}_{\text{cur}}\right) - K_{\text{min}}\mathbf{q}_{\text{cur}}$$

where:

- $K_{\text{home}} = 0.1$ is the gain for home configuration attraction
- $K_{\text{min}} = 0.05$ is the gain for joint velocity minimization
- $\mathbf{q}_{\text{home}}$ is the preferred home configuration

G. VLM Prompting Implementation Detail

Function Point Detection Prompt

Given an interaction frame between two objects, select pre-defined keypoints.

The input request contains:
- The task information as dictionaries. The dictionary contains these fields:
  – '**instruction**': The task in natural language forms.
  – '**object_grasped**': The object that the human holds in hand while executing the task.
  – '**object_unattached**': The object that the human will interact with 'object_grasped' without holding it in hand.
- An image of the current table-top environment captured from a third-person view camera, annotated with a set of visual marks:
  – **candidate keypoints on 'object_grasped'**: Red dots marked as 'P_i' on the image, where [i] is an integer.

The interaction is specified by 'function_keypoint' on the 'object_grasped':
- The human hand grasps 'object_grasped' and moves the 'function_keypoint' to approach 'object_unattached'.
- '**function_keypoint**': The point on 'object_grasped' indicating the part that will contact 'object_unattached'.

The response should be a dictionary in JSON form, which contains:
- '**function_keypoint**': Selected from candidate keypoints marked as 'P_i' on the image.

Think about this step by step:
1. Describe the region where interaction between 'object_grasped' and 'object_unattached' happens.
2. Select 'function_keypoint' on the 'object_grasped' within the interaction region.

Function Point Transfer Prompt

Refer to the position of red keypoint on the first example image, select corresponding pre-defined keypoints on the second test image.

The input request contains:
- The task information as dictionaries. The dictionary contains these fields:
  – '**instruction**': The task in natural language forms.
  – '**object_grasped**': The object that the human holds in hand while executing the task.
  – '**object_unattached**': The object that the human will interact with 'object_grasped' without holding it in hand.
- An example image annotated with a red keypoint.
- A test image of the current table-top environment captured from a third-person view camera, annotated with a set of visual marks:
  – **candidate keypoints on 'object_grasped'**: Red dots marked as 'P_i' on the image, where [i] is an integer.

The interaction is specified by 'function_keypoint' on the 'object_grasped':
- Select the candidate keypoint on the test image corresponds to the red keypoint annotated on the example image.
- '**function_keypoint**': The point on 'object_grasped' indicating the part that will contact 'object_unattached'.

The response should be a dictionary in JSON form, which contains:
- '**function_keypoint**': Selected from candidate keypoints marked as 'P_i' on the image.

Think about this step by step:
1. Describe the object part where keypoint is located on the example image.
2. Describe the region where interaction between 'object_grasped' and 'object_unattached' happens.
3. Select 'function_keypoint' on the 'object_grasped' within the interaction region on the test image.

Grasp Point Transfer Prompt

Refer to the position of red keypoint on the first example image, select corresponding pre-defined keypoints on the second test image.

The input request contains:
- The task information as dictionaries. The dictionary contains these fields:
  - '**instruction**': The task in natural language forms.
  - '**object_grasped**': The object that the human holds in hand while executing the task.
  - '**object_unattached**': The object that the human will interact with 'object_grasped' without holding it in hand.
- An example image annotated with a red keypoint.
- A test image of the current table-top environment captured from a third-person view camera, annotated with a set of visual marks:
  - **candidate keypoints on 'object_grasped'**: Red dots marked as 'P$_i$' on the image, where [i] is an integer.

The interaction is specified by 'grasp_keypoint' on the 'object_grasped':
- Select the candidate keypoint on the test image corresponds to the red keypoint annotated on the example image.
- The human hand grasps the 'object_grasped' at the 'grasp_keypoint'.
- '**grasp_keypoint**': The point on 'object_grasped' indicates the part where the hand should hold.

The response should be a dictionary in JSON form, which contains:
- '**grasp_keypoint**': Selected from candidate keypoints marked as 'P$_i$' on the image.

Think about this step by step:
1. Describe the object part where keypoint is located on the example image.
2. Find the part on 'object_grasped' where humans usually grasp.
3. Select 'grasp_keypoint' on the 'object_grasped' within the interaction region on the test image.

Function Axis Alignment Prompt

From a list of interaction frames between the tool and target objects, select the image that represents the state most conducive to completing the task.

The input request contains:
- The task information as dictionaries. The dictionary contains these fields:
  - '**instruction**': The task in natural language forms.
  - '**object_grasped**': The object that the human holds in hand while executing the task.
  - '**object_unattached**': The object that the human will interact with 'object_grasped' without holding it in hand.
- A list of interaction frames between the tool and target objects.

The response should be a dictionary in JSON form, which contains:
- '**selected_idx**': the idx of the selected image.

H. Q&A Section

In this section, we address common questions about our method, clarifying potential concerns, discussing limitations, and providing insights into its design, implementation, future improvements, and broader applicability.

- **Q1: Can this method be applied to a wider range of tool manipulation tasks?**
  A1: FUNCTO is generally applicable to tool manipulation tasks involving two-object interactions (tool and target), where object dynamics are less critical to task success. Examples include peeling, sweeping, stirring, picking, placing, mixing, inserting, stacking, and flipping.

- **Q2: Can the proposed functional keypoint transfer strategy be extended to a broader range of applications beyond those demonstrated in the paper**
  A2: Yes, the proposed functional keypoint transfer strategy can be applied to other tasks involving semantic correspondences. Notably, we have adapted this approach for semantic keypoint transfer in cloth manipulation.

- **Q3: How can this method leverage additional demonstrations for improved performance?**
  A3: As discussed in the Limitations subsection, a function is inherently multi-modal. Therefore, FUNCTO can leverage few-shot human demonstrations for multi-modal modeling. During inference, the robot can retrieve the "best" demonstration from the database to enhance task execution.

- **Q4: What are the benefits of explicitly representing skill-use keypoints and trajectories?**
  A4: The interpretable explicit representation can be integrated with existing task and motion planning algorithms, enabling the execution of long-horizon tool manipulation tasks.

- **Q5: What is the role of foundation models in this approach, and why are they essential?**
  A5: Foundation models provide commonsense knowledge, enabling the inference of information that cannot be directly extracted from geometric or visual cues. For instance, tools with the same function may exhibit significant intra-function variations. Transferring functional keypoints based solely on geometric or visual similarities prone to failures. Additionally, without the commonsense reasoning embedded in foundation models, the robot may struggle to accurately infer the correct functional axis alignment transformation.