MimicFunc: Imitating Tool Manipulation from a Single Human Video via Functional Correspondence

Chao Tang[1,2] **Anxing Xiao** [2] **Yuhong Deng** [2] **Tianrun Hu** [2] **Wenlong Dong** [1]
Hanbo Zhang [2] **David Hsu** [2] **Hong Zhang** [1]
Southern University of Science and Technology[1] National University of Singapore[2]

Abstract: Imitating tool manipulation from human videos offers an intuitive approach to teaching robots, while also providing a promising and scalable alternative to labor-intensive teleoperation data collection for visuomotor policy learning. While humans can mimic tool manipulation behavior by observing others perform a task just once and effortlessly transfer the skill to diverse tools for functionally equivalent tasks, current robots struggle to achieve this level of generalization. A key challenge lies in establishing function-level correspondences, considering the significant geometric variations among functionally similar tools, referred to as intra-function variations. To address this challenge, we propose MimicFunc, a framework that establishes *functional correspondences* with function frame, a function-centric local coordinate frame constructed with keypoint-based abstraction, for imitating tool manipulation skills. Experiments demonstrate that MimicFunc effectively enables the robot to generalize the skill from a single RGB-D human video to manipulating novel tools for functionally equivalent tasks. Furthermore, leveraging MimicFunc's one-shot generalization capability, the generated rollouts can be used to train visuomotor policies without requiring labor-intensive teleoperation data collection for novel objects. Our code and video are available at https://sites.google.com/view/mimicfunc.

Keywords: Tool Manipulation, Imitation from Human Video

1 Introduction

The ability to use tools has long been recognized as a hallmark of human intelligence. While recent advances in developing generalist robots and large robotics foundation models [1, 2] have shown promise in endowing robots with similar capabilities, these approaches typically rely on extensive domain expertise and labor-intensive teleoperation data collection. In contrast, humans can mimic tool manipulation behavior by observing others perform a task just once and seamlessly transfer the skill to diverse tools for functionally equivalent tasks. Endowing robots with such an ability not only offers an intuitive approach to teaching robots through human demonstration but also provides a promising and scalable alternative to labor-intensive teleoperation data collection, unlocking the potential to leverage Internet-scale human videos for training visuomotor policies efficiently.

Toward this goal, this paper tackles the problem of imitation of tool manipulation through a single demonstration. The objective is to *enable the robot to imitate from a single human video and generalize the skill to manipulate novel tools for functionally equivalent tasks.* While humans can effortlessly achieve this goal, it remains a non-trivial challenge for robots due to intra-function variations among functionally similar tools, such as differences in shape, size, and topology, as illustrated in Figure 1. The key challenge lies in establishing function-level correspondences among tools in the presence of such variations, which requires capturing invariances in both functionality, understanding how to enable intended uses of tools, and actionability, understanding how to interact with tools to accomplish tasks. Previous methods [3–10] typically establish correspondences based on geometric or visual similarities. As a result, they have shown limited flexibility and adaptability.

9th Conference on Robot Learning (CoRL 2025), Seoul, Korea.

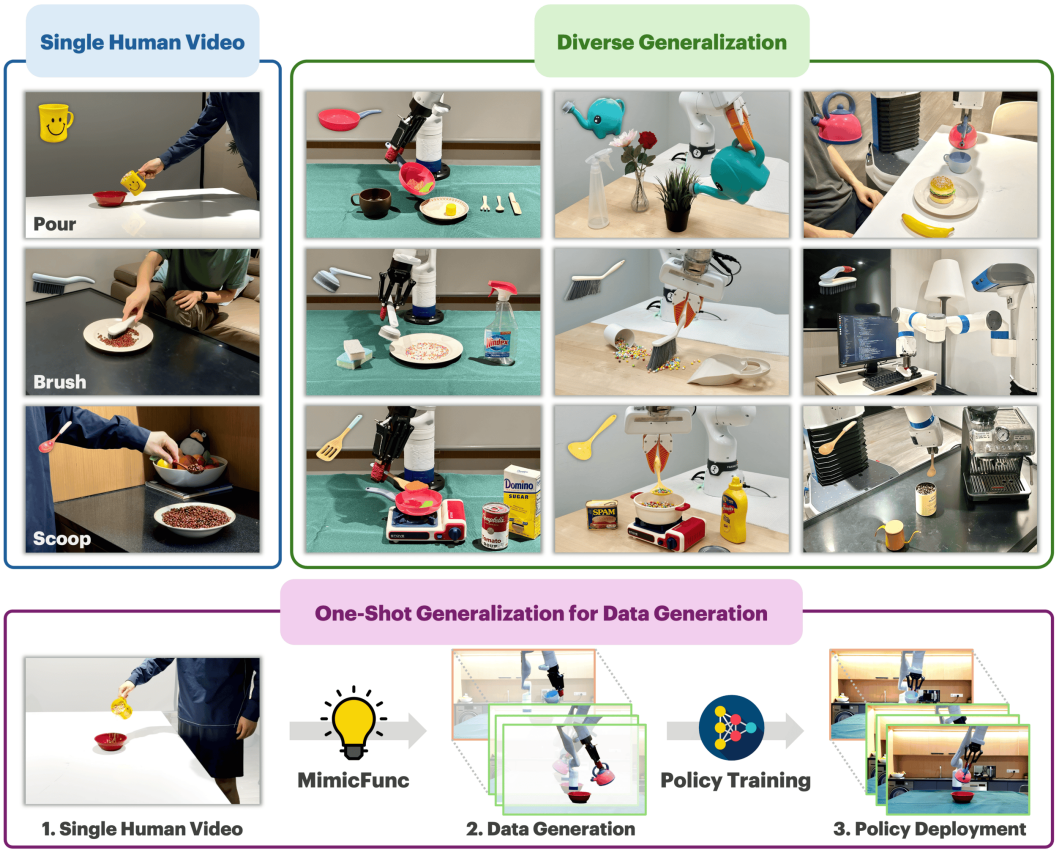


Figure 1: Given a single human video, MimicFunc enables the robot to manipulate novel tools for functionally equivalent tasks. Through the one-shot generalization capability, the rollout data generated by MimicFunc can be further leveraged to train visuomotor policies efficiently.

This limitation motivates us to ask: *How to capture invariances in tool manipulation despite significant intra-function variations?* Pioneering studies in cognitive anthropology [11] reveal that humans exhibit consistent behavioral patterns when using different tools to achieve the same function. For instance, the behavioral pattern of pouring involves approaching the tool, grasping it, and directing its spout toward the target container. This spatiotemporal pattern remains invariant across different tools affording pouring. Inspired by this observation, we propose MimicFunc, which emphasizes the functional aspects of correspondences over geometric or visual similarities for tool manipulation imitation. MimicFunc achieves this by establishing *functional correspondences* with function frame, a function-centric local coordinate frame constructed with keypoint-based abstraction, consisting of a function point, capturing tool–target interaction and serving as a spatial anchor for function frame construction; a grasp point, capturing hand–tool interaction; and a center point, serving as a consistent, object-agnostic reference for defining the function frame. Such a state representation captures the invariant spatiotemporal pattern of tool manipulation, while ignoring function-irrelevant geometric details, enabling consistent and interpretable correspondences across tools for one-shot, generalizable skill transfer.

Technically, MimicFunc is factorized into three stages: (1) Functional keypoint extraction, which detects 3D functional keypoints and extracts their motions from the human video; (2) Functional correspondence establishment, which constructs function frames with 3D functional keypoints and establishes functional correspondences via function frame alignment; (3) Function frame-based action generation, which transfers the skill by synthesizing a motion trajectory through function frame-based optimization for robot execution. Extensive real-robot experiments on diverse tool manipulation tasks demonstrate that, given a single RGB-D human video, MimicFunc enables superior generalization to novel tools with significant intra-function variations, as well as to novel spatial configurations, different embodiments, and environments. Furthermore, leveraging MimicFunc's one-shot generalization capability, the generated rollouts can be used to train visuomotor policies without requiring labor-intensive teleoperation data collection for novel objects.

2 Related Work

Imitation-based Robotic Manipulation. Imitating human behavior has been a long-standing approach in robotic manipulation [12, 13]. Recently, there has been a growing trend of training visuomotor policies using BC frameworks [14–16] on expert demonstrations. However, these approaches typically rely on substantial domain expertise and labor-intensive teleoperation data collection. Another line of research explores one-shot imitation learning [9, 17–19], yet these approaches often require extensive data collection for meta-training and struggle to generalize to out-of-domain objects with geometric or visual differences. More closely related to our work, recent studies [3–8] investigate imitating tool manipulation from a single human video through techniques such as keypoint-based pose estimation [3–5], global point set registration [6, 7], and shape warping [8]. Nevertheless, these methods typically assume that demo and test tools share highly similar shapes or appearances, which limits their generalization to novel tools. More recently, DenseMatcher [10] learns 3D dense correspondences to enable category-level manipulation from a single demonstration. However, it exhibits poor generalization under large cross-category, intra-function variations. Our work is also closely related to motion retargeting [7, 20, 21], as both aim to transfer motions across embodiments with differing kinematics. In fact, MimicFunc can be viewed as a specialized form of motion retargeting that uniquely focuses on object-centric functional intent rather than purely on embodiment-level motion transfer.

Keypoint Representation for Tool Manipulation. Keypoint representation has been extensively studied in tool manipulation [22–26]. For instance, KPAM [22] and K-VIL [25, 26] leverage 3D semantic keypoint representation to accomplish category-level tool manipulation tasks. More recent works leverage foundation models to predict semantic keypoints for tool manipulation. MOKA [23] generates planar manipulation motions via mark-based visual prompting [27], while MimicFunc extends this idea by synthesizing 3D tool manipulation trajectories, supporting more complex tasks. ReKep [24] encodes manipulation tasks as task-specific keypoint constraints in VLM prompts, but these require substantial manual effort and hand-engineering. In contrast, MimicFunc automatically extracts constraints from human videos and enables more effective trajectory generation. More importantly, unlike previous methods [23–26] that extract keypoints without explicit semantic grounding, MimicFunc builds upon an abstraction that captures both functional and physical semantics, forming a structured "functional skeleton" of the tool. Such a formulation enables more consistent and interpretable correspondences across tools for function-level generalization.

3 MimicFunc

In this section, we introduce MimicFunc, a method for imitating tool manipulation from a single human video. MimicFunc consists of three stages: (1) Functional keypoint extraction from human video (Section 3.2), (2) Functional correspondence establishment with function frame (Section 3.3), and (3) Function frame-based action generation (Section 3.4). Each will be detailed for the rest of this section. An overview of the pipeline is presented in Figure 2.

3.1 Problem Formulation

We consider the problem of enabling the robot to imitate the tool manipulation behavior from a single RGB-D human video to accomplish functionally equivalent tasks using novel tools. Specifically, each task involves manipulating a tool (object) to interact with a target (object) in a tabletop environment. During the demonstration phase, a human performs a tool manipulation task, recording a sequence of RGB-D frames, $\mathcal{V}_H = \{I_t\}_{t=0}^{N-1}$, 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, a target, and a function. During inference, with novel tools, environments, and task configurations, the objective is to map the robot observation o_R and task description l_R to a trajectory $\tau_R = \{a_t\}_{t=0}^{N-1}$ for robot execution. Here, $a_t = (R_t, T_t) \in \mathrm{SE}(3)$

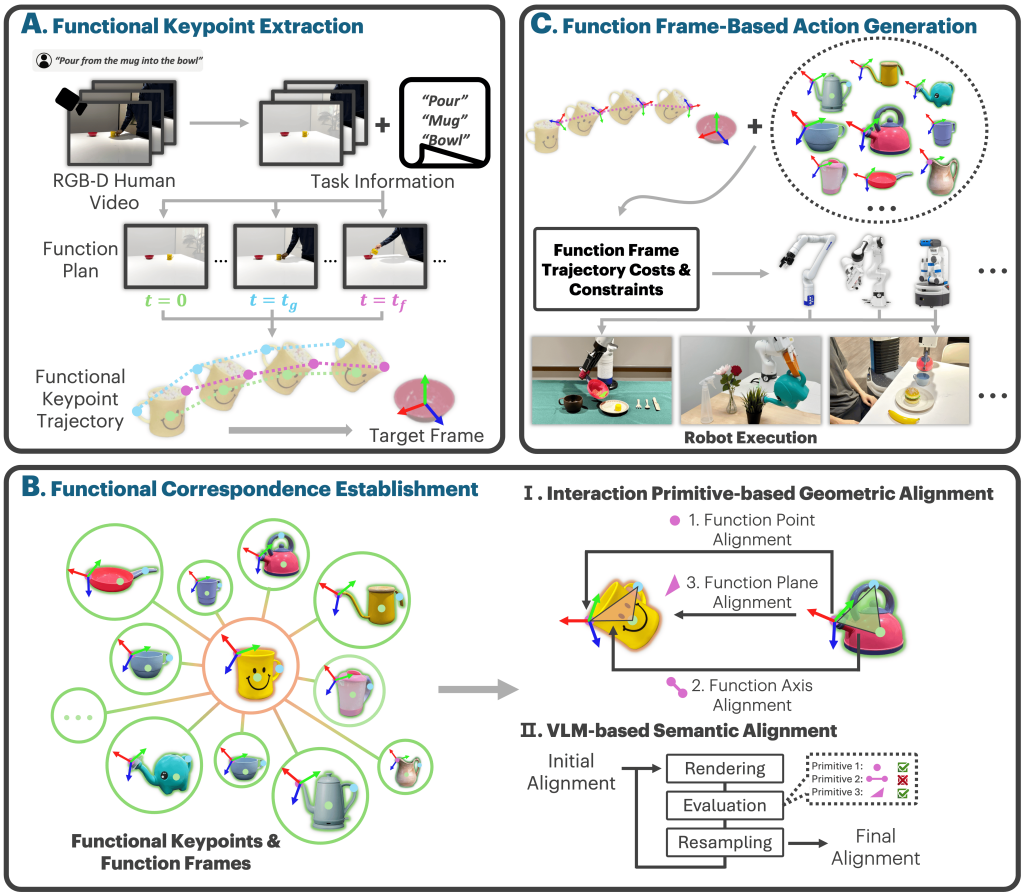


Figure 2: **Overview of MimicFunc Pipeline.** MimicFunc consists of three stages: (1) Functional keypoint extraction from human video, (2) Functional correspondence establishment with function frame, and (3) Function frame-based action generation.

represents the 6-DoF end-effector pose at timestep t, where $R_t \in \mathrm{SO}(3)$ and $T_t \in \mathbb{R}^3$ denote 3D orientation and translation, respectively.

3.2 Functional Keypoint Extraction from Human Video

Function Plan Generation with Keyframe Discovery. Since a tool manipulation task involves multiple stages, MimicFunc first generates a function plan with keyframe discovery to guide the robot's execution. Specifically, a function plan includes three stages: (1) the initial keyframe I_0 ($t = 0$), where the tool and target are in their initial states; (2) the grasping keyframe I_g ($t = t_g$), where the hand grasps the tool; and (3) the function keyframe I_f ($t = t_f$), where the tool interacts with the target. These keyframes satisfy the temporal constraint $0 < t_g < t_f < N - 1$. We use VideoCLIP [28] to discover these keyframes by computing similarity between video frames and predefined keyframe descriptions. For long-horizon tasks, MimicFunc generates a high-level task plan by chaining multiple function plans.

Functional Keypoint Extraction. MimicFunc detects 3D functional keypoints, representing a structured functional abstraction of the tool, using keyframes from the function plan and tracks their motions. For grasp point detection, MimicFunc first uses HaMeR [29] to reconstruct the hand mesh using I_g. The grasp point is then determined as the center of the intersection between the fingertip region and the tool. The center point is defined as the 3D bounding box center of the tool in I_0. Detecting the function point is non-trivial, as the tool may not physically contact the target (e.g., pouring) and requires commonsense knowledge about tool usage. To address this, MimicFunc employs mark-based visual prompting [27] to identify the function point on I_f (similar to [23]) and then projects the point into 3D space. To extract their motions, MimicFunc first estimates the tool's relative transformations between consecutive timesteps using CoTracker [30] and then computes the

3D functional keypoint trajectory $\{K_H^t\}_{t=0}^{N-1} = \{[p_{\text{func}}^t, p_{\text{grasp}}^t, p_{\text{center}}^t]\}_{t=0}^{N-1}$, where $p \in \mathbb{R}^3$ and H denotes human. To ensure that the extracted motion is independent of the absolute positions of the tool and target, MimicFunc transforms 3D elements from the camera frame to the target (object) frame by estimating the target object's pose relative to the camera. Unless otherwise specified, all 3D elements are represented in this relative target frame.

3.3 Functional Correspondence Establishment with Function Frame

Functional Keypoint Transfer. To construct function frames, MimicFunc first transfers keypoints from the demonstration tool to the test tool in a coarse-to-fine manner. It first performs in-context visual prompting to propose coarse regions for both function and grasp points, using p_{func}^0 and p_{grasp}^0 as references. The decision to propose regions rather than directly predict points stems from two factors: (1) VLMs lack point-level correspondence, and (2) they may produce discrete point predictions that do not align with the ideal keypoint locations. Then, a dense semantic correspondence model [31] with learned geometric priors is used to accurately transfer the keypoints to their corresponding regions, resulting in q_{func}^0 and q_{grasp}^0. Similarly, the test tool can be functionally abstracted as $K_R^0 = [q_{\text{func}}^0, q_{\text{grasp}}^0, q_{\text{center}}^0]$, where $q \in \mathbb{R}^3$ and the R denotes the robot.

Function Frame Construction. Based on two sets of keypoint abstractions, MimicFunc constructs function frames to represent function-centric spatiotemporal patterns of tool manipulation. Specifically, given K_H^t and K_R^t, the function frames Π_H^t and Π_R^t are constructed as follows. For Π_H^t, the origin is placed at the function point p_{func}^t, where the interaction between the tool and target occurs. The orientation is defined by an orthonormal basis constructed from K_H^t, where the unit vector from the center to the function point, $\mathbf{v}_H^t$, serves as the principal axis, referred to as the function axis, which provides a stable, reproducible directional cue reflecting how the tool operates. The same procedure is repeated to construct Π_R^t, with q_{func}^t as the origin and the function axis $\mathbf{v}_R^t$ as the principal axis. More details on function frame construction are available in Appendix A.2.

Function Frame Alignment. To transfer functional intent across different tools, MimicFunc aligns the spatiotemporal patterns encoded in function frames. In this section, we specifically focus on function keyframe alignment $\mathbf{\Pi}_{\text{func}}$, which defines the desired test tool state at timestep t_f, denoted as $\mathbf{\Pi}_R^{t_f}$. Aligning this critical state preserves the core functionality of human behavior, while the remaining motion can be flexibly optimized based on task context, as discussed in the following section. Function frame alignment is divided into two stages: (1) an initial stage for interaction primitive-based geometric alignment and (2) a refinement stage for VLM-based semantic alignment.

Inspired by [22, 25], MimicFunc first establishes a coarse alignment by enforcing function-relevant geometric constraints on function frames. These constraints operate on three types of interaction primitives, each corresponding to a physically meaningful spatial element critical for manipulation:

1. **Interaction primitive 1: point.** Function point alignment ensures that the interaction occurs at the intended location on the test tool by aligning q_{func}^0 with $p_{\text{func}}^{t_f}$.
2. **Interaction primitive 2: axis.** Function axis alignment ensures the proper operational direction for executing function-specific actions (e.g., tilting for pouring) by aligning $\mathbf{v}_R^0$ with $\mathbf{v}_H^{t_f}$.
3. **Interaction primitive 3: plane.** Function plane alignment preserves orientation by aligning the normal vectors of Π_R^0 and $\Pi_H^{t_f}$.

Each constraint is represented by a SE(3) transformation. Despite satisfying the geometric constraints derived above, the resulting interaction may still be functionally invalid, primarily due to (1) inaccurate perception and (2) structural differences between tools. To improve the robustness and adaptability, we further incorporate a VLM-based state evaluator for semantic refinement, similar to [32]. Specifically, MimicFunc first renders the predicted function keyframe interaction by back-projecting the combined point cloud of the test tool and the target onto the camera plane. It then prompts the VLM to evaluate whether the predicted state is functionally valid. If deemed valid, the alignment is accepted for action generation. Otherwise, the VLM sequentially checks each primitive to automatically identify those contributing to failure. Guided by this feedback, MimicFunc

uniformly resamples candidate (points or axes) around the initial constraint and iteratively repeats the process until a valid alignment is found. The alignment process is illustrated in Figure 2(b).

3.4 Function Frame-Based Action Generation

To enable a functionally consistent rollout of the intended behavior, MimicFunc computes the complete function frame trajectory for the test tool, formulated as a constrained optimization problem, using the human demonstration as the reference:

$$\min_{\{\mathbf{\Pi}_R^t\}_{t=0}^{N-1}} \sum_{t=0}^{N-1} \left(\|q_{\text{func}}^t - p_{\text{func}}^t\|_2^2 + \|\text{Log}(\mathbf{R}_R^t (\mathbf{R}_H^t)^\top)\|^2 \right) \quad \text{s.t.} \quad \mathbf{\Pi}_R^0 = \mathbf{\Pi}_{\text{init}},\ \mathbf{\Pi}_R^{t_f} = \mathbf{\Pi}_{\text{func}}$$

where $\text{Log} : \text{SO}(3) \to \mathbb{R}^3$ [33], and $\mathbf{R}^t$ denotes the rotation matrix derived from the function frame. The constraints $\mathbf{\Pi}_{\text{init}}$ and $\mathbf{\Pi}_{\text{func}}$ represent the initial and function keyframe alignments, respectively. This optimization framework provides a flexible mechanism for enforcing semantic-geometric alignment between function frames and can be extended to incorporate additional terms such as trajectory smoothness and collision avoidance. Implementation details are provided in Appendix A.3.

For robot execution, the test function frame trajectory is first transformed into the robot base frame. Then, MimicFunc samples a 6-DoF grasp pose around q_{grasp}^0 on the test tool. The robot end-effector trajectory τ_R is subsequently computed and executed.

4 Experiments

The experimental section aims to answer the following questions: (1) How well does MimicFunc generalize from a single human video to novel tools? (2) How does MimicFunc perform compared to existing methods? (3) How does MimicFunc perform in long-horizon tasks? (4) Can the rollout trajectories generated by MimicFunc be leveraged to train visuomotor policies?

4.1 Experimental Setup

Baselines. We compare MimicFunc against the following baselines: (1) **DINOBOT** [5], which uses DINOv2 [34] to perform semantic feature extraction and correspondence. (2) **DITTO** [4], which employs LOFTR [35] for local feature matching. (3) **ORION** [6], which establishes geometric correspondences with point cloud global registration. We adopt the original correspondence implementations of these baselines while keeping the low-level execution consistent with MimicFunc.

Task Description. We evaluate each method on five functions: `Pour`, `Cut`, `Scoop`, `Brush`, and `Pound`. A task is defined by pairing a function with a tool and a target. For each function, we design five tasks using different tools, divided into three levels of generalization: (1) spatial generalization, (2) instance generalization, and (3) category generalization.

Experimental Protocol. Each method is evaluated on 25 tasks across five functions, with 10 trials per task. The detailed task success conditions are described in Appendix A.1. The average success rate is used as the evaluation metric. Test objects are randomly initialized within the intersection of the camera view and the robot workspace.

4.2 Experimental Results

Quantitative Comparison to Baselines. The quantitative results are reported in Figure 3. For each function, the first object is used to evaluate spatial generalization, the next two evaluate instance-level generalization, and the final two evaluate category-level generalization. All methods perform reasonably well in spatial generalization, achieving success rates above 70%. However, all baselines exhibit significant performance drops (from 20% to 40%) when generalizing to novel tool instances and categories, especially for those with substantial intra-function variations. Among all baselines, ORION relies solely on geometric features, rendering it ineffective at handling large intra-function

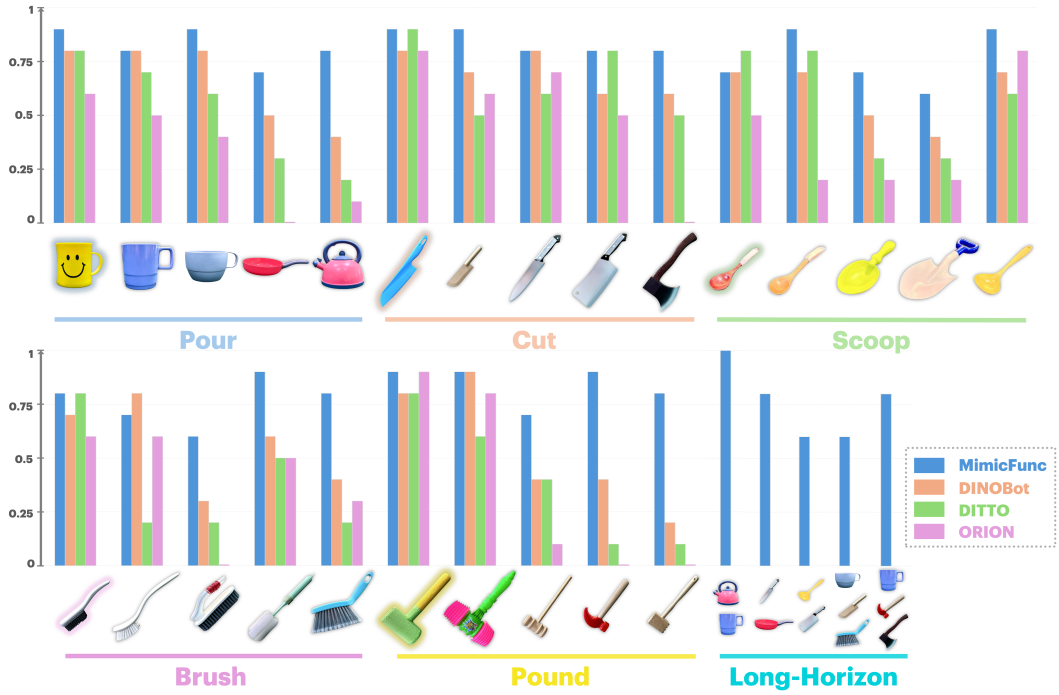


Figure 3: Quantitative comparison to baselines. Highlighted tools are used in human videos.

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. In contrast, MimicFunc significantly outperforms all baselines, achieving a high success rate of 79.5% across five functions for novel tool generalization. Figure 5 visualizes the qualitative results of real-robot executions.

Evaluation on Long-Horizon Tasks. We evaluate the performance of MimicFunc on long-horizon tasks involving multiple sequential steps. MimicFunc first generates a high-level task plan by chaining multiple function plans and then executes them sequentially. As reported in Table 1, MimicFunc achieves a 76.0% task success (TS) rate, slightly lower than the single-step setting, and an 80% step completion (SC) rate. The primary challenge arises from the limited reachability of the single-arm manipulator in larger object layouts. Nonetheless, by explicitly representing skills as function frame trajectories, MimicFunc can be integrated with Task and Motion Planning frameworks [36] to handle more complex long-horizon tasks with geometric constraints.

Task	P-P	C-P	S-C	P-C-B	P-O-C	Overall
TS	5/5	4/5	3/5	3/5	4/5	**76.0%**
SC	10/10	8/10	7/10	10/15	13/15	**80.0%**

Table 1: Quantitative results on long-horizon tasks. Abbreviations: P = `Pour`, C = `Cut`, S = `Scoop`, B = `Brush`, O = `Pound`.

Evaluation on Data Generation for Visuomotor Policy Training. To support the claim that the rollout data generated by MimicFunc can be leveraged for visuomotor policy training, we conduct experiments using the BC method ACT [15] on `Pour`. We first collect 50 teleoperation demonstrations to train ACT. As shown in Figure 4, ACT exhibits limited generalization to novel instances and categories. To enhance its performance, we deploy MimicFunc on these novel objects over randomly initialized layouts, automatically generating 30 rollouts per object using only a single human video. Successful trajectories are collected to train

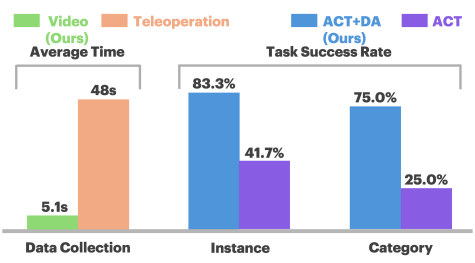


Figure 4: Performance evaluation of visuomotor policy training.

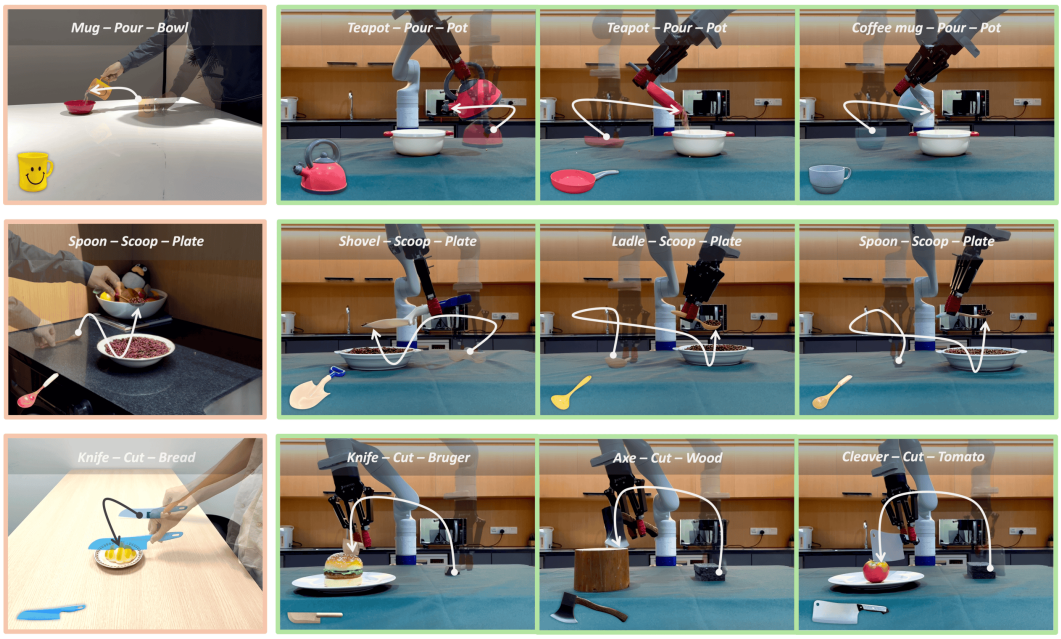


Figure 5: Visualization of grasping and function keyframes of human demonstrations and robot rollouts for `Pour`, `Scoop`, and `Cut`.

ACT. As illustrated in Figure 4, ACT+DA (data augmentation) improves performance on these instances and categories by 41.6% and 50.0%, respectively, using high-quality, consistently generated data from MimicFunc without requiring labor-intensive teleoperation data collection for novel objects. Moreover, this approach surpasses the variability and limited precision typically observed in human teleoperation. In terms of data collection efficiency, each teleoperation demonstration takes approximately 48 seconds, while MimicFunc requires only 5.1 seconds on average to capture a human video. These results demonstrate the potential of MimicFunc as a scalable and efficient data generator for visuomotor policy learning.

Ablation Study. We conduct an ablation study on the functional keypoint transfer strategy. Three strategies are evaluated: (1) Demo+VLM+DSC (ours); (2) Demo+DSC, which relies solely on a dense semantic correspondence model for keypoint transfer, following the approach in Robo-ABC [37]; and (3) VLM only, which directly prompts the VLM to propose keypoints in a zero-shot manner, as done in ReKep [24]. As shown in Figure 6, the proposed strategy consistently outperforms ablated versions, demonstrating that (1) the dense semantic correspondence model alone struggles with large intra-function variations, and (2) incorporating human demonstrations as references for VLMs significantly improves keypoint localization accuracy.

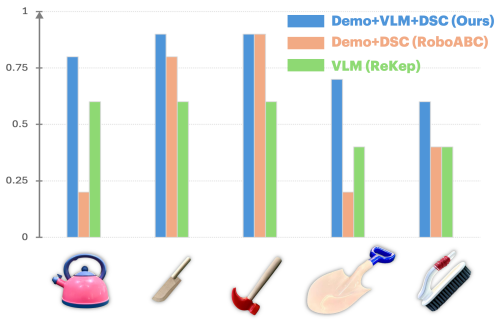


Figure 6: Ablation study results.

5 Conclusion

In this work, we present MimicFunc, a method for imitating tool manipulation from a single human video. At the core of MimicFunc is the ability to establish functional correspondences with function frame. This enables robots to generalize the skill from a single human video to novel tools despite significant intra-function variations. Furthermore, leveraging MimicFunc's one-shot generalization capability, the generated rollouts can be used to train visuomotor policies without requiring labor-intensive teleoperation data collection for novel objects.

6 Limitations and Future Work

Despite the promising results, several limitations remain that point to directions for future work. (1) MimicFunc currently relies on RGB-D input, which limits its applicability in directly leveraging RGB Internet human videos. Extending MimicFunc to support RGB videos could further enhance flexibility and reduce the human effort required for data collection. A potential solution is incorporating monocular depth estimation models, such as Depth Anything, to infer depth information from RGB inputs. (2) Although we have demonstrated the potential of data generation for visuomotor policy training, the current system remains limited in scope. Future work will aim to extend MimicFunc into a complete system in simulated environments capable of efficiently generating diverse data from a single human demonstration for visuomotor policy learning. (3) Complex tool manipulation tasks may involve bimanual coordination, whereas the current implementation only considers single-handed manipulation. Future work will extend MimicFunc to support dual-arm or even multi-fingered coordination to handle more complex manipulation scenarios.

7 Acknowledgement

We thank Linfeng Li (NUS) for helpful discussion and technical support. This work was supported in part by the Shenzhen Science and Technology Program (No. SGDX20240115111759002) and in part by Meituan.

References

[1] A. O'Neill, A. Rehman, A. Maddukuri, A. Gupta, A. Padalkar, A. Lee, A. Pooley, A. Gupta, A. Mandlekar, A. Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.

[2] C. Cheang, S. Chen, Z. Cui, Y. Hu, L. Huang, T. Kong, H. Li, Y. Li, Y. Liu, X. Ma, et al. Gr-3 technical report. *arXiv preprint arXiv:2507.15493*, 2025.

[3] P. Vitiello, K. Dreczkowski, and E. Johns. One-shot imitation learning: A pose estimation perspective. In *Conference on Robot Learning*, pages 943–970. PMLR, 2023.

[4] N. Heppert, M. Argus, T. Welschehold, T. Brox, and A. Valada. Ditto: Demonstration imitation by trajectory transformation. *arXiv preprint arXiv:2403.15203*, 2024.

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

[6] Y. Zhu, A. Lim, P. Stone, and Y. Zhu. Vision-based manipulation from single human video with open-world object graphs. *arXiv preprint arXiv:2405.20321*, 2024.

[7] J. Li, Y. Zhu, Y. Xie, Z. Jiang, M. Seo, G. Pavlakos, and Y. Zhu. Okami: Teaching humanoid robots manipulation skills through single video imitation. In *8th Annual Conference on Robot Learning*.

[8] O. Biza, S. Thompson, K. R. Pagidi, A. Kumar, E. van der Pol, R. Walters, T. Kipf, J.-W. van de Meent, L. L. Wong, and R. Platt. One-shot imitation learning via interaction warping. In *Conference on Robot Learning*, pages 2519–2536. PMLR, 2023.

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

[10] J. Zhu, Y. Ju, J. Zhang, M. Wang, Z. Yuan, K. Hu, and H. Xu. Densematcher: Learning 3d semantic correspondence for category-level manipulation from a single demo. In *The Thirteenth International Conference on Learning Representations*.

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

[12] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[13] H. Ravichandar, A. S. Polydoros, S. Chernova, and A. Billard. Recent advances in robot learning from demonstration. *Annual review of control, robotics, and autonomous systems*, 3 (1):297–330, 2020.

[14] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, page 02783649241273668, 2023.

[15] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[16] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu. 3d diffusion policy. *arXiv preprint arXiv:2403.03954*, 2024.

[17] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine. One-shot visual imitation learning via meta-learning. In *Conference on robot learning*, pages 357–368. PMLR, 2017.

[18] Y. Duan, M. Andrychowicz, B. Stadie, O. Jonathan Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba. One-shot imitation learning. *Advances in neural information processing systems*, 30, 2017.

[19] T. Yu, C. Finn, S. Dasari, A. Xie, T. Zhang, P. Abbeel, and S. Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. *Robotics: Science and Systems XIV*, 2018.

[20] M. Gleicher. Retargetting motion to new characters. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 33–42, 1998.

[21] K. Hu, C. Ott, and D. Lee. Online human walking imitation in task and joint space based on quadratic programming. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3458–3464. IEEE, 2014.

[22] L. Manuelli, W. Gao, P. Florence, and R. Tedrake. kpam: Keypoint affordances for category-level robotic manipulation. In *The International Symposium of Robotics Research*, pages 132–157. Springer, 2019.

[23] F. Liu, K. Fang, P. Abbeel, and S. 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*.

[24] W. Huang, C. Wang, Y. Li, R. Zhang, and L. Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. In *8th Annual Conference on Robot Learning*.

[25] J. Gao, Z. Tao, N. Jaquier, and T. Asfour. K-vil: Keypoints-based visual imitation learning. *IEEE Transactions on Robotics*, 2023.

[26] J. Gao, X. Jin, F. Krebs, N. Jaquier, and T. 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.

[27] S. Nasiriany, F. Xia, W. Yu, T. Xiao, J. Liang, I. Dasgupta, A. Xie, D. Driess, A. Wahid, Z. Xu, et al. Pivot: Iterative visual prompting elicits actionable knowledge for vlms. In *Forty-first International Conference on Machine Learning*.

[28] H. Xu, G. Ghosh, P.-Y. Huang, D. Okhonko, A. Aghajanyan, F. Metze, L. Zettlemoyer, and C. Feichtenhofer. Videoclip: Contrastive pre-training for zero-shot video-text understanding. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6787–6800, 2021.

[29] G. Pavlakos, D. Shan, I. Radosavovic, A. Kanazawa, D. Fouhey, and J. Malik. Reconstructing hands in 3d with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9826–9836, 2024.

[30] N. Karaev, I. Rocco, B. Graham, N. Neverova, A. Vedaldi, and C. Rupprecht. Cotracker: It is better to track together. In *European Conference on Computer Vision*, pages 18–35. Springer, 2025.

[31] J. Zhang, C. Herrmann, J. Hur, L. Polania Cabrera, V. Jampani, D. Sun, and M.-H. Yang. A tale of two features: Stable diffusion complements dino for zero-shot semantic correspondence. *Advances in Neural Information Processing Systems*, 36, 2024.

[32] M. Pan, J. Zhang, T. Wu, Y. Zhao, W. Gao, and H. Dong. Omnimanip: Towards general robotic manipulation via object-centric interaction primitives as spatial constraints. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 17359–17369, 2025.

[33] J. Sola, J. Deray, and D. Atchuthan. A micro lie theory for state estimation in robotics. *arXiv preprint arXiv:1812.01537*, 2018.

[34] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *Transactions on Machine Learning Research Journal*, pages 1–31, 2024.

[35] J. Sun, Z. Shen, Y. Wang, H. Bao, and X. Zhou. Loftr: Detector-free local feature matching with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8922–8931, 2021.

[36] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling. Pddlstream: Integrating symbolic planners and blackbox samplers via optimistic adaptive planning. In *Proceedings of the international conference on automated planning and scheduling*, volume 30, pages 440–448, 2020.

[37] Y. Ju, K. Hu, G. Zhang, G. Zhang, M. Jiang, and H. Xu. Robo-abc: Affordance generalization beyond categories via semantic correspondence for robot manipulation. In *European Conference on Computer Vision*, pages 222–239. Springer, 2024.

A Appendix

A.1 Experimental Setup and Results

A.1.1 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 bristles.

A.1.2 Examples of Human Demonstrations

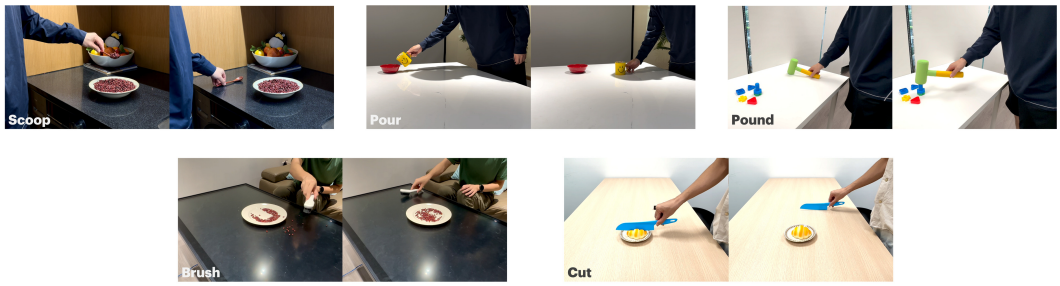


Figure 7: Visualization of grasping and function keyframes of human demonstrations.

A.1.3 Failure Analysis

The modular design of MimicFunc 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 frame alignment, (2) functional keypoint transfer, (3) trajectory generation, (4) grasping, and (5) others (e.g., segmentation, detection).

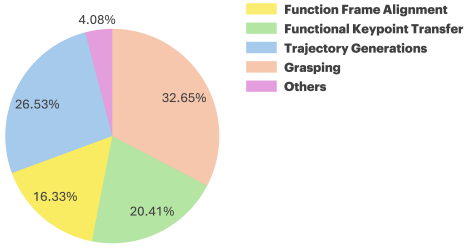


Figure 8: Failure analysis of system components.

The primary failures arise from (4) and (3). Grasping failures often arise from the inherent constraints of certain gripper types, especially rigid or underactuated designs. These grippers may lack the adaptability required to conform to diverse object geometries or provide sufficient contact stability, leading to issues such as tool flipping or slipping during tasks. Such failures are primarily due to unstable contact between the tool and the gripper, preventing the robot from completing its intended actions. Failures in trajectory generation primarily result from unexpected contact between the tool and the target, particularly in contact-rich tasks (e.g., "*use scrubber to brush the plate*"). These tasks require precise force application and adaptability to varying contact conditions. 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. Function frame alignment errors are mainly attributed to inaccurate depth information of the functional keypoints. Empirically, the functional correspondences are well established with accurate 3D functional keypoint locations. Ensuring precise depth sensing and calibration can significantly reduce these alignment errors.

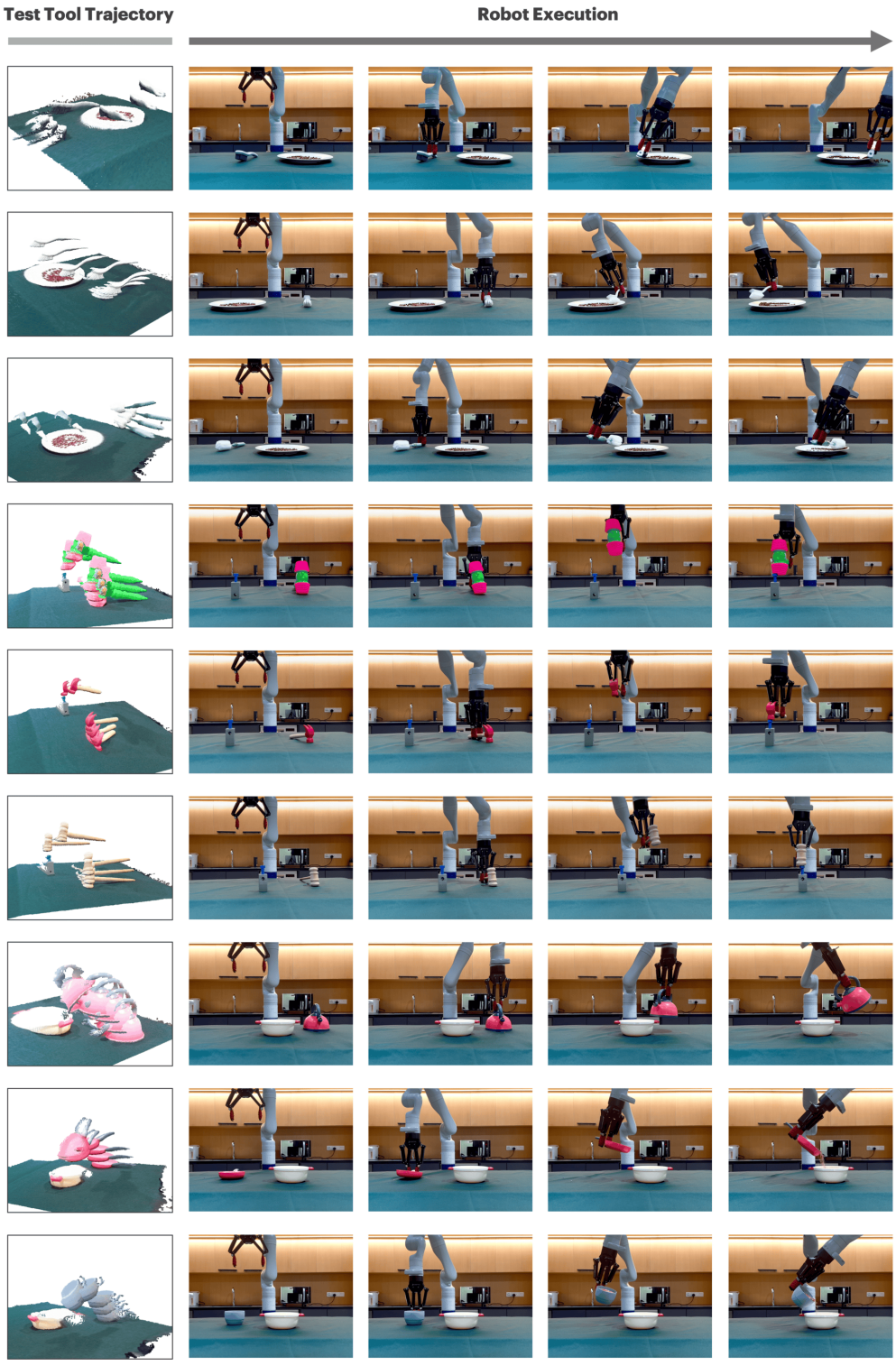
A.1.4 Target Frame Detection

To ensure that the tool motion is independent of the absolute positions of both the tool and the target, MimicFunc transforms all 3D elements from the camera coordinate frame into the target object's coordinate frame. This transformation requires estimating the target object's pose relative to the camera, which our method currently performs without relying on pre-existing mesh models. Instead, the target frame is estimated on the fly from the observed scene.

Specifically, we first acquire the target object's segmented point cloud from the camera. We then compute its principal axes via Principal Component Analysis (PCA) to determine the dominant orientation. The origin of the target frame is set to the center of the object's 3D bounding box, computed from the point cloud. Among the principal axes, the one most closely aligned with the estimated surface normal is assigned as the z-axis, ensuring that the frame is physically consistent with the object's geometry. The remaining axes are chosen to form a right-handed coordinate system, preserving orthogonality.

This procedure produces a stable, object-centered reference frame that remains robust to variations in both object position and camera viewpoint. Moreover, it can be seamlessly integrated with existing pose estimators (e.g., FoundationPose) to further improve robustness and adaptability, ensuring consistent target frame detection across diverse tasks, object geometries, and environments.

A.1.5 Qualitative Results



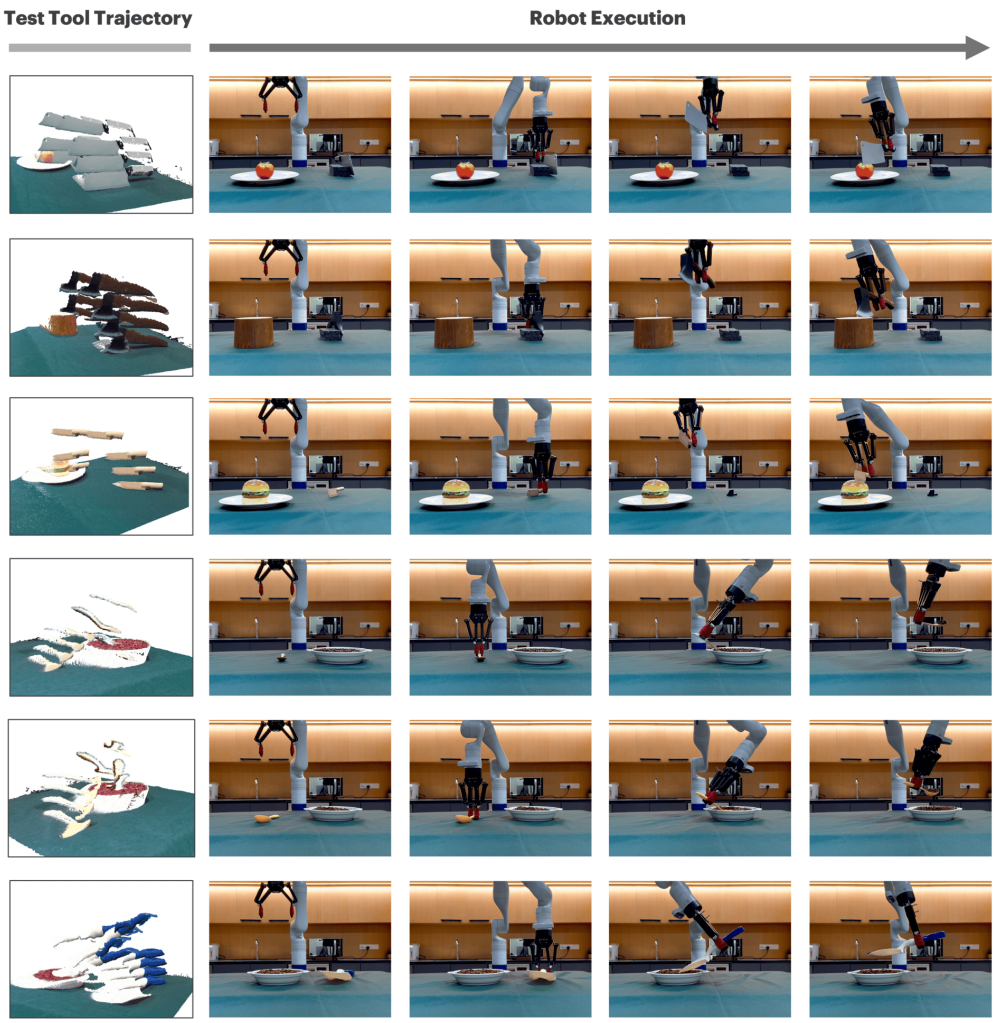


Figure 9: Qualitative results of real-robot executions.

A.2 Function Frame Construction and Alignment

A.2.1 Function Keypoint Transfer

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

Algorithm 1 Functional Keypoint Transfer.

Input:
Demo functional keypoints $K_H^0 = [p^0_{\text{func}}, p^0_{\text{grasp}}, p^0_{\text{center}}]$, Initial keyframe I_0, Robot observation o_R, Test tool mask M,
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 $K_R^0 = [q^0_{\text{func}}, q^0_{\text{grasp}}, q^0_{\text{center}}]$

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

In addition to the real-robot experiments, we compare the performance of different functional keypoint transfer strategies from a perception perspective, focusing 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 (Robo-ABC), which relies solely on a dense semantic correspondence model for functional keypoint transfer, following the approach in Robo-ABC;
- VLM (ReKep), which directly prompts the VLM to propose functional keypoints in a zero-shot manner, as done in ReKep.

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.

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.

Quantitative results. The quantitative results of function point transfer are presented in Table A.2.1. The proposed Demo+VLM+DCS consistently outperforms the ablated strategies in both AKD and

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	56.09	15.56	36.22	52.67
Demo+VLM+DSC	**18.54**	**51.33**	**85.78**	**94.44**

Table 2: Quantitative results of function point transfer

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 highlights the importance of using demonstrations as in-context references for the keypoint proposal.

A.2.2 Function Frame Construction

Function frames Π_R^t and Π_H^t are constructed based on the 3D functional keypoints $K_H^t = [p_{\text{func}}^t, p_{\text{grasp}}^t, p_{\text{center}}^t]$ and $K_R^t = [q_{\text{func}}^t, q_{\text{grasp}}^t, q_{\text{center}}^t]$, respectively. Π_H^t is defined by the following elements:

1. **Function axis**
   - Definition:
   $$\mathbf{v}_H^t = \frac{p_{\text{func}}^t - p_{\text{center}}^t}{\|p_{\text{func}}^t - p_{\text{center}}^t\|}$$
   - Description: $\mathbf{v}_H^t$ is a normalized vector that defines the function axis. It points from the center point p_{center}^t to the function point p_{func}^t at t. This axis represents the principal direction along which the function operates.
2. **Grasp vector**
   - Definition:
   $$\mathbf{u}_H^t = \frac{p_{\text{grasp}}^t - p_{\text{func}}^t}{\|p_{\text{grasp}}^t - p_{\text{func}}^t\|}$$
   - Description: $\mathbf{u}_H^t$ is a normalized vector that points from the function point p_{func}^t to the grasp point p_{grasp}^t at t.
3. **Unit normal vector**
   - Definition:
   $$\mathbf{n}_H^t = \frac{\mathbf{u}_H^t \times \mathbf{v}_H^t}{\|\mathbf{u}_H^t \times \mathbf{v}_H^t\|}$$
   - Description: $\mathbf{n}_H^t$ is the unit normal vector of the function plane $\mathcal{P}_H^t$.
4. **Function plane**
   - Definition:
   $$\mathcal{P}_H^t : (\mathbf{p} - p_{\text{func}}^t) \cdot \mathbf{n}_H^t = 0$$
   - Description: $\mathcal{P}_H^t$ is defined by the function point and its normal vector, describing the tool's spatial configuration at t.

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

A.2.3 Function Frame Alignment

To enhance the robustness and adaptability of MimicFunc, we introduce a VLM-based state evaluator for semantic refinement in the second stage of function frame alignment. In this stage, MimicFunc first renders the predicted function keyframe interaction by back-projecting the combined point cloud of the test tool and target object onto the camera plane. The rendered scene is then provided as input to the VLM, which assesses whether the predicted state is functionally valid.

If the state is deemed valid, the alignment is accepted for downstream action generation. Otherwise, the VLM sequentially inspects each primitive to automatically pinpoint those responsible for the failure. Using this feedback, MimicFunc uniformly resamples candidate points or axes around the initial constraint and iteratively repeats the evaluation process until a valid alignment is achieved. Figure 10 illustrates an example of intermediate rendering results during function axis refinement, where blue denotes the initial alignment and green indicates the refined result.

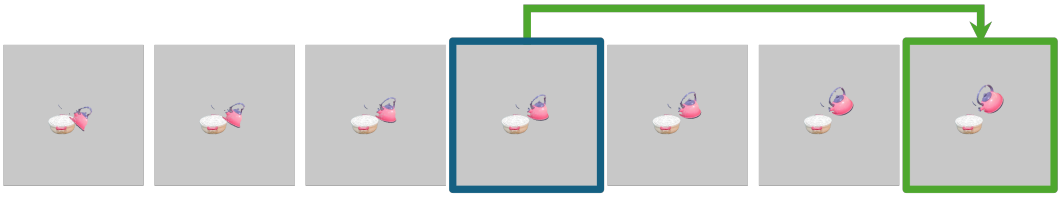


Figure 10: Intermediate rendering results of function axis refinement.

A.3 Function Frame-based Trajectory Generation

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

Trajectory Warping. Given a demonstration function frame trajectory $\{\mathbf{\Pi}_{\mathrm{H}}^t\}_{t=0}^{N-1}$ and its associated function point trajectory $\{p_{\text{func}}^t\}_{t=0}^{N-1}$, trajectory warping adapts these references to a new test scenario by leveraging geometric symmetries and relational transformations. The process consists of three main stages:

1. **Symmetry-Based Repositioning:**
   If the target object exhibits geometric symmetries (e.g., rotational symmetry about one of the principal axes), we exploit this property to reposition the demonstration so that the test tool can adopt a more feasible approach direction. Let $R_{\text{sym}} \in \mathrm{SO}(3)$ denote a symmetry rotation. Then, the demonstration function frames and points are transformed as:
   $$\mathbf{\Pi}_{\mathrm{H}}^t = R_{\text{sym}} \cdot \mathbf{\Pi}_{\mathrm{H}}^t, \quad p_{\text{func}}^t = R_{\text{sym}} \cdot p_{\text{func}}^t$$
2. **Function Frame Trajectory Pre-processing:**
   We pre-process the demonstration's function frame trajectory by first applying a rotation around one of the principal axes (e.g., x-, y-, or z-axis). The alignment angle θ is computed based on the angular difference between the initial function points of the demonstration and the test tool:
   $$\theta = \angle(p_{\text{func}}^0, q_{\text{func}}^0), \quad R_{\text{align}}(\theta) \in \mathrm{SO}(3)$$
   The aligned frame is then obtained by:
   $$\mathbf{\Pi}_{\text{align}}^t = R_{\text{align}}(\theta) \cdot \mathbf{\Pi}_H^t$$
3. **Function Frame Trajectory Transformation:**
   To account for differences in position and scale between the demonstration and the test tool, we apply a translation $\mathbf{t} \in \mathbb{R}^3$ and an optional scaling factor $s \in \mathbb{R}$:
   $$\mathbf{\Pi}_{\text{warp}}^t = s \cdot \mathbf{\Pi}_{\text{align}}^t + \mathbf{t}$$

Optimization Constraints and Costs. Beyond the trajectory cost and keyframe constraints detailed in the manuscript, we introduce the following enhancements:

- **Early Trajectory Cost Relaxation.** To encourage smoother transitions and allow flexibility during the approach phase, the trajectory cost is omitted for the initial 30% of the trajectory. This is particularly beneficial when the initial states of the demonstration and test tools differ significantly, as the primary interaction occurs later in the motion.
- **Velocity Constraint.** We constrain both translational and angular velocities of the test tool to ensure smooth motion and physical feasibility throughout the trajectory.
- **Collision Avoidance Constraint.** A minimum Euclidean distance is enforced between the test tool and the 3D bounding box of nearby obstacles, preventing collisions during execution.

We use `CasADi` for symbolic modeling and automatic differentiation, and solve the resulting nonlinear constrained optimization problem with `IPOPT`.

A.4 Data Generation for Visuomotor Policy Training

Data Generation. To acquire data for visuomotor policy training, we leverage MimicFunc to generate rollout trajectories for novel tools, without requiring labor-intensive teleoperation data collection for novel objects. The process begins with the robot randomly sampling object layouts, including the positions and orientations of the tool, within its workspace. The robot then places the tool at the sampled configuration. MimicFunc then generates a candidate motion to accomplish the task. After executing the rollout, the final scene is captured by the camera and evaluated using a VLM to determine task success. Only successful rollouts are retained to construct a demonstration dataset, which is used to train visuomotor policies capable of generalizing across diverse tools and object arrangements.

Policy Training. We experiment with two state-of-the-art behavioral cloning approaches: Action Chunking Transformer (ACT) and Diffusion Policy (DP), both of which utilize a DINOv2-pretrained ResNet-18 as the visual encoder backbone. ACT has demonstrated effectiveness in learning complex manipulation skills using transformer architectures. In our implementation, we adapt the standard ACT by incorporating RGB-D inputs from two viewpoints: a third-person camera and an in-hand camera. The policy receives the most recent frame from each camera and predicts absolute end-effector poses. This design enables the generation of long-horizon trajectories by predicting action chunks of 100 steps in a single forward pass, making it particularly suitable for tasks that demand precise global positioning. DP takes a generative approach using denoising diffusion probabilistic models. It receives two consecutive RGB-D frames as input and predicts the next 16 delta end-effector poses through an iterative denoising process over 100 inference steps. This approach excels at modeling complex, multimodal action distributions and is well-suited for tasks requiring smooth and continuous motion. The hyperparameters used for both policies are summarized in Table 3.

Data Quality. In addition to showcasing the capability of MimicFunc for efficient data generation, we conduct experiments to further evaluate the quality of the generated data. Specifically, we compare the performance of the ACT trained on two different data sources for the `Pour` task: (1) teleoperation-collected demonstrations, and (2) MimicFunc-generated demonstrations (50 samples). The results show that ACT trained on MimicFunc-generated data achieves a higher success rate (53.85%) compared to ACT trained on teleoperation data (46.15%). This performance gain supports our claim that MimicFunc produces more consistent and higher-quality data. In contrast, teleoperation data often suffers from variability, inconsistencies in execution, and imprecision due to human control limitations.

Parameter	ACT	DP
Visual Encoder	DINOv2-ResNet-18	DINOv2-ResNet-18
Action Representation	Absolute EE Pose	Delta EE Pose
Observation Horizon	1	2
Chunk Size	100	16
Hidden Dimension	512	–
Feedforward Dimension	3200	–
Encoder Layers	4	–
Decoder Layers	7	–
Attention Heads	8	–
Batch Size	64	64
Epochs	500	500
Learning Rate	1e-4	1e-4
Scheduler	Cosine Annealing	Cosine Annealing
KL Weight	10.0	–
Diffusion Timesteps	–	100
EMA Power	–	0.75

Table 3: Hyperparameters used for ACT and DP training. "–" indicates the parameter is not applicable.