FastUMI: A Scalable and Hardware-Independent Universal Manipulation Interface with Dataset

Zhaxizhuoma[1,2†], **Kehui Liu**[2,6†], **Jeff Guan**[2†], **Zhongjie Jia**[1,2†], **Ziniu Wu**[2,3†], **Xin Liu**[1,2†], Tianyu Wang[2,4*], Shuai Liang[1,2*], Pengan Chen[2,5*], Pingrui Zhang[2,4*], Haoming Song[1,2], Delin Qu[2,4], Dong Wang[2], Zhigang Wang[2], Nieqing Cao[7], Yan Ding[2,8†‡], Bin Zhao[2,6‡], Xuelong Li[9]

[1]Shanghai Jiao Tong University, [2]Shanghai AI Lab, [3]University of Bristol,
[4]Fudan University, [5]The University of Hong Kong,
[6]Northwestern Polytechnical University, [7]Xi'an Jiaotong-Liverpool University,
[8] Suzhou OneStar Robotics Co., Ltd., [9]Institute of AI, China Telecom Corp Ltd.
† ∗ Equal Contribution, ‡Project Leader, Project Website: https://fastumi.com/

Abstract: Real-world manipulation datasets for robotic arms remain scarce due to the high costs, rigid hardware dependencies, and complex setup procedures associated with existing data collection methods. We introduce **FastUMI**, a redesigned Universal Manipulation Interface (UMI) that addresses these challenges, enabling low-cost, scalable, and rapid deployment across heterogeneous platforms. FastUMI achieves this through: (i) hardware decoupling via extensive mechanical reengineering, which removes dependence on specialized robotic components while preserving a consistent visual perspective; (ii) replacement of complex visual–inertial odometry with a commercial off-the-shelf tracker, simplifying the software stack without compromising pose estimation accuracy; and (iii) the provision of an integrated ecosystem that streamlines data acquisition, automates quality control, and ensures compatibility with both standard and enhanced imitation-learning pipelines. To facilitate further research, we release an open-access dataset comprising over 15,000 real-world demonstrations spanning 24 tasks—constituting one of the most extensive UMI-like resources to date. Empirical evaluations show that FastUMI supports rapid deployment, reduces operational overhead, and delivers robust performance across diverse manipulation scenarios, advancing scalable data-driven robotic learning.

Keywords: Manipulation, Imitation Learning, Universal Manipulation Interface

1 Introduction

The scarcity of large-scale, high-quality, real-world interaction data remains a major bottleneck to progress in robotic manipulation, primarily due to challenges associated with efficient and scalable data collection methods [1, 2, 3, 4]. Current methods can be broadly categorized into teleoperation-based techniques [5, 6, 7, 8], vision-driven demonstrations [1, 9, 10, 11, 12], and sensor-enhanced interfaces [13, 14, 15]. While teleoperation enables precise data acquisition, it remains labor-intensive, costly, and constrained by the challenges of non-intuitive and task-specific human-to-robot mapping[16, 17, 18]. Vision-driven approaches can provide large-scale, low-cost data but typically lack the rich, fine-grained interaction dynamics essential for policy learning. In contrast, sensor-enhanced interfaces—exemplified by systems such as the Universal Manipulation Interface (UMI) [2]—offer a promising alternative. They directly capture diverse, multimodal signals that closely align with a robot's onboard sensory modalities, preserving fidelity and precision, while *enabling human demonstration data to be seamlessly transferred into robotic frameworks.* In doing so, they narrow the gap between human demonstrations and autonomous robotic execution, enabling rapid and high-quality data collection.

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

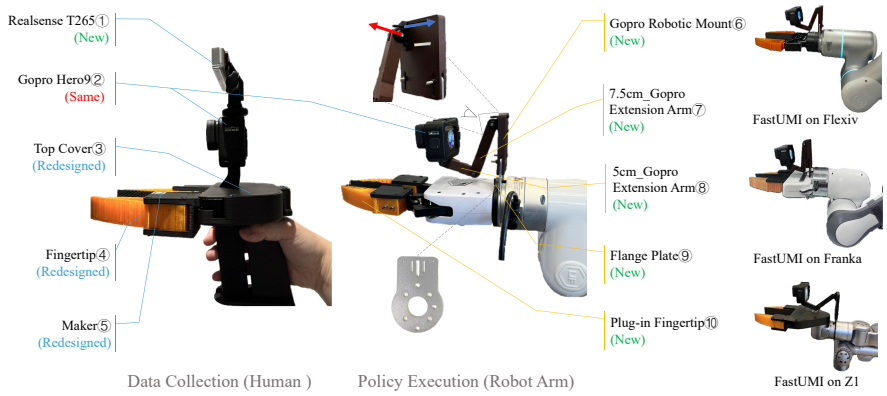

Figure 1: Physical prototypes of FastUMI. **Left**: The handheld device, *used to collect demonstration data from human operators*. **Middle**: A robot-mounted device, *used for executing learned policies on the robotic arm*, mirrors the handheld configuration. **Right**: FastUMI can be easily deployed on various robotic arms and grippers. Color coding is used to differentiate FastUMI's hardware from the original UMI. Component details are provided in Appendix 8.1.

While the UMI system addresses key challenges in human demonstration data collection and supports action policy learning in diverse scenarios, its *current* system design and implementation suffers from two key limitations. First, its tight coupling with specific robotic components (*e.g.*, the Weiss WSG-50 gripper) restricts adaptability and increases both financial and logistical burdens. The second limitation arises from the software framework, particularly the reliance on a GoPro-based VIO (Visual-Inertial Odometry) pipeline in conjunction with open-source SLAM algorithms [19]. Through experimental evaluation, we observe that the UMI system encounters difficulties in tasks that involve prolonged occlusions, such as hinged operations. As a result, the UMI software configuration struggles to maintain robust operation when visual signals are intermittently lost, thereby diminishing data quality and reducing its utility for subsequent learning tasks.

As a key contribution, our proposed FastUMI presents an extensive redesign structured around three primary objectives: 1) *Enhancing adaptability through hardware decoupling*. By removing strict dependencies on specific robotic components, our hardware design can be seamlessly integrated with a wide range of robotic arms and grippers, facilitating rapid deployment across diverse platforms. 2) *Improving efficiency with software-driven plug-and-play functionality*. Our software stack emphasizes immediate usability, requiring minimal configuration and user training. This design choice facilitates rapid data collection and significantly reduces operational complexity. 3) *Establishing a robust ecosystem to ensure data quality and algorithmic compatibility*. Our ecosystem is designed to support various imitation learning algorithms (*e.g.*, Action Chunking with Transformers (ACT) [16] and Diffusion Policy (DP) [20]) by providing essential data types, such as end-effector trajectory and joint trajectory. To address the distinctive data properties of FastUMI, we further introduce three refined algorithms—Smooth-ACT, PoseACT, and Depth-Enhanced DP—which enhance policy smoothness, cross-platform generalization, and precision under limited perceptual depth perception.

Experimental evaluations demonstrate that the redesigned system delivers an integrated, user-friendly solution that seamlessly aligns the handheld interface with robot-mounted equipment, effectively enabling efficient and scalable data acquisition for robotic learning. We open-source over **15,000 demonstration trajectories** collected in real-world settings across 24 everyday tasks, establishing our dataset as one of the most comprehensive UMI-like collections in terms of *task variety*.

2 Hardware-Centric Prototype Design

The FastUMI system embodies a hardware-centric redesign based on a decoupled design philosophy. This section introduces the guiding principles behind this approach and provides an overview of the hardware components. Aligned with the stated objectives, our hardware design must overcome several critical challenges. The first involves *decoupling the system from specific robotic hardware*. By designing mechanical components that seamlessly integrate with diverse robotic arms and grip-

pers—each varying in size, shape, and mechanical interface—we aim to minimize redesign efforts and configuration overhead. The second major challenge is *maintaining visual consistency between handheld and robot-mounted devices*, which is essential for effective policy transfer in robotic learning algorithms. Given the wide range of possible gripper dimensions, preserving uniform camera perspectives across different hardware setups becomes essential.

In addition, our design must *accommodate a wider variety of robot-mounted grippers*, moving beyond the parallel-jaw restriction and thereby broadening its applicability across various robotic platforms. Furthermore, *fast deployment* is a key requirement, so we strive for a plug-and-play solution that streamlines user setup, minimizing calibration and configuration efforts to promote widespread adoption. Finally, *ensuring high data quality* underlies the entire effort. Reliable hardware is crucial for obtaining accurate and consistent data, thus mitigating barriers in downstream learning tasks and enhancing overall system performance. To address these challenges, FastUMI adopts a **decoupled design philosophy** that underpins its hardware architecture. The system is systematically decoupled along three primary dimensions, which are summarized below. Further implementation details are provided in Appendix 8.2, 8.3, and 8.4.

- **Physical Decoupling**: Standardized interfaces and modular components enable seamless integration across robotic platforms, eliminating the need for extensive hardware-specific modifications.
- **Visual Consistency**: Uniform camera perspectives between handheld and robot-mounted configurations ensure that data acquired in one setting can be readily transferred to another without requiring extensive recalibration. This consistency allows data from human demonstrations to be directly applicable to robotic execution.
- **Operational Independence**: The system incorporates self-contained tracking and sensing modules, reducing reliance on external computational frameworks and ensuring robust performance across diverse deployment scenarios.

3 Software-Focused Framework

3.1 Raw Data Acquisition and Quality Assessment

FastUMI employs three core ROS nodes to record multimodal demonstration data. First, a *camera node* continuously streams wide-angle images (e.g., 1920×1080 at 60 fps) from a GoPro Camera. Second, a *tracking node* provides pose estimates from the T265 or RoboBaton MINI sensor at a higher rate (e.g., 200 Hz). Each pose is represented as $(x, y, z, q_x, q_y, q_z, q_w)$, where (x, y, z) represents the translation vector and (q_x, q_y, q_z, q_w) represents orientation in quaternion form. Finally, a *storage node* aggregates and synchronizes these streams in an *Hierarchical Data Format version 5 (HDF5)* file for subsequent processing. Because each sensor runs independently, the system is readily extensible to accommodate additional modalities (e.g., tactile sensors) by adding corresponding ROS nodes. FastUMI's reliance on the T265 not only improves reliability under partial occlusions but also removes the need for extensive calibrations and VIO parameter tuning, significantly accelerating deployment. Nonetheless, *it introduces more demanding dual-sensor synchronization and drift management requirements*, as discussed in Appendix 8.5 and 8.6. To ensure reliable demonstrations for downstream learning, we also implement a data quality assessment pipeline based on *sensor confidence checks* and *trajectory smoothness metrics*. Full implementation details and evaluation criteria are provided in Appendix 8.7.

3.2 Data Preparation for Training

To address the requirements of diverse imitation learning algorithms, we categorize trajectory data into two main types—**TCP trajectories** (both absolute and relative), and **joint trajectories**. At the software level, FastUMI facilitates seamless integration of various data formats and evolving algorithmic needs with minimal configuration.

The principal raw inputs are pose estimates from the T265 camera, given by $\mathbf{p}_i$ (position) and $\mathbf{R}_i$ (orientation) in the camera's local frame. The following additional information is also required: (i) The *known* robotic arm's Unified Robot Description Format (URDF). (ii) The *known* offset Δ_{c2g} from

the T265 camera center to the gripper center (expressed in the camera frame), as shown in Figure 2 (Left). (iii) The *known* pose $\left(\mathbf{p}_{b2g}, \mathbf{R}_{b2g}\right)$ of the gripper center in the robot base frame, as shown in Figure 2 (Right), where $\mathbf{p}_{b2g}$ is the position, and $\mathbf{R}_{b2g}$ is the rotation. We assume that the hand-held device motion precisely mirrors that of the robot end-effector. Under these conditions, the following trajectories can be derived: 1) *Absolute TCP trajectories*, 2) *Relative TCP trajectories*, and 3) *Absolute joint trajectories*. Full computational procedures are presented in Appendix 8.8.

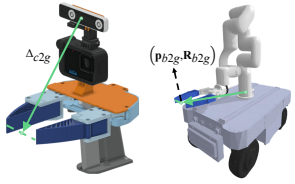

Figure 2: Illustration of offset Δ_{c2g} from T265 center to the gripper center, and the gripper center pose $\left(\mathbf{p}_{b2g}, \mathbf{R}_{b2g}\right)$ in the robot base frame.

Continuous Gripper Width Computation: We propose a marker-based method that *decouples software from the underlying mechanical structure*, thereby facilitating compatibility with diverse gripper designs. Specifically, we measure the pixel distance between ArUco markers [21] on the gripper jaws and map it linearly to the gripper's physical opening width. This approach obviates rigid hardware dependencies, reducing design constraints and streamlining integration of new or differently sized grippers. Further implementation details are provided in Appendix 8.9.

4 Algorithmic Adaptations for FastUMI

In earlier sections, we introduce how FastUMI attains hardware decoupling. This design choice lowers the cost and complexity of system deployment across heterogeneous platforms, including handheld and robot-mounted configurations. However, while hardware decoupling supports seamless data acquisition across various setups, it does not by itself address the **distinct policy-learning challenges arising from FastUMI's data distributions** (details in Section 4.1). Hence, hardware decoupling alone cannot fully realize FastUMI's potential. By incorporating data-driven refinements into baseline algorithms to accommodate FastUMI's unique data characteristics, we enable efficient multi-platform deployment with consistently high performance while also *laying the groundwork for more advanced methods in the future*.

4.1 Data Challenges with FastUMI

Compared to conventional third-person or fixed-base perspectives, FastUMI's wrist- or handheld-mounted viewpoints introduce several distinct data characteristics:

- **Close-up First-Person Perspective**: Cameras positioned near the end-effector capture detailed manipulation cues but offer *limited visibility of the full robotic arm*, increasing dependence on priors to maintain kinematic feasibility.
- **Variable Geometry and Scene Layout**: FastUMI's hardware-agnostic design generates heterogeneous data across different arm configurations, base frames, and environments, complicating efforts to achieve consistent policy learning.
- **Limited Depth Information**: Single-view fisheye images lack explicit three-dimensional spatial cues, making precise depth estimation difficult. Tasks demanding accurate positioning, including object alignment and gripper closure, are particularly vulnerable to errors when depth signals are absent.

To address these challenges, we present adaptations for two primary imitation learning algorithms: ACT and DP. These enhancements promote robust policy execution, ensure kinematic feasibility, and integrate depth-awareness for tasks that require higher precision.

4.2 Enhanced ACT for First-Person Perspectives

The standard ACT predicts *absolute joint trajectories* for the robotic arm, performing effectively in third-person or fixed-camera scenarios. However, under FastUMI's first-person wrist-mounted perspective, large portions of the robotic arm remain unseen, making ACT susceptible to producing **illicit joint configurations** during inference. These configurations can exhibit extreme end-effector orientations that violate kinematic constraints or diverge substantially from demonstration trajectories. Consequently, we introduce two targeted refinements to the original ACT to address visibility limitations inherent in first-person data.

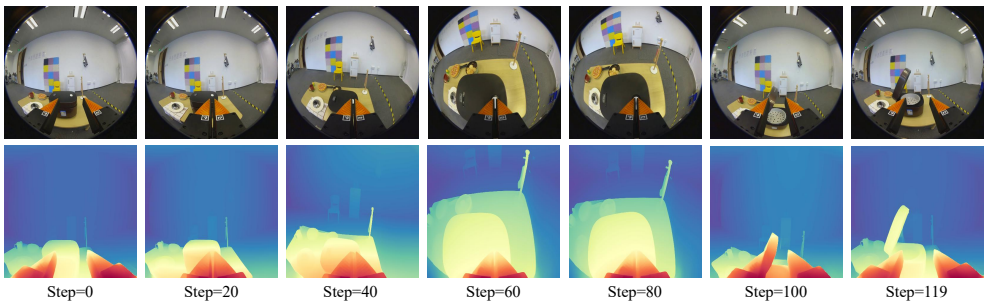

Figure 3: Illustration of the depth-mapping process from Step 0 to Step 119. The top row displays the cropped GoPro frames (rectified from their original circular format), and the bottom row presents corresponding depth estimations produced by Depth Anything V2.

1) Smooth-ACT: Local Temporal Smoothing. To address abrupt or infeasible joint transitions, we introduce Smooth-ACT, which enhances action continuity by incorporating a Gated Recurrent Unit (GRU) layer on top of the Transformer decoder [22, 23]. While the Transformer captures global spatiotemporal patterns, the GRU refines local continuity, smoothing sudden deviations between successive frames. During training, two action sequences are produced: $\hat{a}$ from the Transformer decoder and $\hat{a}_{\text{GRU}}$ from the GRU layer. Both are compared against ground truth actions with the loss function $\mathscr{L}$:

$$\mathscr{L} = \|\hat{a} - a\|_1 + \|\hat{a}_{\text{GRU}} - a\|_1 + \lambda \, \text{KL}\big(\mu, \log \sigma^2\big), \tag{1}$$

where $\text{KL}\big(\mu, \log \sigma^2\big)$ regularizes model outputs for stability. This hierarchical setup preserves the Transformer's capacity for global attention while enforcing local smoothing, thereby reducing kinematically invalid actions.

2) PoseACT: End-Effector Pose Prediction. Beyond mitigating trajectory discontinuities, we further enhance ACT's robustness by introducing PoseACT, a variant that replaces absolute joint predictions with an end-effector (TCP) pose representation. This formulation incorporates both absolute and relative motion trajectories, offering two key benefits: 1) **Platform Independence**: Expressing actions in terms of local end-effector movement reduces sensitivity to base-frame or arm-geometry variations, facilitating multi-platform policy transfer. 2) **Numerical Stability**: Relative trajectories generally show less variability, mitigating outlier effects and improving generalization to novel configurations. During inference, the policy outputs relative poses, which are subsequently mapped back to absolute joint angles via the robot's kinematic model. Our evaluations suggest this base-agnostic approach increases policy robustness and minimizes extreme joint commands, especially under limited first-person observations.

4.3 Depth-Enhanced Diffusion Policy

We apply DP from the original UMI (which includes relative TCP trajectory prediction and latency matching) to our robotic platform, and observe promising initial results. However, we identify a limitation: **the DP struggle with tasks requiring high precision in depth estimation.** For instance, it occasionally fails to accurately reach the target or prematurely closed the grippers. This reveals the inadequacy of the current policy when operating without depth information, especially in scenarios where precise spatial reasoning is essential. To address this issue, we incorporate depth information and utilize the cross-attention mechanism [23] to fuse features extracted from the depth and RGB inputs to improve the original DP, resulting in a variant, called **Depth-Enhanced DP**. While existing works incorporating depth into DP often rely on dedicated sensors to capture real-time depth data [24, 25], we aim to explore a *more lightweight and efficient approach* without adding hardware complexity or additional costs. Specifically, we adopt a post-processing strategy to generate depth maps. Using the open-source depth estimation tool Depth Anything V2 [26], we supplement each frame in our dataset with depth maps, as shown in Figure 3. Details are presented in Appendix 8.10.

4.4 Dynamic Error-Compensation Algorithm

Non-parallel-jaw grippers on robotic arms can shift the TCP as the jaws close. Because the jaws move inward, the effective TCP often translates along the gripper's local Z-axis, leading to misalignment

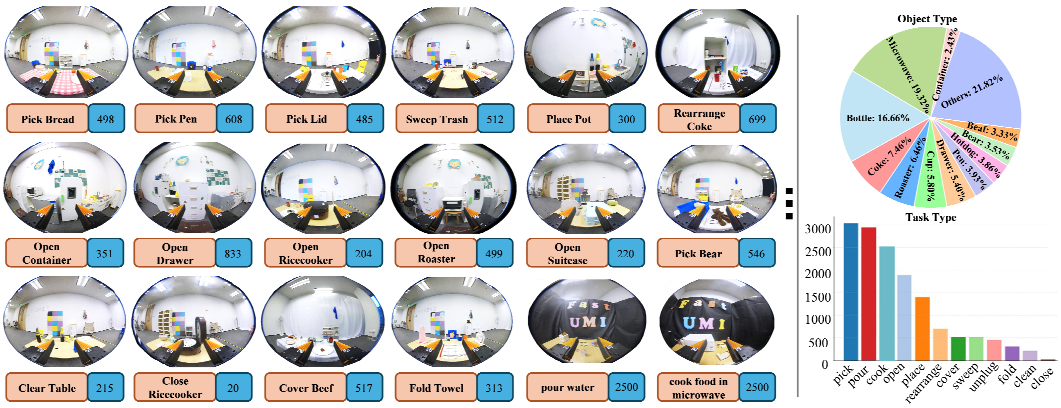

Figure 4: **Left**: Representative GoPro frames from the FastUMI dataset, with *orange* labels indicating specific tasks and *blue* numerals showing demonstration counts. **Right**: Two distribution plots—the **top** plot depicts the proportion of various tasks in the dataset, while the **bottom** plot shows the distribution of manipulation skills.

in tasks that require fine precision, such as picking up small objects. To mitigate these shifts, we propose a *dynamic error-compensation algorithm* that adjusts the commanded TCP in real time. It is structured in two principal stages; detailed procedures are provided in Appendix 8.11.

5 Open-Source Dataset

We present the FastUMI Dataset, consisting of **15,000** demonstration sequences, each containing synchronized GoPro video and end-effector trajectories captured in domestic settings. The dataset covers **24** tasks, **21** object categories, and **13** distinct manipulation skills. Each demonstration comprises approximately **120** $\sim$ **400** frames with most sequences spanning around 400 frames. Owing to its high frame rate and tightly coupled multi-sensor setup, FastUMI captures substantially denser and more informative trajectories than typical teleoperation systems—yielding data that, within a few seconds, encapsulates the equivalent manipulation richness of much longer conventional demonstrations. Figure 4 (Left) illustrates representative frames from selected collection environments; orange text boxes denote specific tasks, while blue numerals indicate corresponding demonstration counts. Figure 4 (Right) provides two distribution plots: the upper plot details task-level proportions, whereas the lower plot illustrates the breakdown of manipulation skills (e.g., pick, open, etc.). Details on data acquisition and storage format are provided in Appendix 8.12 and 8.13, respectively.

6 System Evaluation

We assess our system across four primary dimensions: **1) Baseline Performance**, **2) Algorithmic Enhancements**, **3) Data Quality**, and **4) Further Analyses**. We present the results for Baseline Performance and Algorithmic Enhancements in Sections 6.1 and 6.2, while the evaluations of Data Quality and Further Analyses are provided in Appendix 8.14 and 8.15, respectively.

We begin by demonstrating the system's effectiveness across a diverse set of manipulation tasks, establishing baseline performance through comprehensive experiments with ACT and DP methodologies. We then quantify the performance gains achieved via our algorithmic enhancements, which address challenges such as trajectory smoothness, perceptual robustness, and cross-platform generalization. To validate data quality, we conduct a reliability assessment of the T265 and MINI tracking modules, ensuring that the collected datasets meet accuracy and consistency requirements for downstream applications. Finally, we present further analyses that evaluate additional factors—including the impact of fisheye camera utilization, training dataset scale, and a quantitative performance compared with the original UMI—thereby providing a holistic benchmark of system improvements.

Prior to presenting the experimental results, we introduce the **12 tasks** employed for policy inference evaluation, as illustrated in Figure 5. These tasks are designed to encompass a broad spectrum of real-world manipulation challenges, including hinged operations and pick-and-place activities,

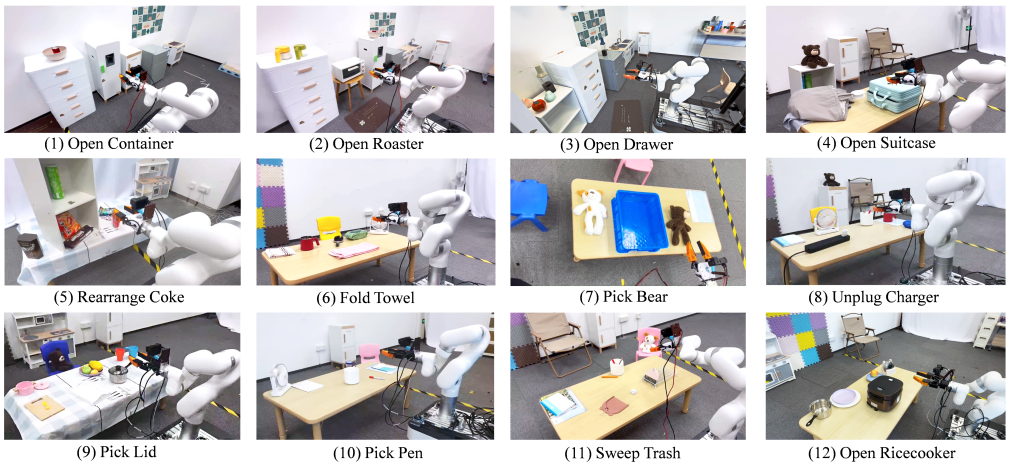

Figure 5: Twelve tasks used for policy inference evaluation, covering a broad range of real-world manipulation challenges. These include **hinged operations** (Tasks 1–4), **pick-and-place activities** (Tasks 5–10), **pick-push manipulation** (Task 11), and **button press actions** (Task 12), providing a comprehensive benchmark for the proposed system.

thereby providing a comprehensive benchmark for assessing the proposed system. Unless otherwise specified, all experiments are conducted using an xArm 6 robotic platform.

6.1 Baseline Performance

Table 1: Success rates for DP and ACT in different tasks, sorted by manipulation type.

Index	Task	Manipulation Type	Success Rate (%) of DP (Relative TCP)	Success Rate (%) of ACT (Absolute Joint)
1	Open Container	Hinged	93.33	86.67
2	Open Roaster	Hinged	80.00	86.67
3	Open Drawer	Hinged	53.33	80.00
4	Open Suitcase	Hinged	40.00	86.67
5	Rearrange Coke	Pick-Place	80.00	86.67
6	Fold Towel	Pick-Place	93.33	73.33
7	Pick Bear	Pick-Place	80.00	20.00
8	Unplug Charger	Pick-Place	86.67	86.67
9	Pick Lid	Pick-Place	53.33	93.33
10	Pick Pen	Pick-Place	53.33	20.00
11	Sweep Trash	Pick-Push	46.67	6.67
12	Open Ricecooker	Button Press	20.00	80.00

We conduct a comparative study of two baseline approaches for policy inference—ACT with absolute joint-space outputs and DP with relative TCP-based outputs—across 12 diverse manipulation tasks. Each task is trained on **200 demonstrations**, randomly selected from our open-source dataset. Among the 200 pieces of data, every 50 pieces are a group. The robotic arm's initial configuration and environment in one group are fixed, while the target object's position varies within a predefined range. Across groups, both the arm's starting pose and the scene arrangement are altered. Each algorithm is trained on the same training data. During testing, we use object poses that appear in the training set but **place them in new scene contexts**. Each task is attempted **15 times**, and success rates are recorded.

The results are shown in Table 1. Both ACT and DP achieve relatively high success rates on most tasks, indicating that the collected dataset is sufficiently diverse and general to support different policy representations. Notably, tasks involving substantial occlusion (e.g., "Rearrange Coke" and "Open Container") can still be effectively handled, suggesting that our data collection strategy is robust to partial visibility.

In tasks requiring precise depth estimation —such as "Open Drawer," "Pick Lid," and "Open Ricecooker"—the baseline **DP** algorithm's limitations become evident. In particular, DP struggled with pressing actions in "Open Ricecooker," where small deviations in relative motion can prevent successful button presses. By contrast, in the baseline **ACT** algorithm, tasks like "Open Suitcase," exhibit more accurate depth reasoning but less sensitivity to specific trajectory requirements. In "Pick Bear," ACT occasionally generates joint configurations unseen during training (e.g., producing a fully inverted gripper posture when the dataset predominantly showed a downward TCP orientation), which highlights a known limitation of the original ACT approach, *typically reliant on third-person viewpoints for global state estimation.* Similar issues arise in "Pick Pen" and "Sweep Trash," with the latter also revealing a workspace mismatch: ACT's absolute joint predictions sometimes yield unreachable targets if training data contained trajectories that exceeded the xArm's operational envelope. By contrast, DP's incremental relative-position strategy partly alleviated this problem, although multi-step tasks like "Sweep Trash" remain challenging for both models.

We also investigate FastUMI's compatibility with multiple robotic platforms: Flexiv Rizon, Z1 and Robotiq. The results are shown in Appendix 8.15.

6.2 Algorithmic Enhancements

We evaluate our algorithmic refinements on two tasks—"Pick Lid" and "Open Ricecooker"—where the baseline DP approach most struggles with depth estimation. We retain the same training data and parameters as in the previous experiment for both ACT and DP and employ the same testing protocol.

In the **Depth-Enhanced DP**, we incorporate depth information into the original DP algorithm. As shown in Table 2, success rates increase by 26.67% on "Pick Lid" and 73.33% on "Open Ricecooker." These results highlight the importance of depth cues for precise object manipulation tasks, particularly those requiring accurate vertical alignment or force application. For ACT, we introduce two variants called **Smooth-ACT** and **PoseACT**, which incorporate GRU-based temporal modeling and integrates TCP (end-effector state) inputs. We demonstrate that these two variants yield substantial improvements in success rate compared to the original ACT, as shown in Table 3. Furthermore, we explore relative TCP to reduce dependence on absolute coordinates, aiming to capture the shape and dynamics of the trajectory more robustly. As the table indicates, this enhancement performs well on tasks with extended or repetitive trajectories (e.g., "Sweep Trash"). However, for tasks requiring precise height estimation (e.g., "Pick Bear"), removing absolute pose information can degrade vertical positioning accuracy, underscoring a trade-off between relative and absolute coordinate representations.

Table 2: Comparison of success rates for DP and DP + Depth in tasks with significant depth-estimation challenges.

Task	Success Rate (%) of Original DP	Success Rate (%) of Depth-Enhanced DP
Pick Lid	53.33%	80.00%
Open Ricecooker	20.00%	93.33%

Table 3: Comparison of success rates for different ACT variants across representative tasks.

Task	Joint		TCP	
	ACT	Smooth-ACT	PoseACT (Absolute)	PoseACT (Relative)
Pick Bear	20.00%	60.00%	80.00%	73.33%
Sweep Trash	6.67%	26.67%	53.33%	60.00%

7 Conclusion

In this work, we introduce *FastUMI*, a redesigned system built on the original UMI to streamline real-world data collection for robotic manipulation. Our hardware modifications enable quick deployment across diverse arms and grippers, removing dependencies on specialized components. By replacing complex SLAM with a modular tracking system (e.g., T265 and MINI), FastUMI reduces calibration overhead and maintains robust performance despite occlusions. We also open-source a dataset of 15,000 real-world demonstrations spanning 24 daily tasks. Experiments confirm that FastUMI lowers costs, simplifies deployment, and supports large-scale data-driven policy learning. Future work will focus on integrating richer sensing modalities and extending FastUMI to more complex platforms.

Limitations. While FastUMI demonstrates effective policy execution across diverse tasks, several limitations remain: **1) Limited Sensing Modalities**. FastUMI currently relies on visual data, which may prove insufficient for tasks requiring precise force or tactile feedback—such as handling fragile objects. Integrating tactile or force sensors could enable richer environmental representations and more robust policy learning, particularly for tasks necessitating delicate or high-precision interactions. **2) Restricted Robot Compatibility**. Although FastUMI accommodates single-arm or dual-arm platforms, it is not yet adapted for more complex morphologies, including mobile manipulators requiring whole-body control. Future endeavors could focus on expanding the hardware and software ecosystem to support advanced platforms with larger workspaces and non-static bases.

Acknowledgments

This work is supported by the Shanghai AI Laboratory, the National Natural Science Foundation of China (62376222), and Young Elite Scientists Sponsorship Program by CAST (2023QNRC001).

References

[1] S. Bahl, A. Gupta, and D. Pathak. Human-to-robot imitation in the wild, 2022. URL https://arxiv.org/abs/2207.09450.

[2] C. Chi, Z. Xu, C. Pan, E. Cousineau, B. Burchfiel, S. Feng, R. Tedrake, and S. Song. Universal manipulation interface: In-the-wild robot teaching without in-the-wild robots. *arXiv preprint arXiv:2402.10329*, 2024.

[3] Zhaxizhuoma, P. Chen, Z. Wu, J. Sun, D. Wang, P. Zhou, N. Cao, Y. Ding, B. Zhao, and X. Li. Alignbot: Aligning vlm-powered customized task planning with user reminders through fine-tuning for household robots, 2024. URL https://arxiv.org/abs/2409.11905.

[4] 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.

[5] Y. Zhu, A. Joshi, P. Stone, and Y. Zhu. Viola: Imitation learning for vision-based manipulation with object proposal priors. In *Conference on Robot Learning*, pages 1199–1210. PMLR, 2023.

[6] W. Fan, X. Guo, E. Feng, J. Lin, Y. Wang, J. Liang, M. Garrad, J. Rossiter, Z. Zhang, N. Lepora, L. Wei, and D. Zhang. Digital twin-driven mixed reality framework for immersive teleoperation with haptic rendering. *IEEE Robotics and Automation Letters*, 8(12):8494–8501, 2023. doi: 10.1109/LRA.2023.3325784.

[7] D. Zhang, Z. Wu, J. Zheng, Y. Li, Z. Dong, and J. Lin. Hubotverse: Toward internet of human and intelligent robotic things with a digital twin-based mixed reality framework. *IEEE Robotics Automation Magazine*, pages 2–12, 2024. doi:10.1109/MRA.2024.3417090.

[8] P. Wu, Y. Shentu, Z. Yi, X. Lin, and P. Abbeel. Gello: A general, low-cost, and intuitive teleoperation framework for robot manipulators. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 12156–12163. IEEE, 2024.

[9] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection, 2016. URL https://arxiv.org/abs/1603.02199.

[10] H. Song, D. Qu, Y. Yao, Q. Chen, Q. Lv, Y. Tang, M. Shi, G. Ren, M. Yao, B. Zhao, D. Wang, and X. Li. Hume: Introducing system-2 thinking in visual-language-action model, 2025. URL https://arxiv.org/abs/2505.21432.

[11] J. Hou, T. Wang, T. Pan, S. Wang, X. Xue, and Y. Fu. Tamma: Target-driven multi-subscene mobile manipulation. In *8th Annual Conference on Robot Learning*, 2024.

[12] J. Zhang, Y. Gu, J. Gao, H. Lin, Q. Sun, X. Sun, X. Xue, and Y. Fu. Lac-net: Linear-fusion attention-guided convolutional network for accurate robotic grasping under the occlusion. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10059–10065. IEEE, 2024.

[13] S. Cabi, S. G. Colmenarejo, A. Novikov, K. Konyushkova, S. Reed, R. Jeong, K. Zolna, Y. Aytar, D. Budden, M. Vecerik, O. Sushkov, D. Barker, J. Scholz, M. Denil, N. de Freitas, and Z. Wang. Scaling data-driven robotics with reward sketching and batch reinforcement learning, 2020. URL https://arxiv.org/abs/1909.12200.

[14] C. Wang, L. Fan, J. Sun, R. Zhang, L. Fei-Fei, D. Xu, Y. Zhu, and A. Anandkumar. Mimicplay: Long-horizon imitation learning by watching human play, 2023. URL https://arxiv.org/abs/2302.12422.

[15] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation, 2018. URL https://arxiv.org/abs/1710.04615.

[16] 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.

[17] L. X. Shi, Z. Hu, T. Z. Zhao, A. Sharma, K. Pertsch, J. Luo, S. Levine, and C. Finn. Yell at your robot: Improving on-the-fly from language corrections. *arXiv preprint arXiv:2403.12910*, 2024.

[18] Z. Fu, T. Z. Zhao, and C. Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *arXiv preprint arXiv:2401.02117*, 2024.

[19] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021.

[20] 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.

[21] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014. ISSN 0031-3203. doi:https://doi.org/10.1016/j.patcog.2014.01.005. URL https://www.sciencedirect.com/science/article/pii/S0031320314000235.

[22] R. Dey and F. M. Salem. Gate-variants of gated recurrent unit (gru) neural networks. In *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, pages 1597–1600. IEEE, 2017.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

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

[25] T.-W. Ke, N. Gkanatsios, and K. Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. *arXiv preprint arXiv:2402.10885*, 2024.

[26] L. Yang, B. Kang, Z. Huang, Z. Zhao, X. Xu, J. Feng, and H. Zhao. Depth anything v2, 2024. URL https://arxiv.org/abs/2406.09414.

[27] M. Grupp. evo: Python package for the evaluation of odometry and slam. https://github.com/MichaelGrupp/evo, 2017.

[28] S. Belkhale, Y. Cui, and D. Sadigh. Data Quality in Imitation Learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 80375–80395. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/fe692980c5d9732cf153ce27947653a7-Paper-Conference.pdf.

[29] Y. Park, J. S. Bhatia, L. Ankile, and P. Agrawal. DexHub and DART: Towards Internet Scale Robot Data Collection, Nov. 2024. URL http://arxiv.org/abs/2411.02214. arXiv:2411.02214 [cs].

[30] S. Chen, C. Wang, K. Nguyen, L. Fei-Fei, and C. K. Liu. Arcap: Collecting high-quality human demonstrations for robot learning with augmented reality feedback. *arXiv preprint arXiv:2410.08464*, 2024.

[31] P. Owan, J. Garbini, and S. Devasia. Faster Confined Space Manufacturing Teleoperation Through Dynamic Autonomy With Task Dynamics Imitation Learning. *IEEE Robotics and Automation Letters*, 5(2):2357–2364, Apr. 2020. ISSN 2377-3766. doi: 10.1109/LRA.2020.2970653. URL https://ieeexplore.ieee.org/abstract/document/8976114. Conference Name: IEEE Robotics and Automation Letters.

[32] R. Ding, Y. Qin, J. Zhu, C. Jia, S. Yang, R. Yang, X. Qi, and X. Wang. Bunny-VisionPro: Real-Time Bimanual Dexterous Teleoperation for Imitation Learning, July 2024. URL http://arxiv.org/abs/2407.03162. arXiv:2407.03162 [cs].

[33] H. Liu, S. Nasiriany, L. Zhang, Z. Bao, and Y. Zhu. Robot learning on the job: Human-in-the-loop autonomy and learning during deployment. *The International Journal of Robotics Research*, page 02783649241273901, Oct. 2024. ISSN 0278-3649. doi:10.1177/02783649241273901. URL https://doi.org/10.1177/02783649241273901. Publisher: SAGE Publications Ltd STM.

[34] O. Mees, M. Merklinger, G. Kalweit, and W. Burgard. Adversarial Skill Networks: Unsupervised Robot Skill Learning from Video. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4188–4194, May 2020. doi:10.1109/ICRA40945.2020.9196582. URL https://ieeexplore.ieee.org/abstract/document/9196582. ISSN: 2577-087X.

[35] S. Sontakke, J. Zhang, S. Arnold, K. Pertsch, E. Bıyık, D. Sadigh, C. Finn, and L. Itti. RoboCLIP: One Demonstration is Enough to Learn Robot Policies. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 55681–55693. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/ae54ce310476218f26dd48c1626d5187-Paper-Conference.pdf.

[36] T. Wang, Y. Li, H. Lin, X. Xue, and Y. Fu. Wall-e: Embodied robotic waiter load lifting with large language model. *arXiv preprint arXiv:2308.15962*, 2023.

[37] Y. Yang, Y. Li, C. Fermuller, and Y. Aloimonos. Robot Learning Manipulation Action Plans by "Watching" Unconstrained Videos from the World Wide Web. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1), Mar. 2015. ISSN 2374-3468. doi:10.1609/aaai.v29i1.9671. URL https://ojs.aaai.org/index.php/AAAI/article/view/9671. Number: 1.

[38] S. Wang, P. Chen, J. Zhou, Q. Li, J. Dong, J. Gao, B. Xue, J. Jiang, L. Kong, and C. Wu. Treesynth: Synthesizing diverse data from scratch via tree-guided subspace partitioning, 2025. URL https://arxiv.org/abs/2503.17195.

[39] C. Eze and C. Crick. Learning by Watching: A Review of Video-based Learning Approaches for Robot Manipulation, Sept. 2024. URL http://arxiv.org/abs/2402.07127. arXiv:2402.07127 [cs].

[40] S. Song, A. Zeng, J. Lee, and T. Funkhouser. Grasping in the Wild: Learning 6DoF Closed-Loop Grasping From Low-Cost Demonstrations. *IEEE Robotics and Automation Letters*, 5(3): 4978–4985, July 2020. ISSN 2377-3766. doi:10.1109/LRA.2020.3004787. URL https://ieeexplore.ieee.org/abstract/document/9126187. Conference Name: IEEE Robotics and Automation Letters.

[41] C. Aeronautiques, A. Howe, C. Knoblock, I. D. McDermott, A. Ram, M. Veloso, D. Weld, D. W. Sri, A. Barrett, D. Christianson, et al. Pddl— the planning domain definition language. *Technical Report, Tech. Rep.*, 1998.

[42] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, 2017.

[43] C. Daniel, M. Viering, J. Metz, O. Kroemer, and J. Peters. Active reward learning. In *Robotics: Science and systems*, volume 98, 2014.

[44] S. Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3 (6):233–242, 1999.

[45] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

[46] B. Fang, S. Jia, D. Guo, M. Xu, S. Wen, and F. Sun. Survey of imitation learning for robotic manipulation. *International Journal of Intelligent Robotics and Applications*, 3:362–369, 2019.

[47] S. Stepputtis, J. Campbell, M. Phielipp, S. Lee, C. Baral, and H. Ben Amor. Language-conditioned imitation learning for robot manipulation tasks. *Advances in Neural Information Processing Systems*, 33:13139–13150, 2020.

[48] M. Zare, P. M. Kebria, A. Khosravi, and S. Nahavandi. A survey of imitation learning: Algorithms, recent developments, and challenges. *IEEE Transactions on Cybernetics*, 2024.

[49] H. R. Walke, K. Black, T. Z. Zhao, Q. Vuong, C. Zheng, P. Hansen-Estruch, A. W. He, V. Myers, M. J. Kim, M. Du, et al. Bridgedata v2: A dataset for robot learning at scale. In *Conference on Robot Learning*, pages 1723–1736. PMLR, 2023.

[50] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[51] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[52] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Conference on robot learning*, pages 651–673. PMLR, 2018.

[53] H.-S. Fang, H. Fang, Z. Tang, J. Liu, C. Wang, J. Wang, H. Zhu, and C. Lu. Rh20t: A comprehensive robotic dataset for learning diverse skills in one-shot. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 653–660. IEEE, 2024.

[54] K. Doshi, Y. Huang, and S. Coros. On hand-held grippers and the morphological gap in human manipulation demonstration. *arXiv preprint arXiv:2311.01832*, 2023.

[55] F. Sanches, G. Gao, N. Elangovan, R. V. Godoy, J. Chapman, K. Wang, P. Jarvis, and M. Liarokapis. Scalable. intuitive human to robot skill transfer with wearable human machine interfaces: On complex, dexterous tasks. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6318–6325. IEEE, 2023.

[56] C. Wang, H. Shi, W. Wang, R. Zhang, L. Fei-Fei, and C. K. Liu. Dexcap: Scalable and portable mocap data collection system for dexterous manipulation. *arXiv preprint arXiv:2403.07788*, 2024.

8 Appendix

8.1 Component Details

Figure 6 presents an expanded view of the FastUMI hardware, with each component labeled and annotated to clarify its specific role within the system.

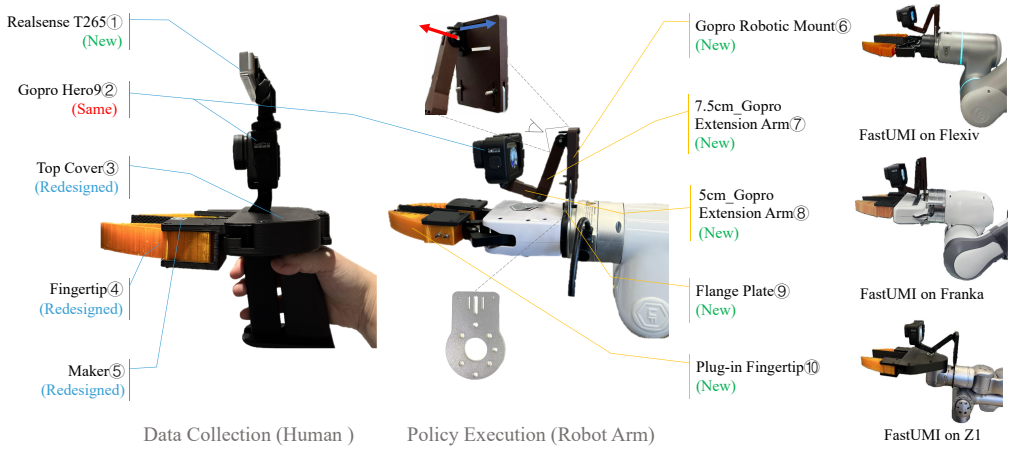

Figure 6: Physical prototypes of FastUMI. **Left**: The handheld device, *used to collect demonstration data from human operators*, includes a GoPro② for visual feedback, a RealSense T265① for end-effector pose tracking, fingertip markers④⑤ to measure the gripper aperture, and a top cover③ to secure both the GoPro and T265. **Middle**: A robot-mounted device, *used for executing learned policies on the robotic arm*, mirrors the handheld configuration. It features an ISO-standard-compatible camera mounting solution (including gopro mount⑥, extension arms⑦⑧, and flange plate⑨) that adapts to varying arm and gripper geometries. This design maintains consistent GoPro perspectives across different setups, enabling direct transfer of human demonstration views to autonomous robotic executions. **Right**: FastUMI can be easily deployed on various robotic arms and grippers.

8.2 Handheld Device Design

The handheld device (see the left subfigure in Figure 6) enables manual data collection for training action policies. It comprises three primary components:

- **Fisheye Camera Module②**: A GoPro camera with a fisheye extension captures wide-angle images with a 169-degree field of view (FOV), significantly reducing occlusions and providing a broad perspective for robotic tasks. This FOV is substantially larger than that of commonly used cameras such as the RealSense D435i, whose narrower field of view has proven suboptimal for first-person view (FPV) data collection in our tests. In contrast, the wider coverage of a fisheye camera effectively captures more environmental context and enhances visual feature extraction, thereby improving policy learning outcomes.

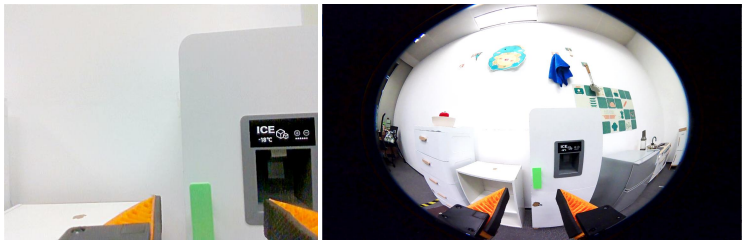

Figure 7: **Left:** D435i camera with a narrower field of view. **Right:** GoPro with a 155-degree wide-angle view.

- **Pose Tracking Module**①: The handheld interface lacks intrinsic joint feedback or an external motion-capture system, so we incorporate a RealSense T265 for robust tracking. The T265, featuring a high-performance integrated IMU, replaces UMI's visual odometry solution, eliminating complex calibration steps and enhancing usability. *In our experiments, this module consistently delivers stable pose estimation across a wide range of scenarios, including those involving partial visual occlusions (e.g., opening cabinets and drawers).* This improved hardware design accommodates a broader set of conditions, which is crucial for data collection. Moreover, our system is *robust to different camera models*, allowing smooth integration of alternative sensors without major modifications to the existing pipeline. For instance, we have verified that the RoboBaton MINI provides performance comparable to the T265 while maintaining a fully compatible data interface, ensuring high-quality data collection and continued availability. A comparison of the T265 and MINI is presented in Section 8.16.
- **Top Cover, Fingertip, and Marker**③④⑤: In the original UMI design, the top cover often appears in the GoPro's field of view, *preventing complete hardware decoupling*. To address this limitation, we reposition the GoPro closer to the fingertips and ensure the top cover remains outside the fisheye lens range, thereby accommodating setups where the cover may be absent (e.g., when mounted on a robot). Although moving the camera closer to the fingertips naturally introduces greater image distortion, we tackle this challenge by optimizing both the size and placement of the markers. These refinements minimize lens distortion effects, improve marker detection accuracy, and enhance durability and ease of attachment. This redesign increases system flexibility and reliability.

In this configuration, the camera is factory-calibrated and aligned with the fingertips, requiring no further user adjustment and enabling a straightforward *plug-and-play* experience. We employ two camera modules in FastUMI, each serving a distinct function: the T265 provides accurate pose tracking even under partial occlusions, while the GoPro delivers an expansive view crucial for environmental context capture, demonstration verification, and learning algorithm support. Because the GoPro is not responsible for pose tracking, it can be mounted more flexibly to maintain consistent viewpoints across both handheld and robot-mounted devices, whereas the T265 is placed in a more protected location to ensure stable pose tracking performance.

8.3 Robot-Mounted Device Design

The robot-mounted device (see the middle subfigure in Figure 6) follows the same design principles as its handheld counterpart but is **engineered for broad compatibility with a wide range of robotic arms and grippers.** Unlike the handheld configuration, the robot-mounted device does *not* include a T265 camera. Its main components include:

- **Flange Plate**⑨: Designed in compliance with ISO standards, the flange plate is compatible with a wide range of robotic arms, ensuring seamless integration and significantly reducing setup time.
- **Plug-in Fingertip**⑩: Outwardly identical to the attachments used in the handheld device, these modules are *internally contoured to accommodate varying gripper shapes while preserving uniform external interaction points*, thereby facilitating effective policy transfer. Interchangeable fingertip modules establish standardized physical interaction points, supporting compatibility with various robotic and handheld grippers. To accommodate a wide range of robotic grippers, we design five customized fingertip attachments (*e.g.*, the xArm gripper and Robotiq 2f-85) based on commonly used grippers in open-source datasets such as Open X-Embodiment [4], covering over 90% of the grippers in these datasets. Although not all gripper types are yet supported, our design can be readily adapted as needed. Figure 9 illustrates our fingertip design integrated with the xArm Gripper.
- **Adjustable Camera Mounting Structure**⑥⑦⑧: Modular extension arms facilitate precise alignment of the GoPro with the gripper's fingertips, ensuring consistent viewpoints across different robot setups. This structure comprises two key parts: 1) *GoPro Robotic Mount*⑥ serves as the primary attachment point for the GoPro. 2) *GoPro Extension Arm*⑦⑧ enable both lateral positioning (indicated by the blue arrow) and vertical positioning (indicated by the red arrow) to align the

camera with the robot gripper, as demonstrated in Figure 6. Standardized male-female interfaces allow sequential connections of extension arms, providing adjustable length with minimal vibration (tested up to three extensions). By adjusting the extension arm, users can replicate the handheld device's camera perspective, even when grippers vary widely in size or shape. Insertable fingertip extensions further ensure consistent viewpoints across heterogeneous hardware configurations.

Visual Alignment: To ensure visual consistency between the handheld and robot-mounted devices, we adopt a straightforward rule: *"The bottom of the GoPro's fisheye lens image aligns with the bottom of the gripper's fingertips."*, as illustrated in Figure 8. This standard viewpoint ensures that all users capture nearly identical observations, enhancing interoperability across different deployments. Although alternative standards could be defined, deviating from this alignment would reduce the utility of shared datasets for broader applications. If gripper sizes vary, our adjustable mechanical design accommodates fine-grained arm adjustments to maintain visual alignment. In practice, the handheld device employs a fixed camera configuration, whereas the robot-mounted device requires an adjustable setup due to variations in arm geometries and end-effector designs.

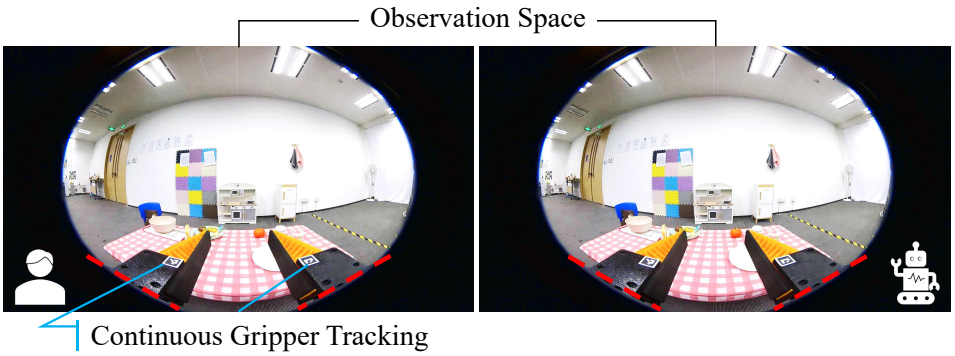

Figure 8: Visual alignment between the handheld device (**Left**) and the robot-mounted device (**Right**). The two views demonstrate the consistent positioning of the GoPro's fisheye lens image, with the bottom of the gripper's fingertips aligned to the red dashed lines.

Although our handheld device is a *parallel-motion gripper*, many robot-mounted grippers, such as the xArm Gripper or Robotiq Gripper, do not strictly maintain parallel motion. For example, the xArm Gripper's effective length changes by approximately 1 centimeter as it moves between fully open and closed positions (see Figure 9). This discrepancy in gripper motion can create mismatches in the observed camera view, especially when transferring demonstrations collected on the handheld device to different robot-mounted setups. To resolve this challenge, we develop a *dynamic error-compensation algorithm* that compensates for gripper-specific motion differences during inference, thereby preserving consistent visual alignment between human demonstrations and robotic executions (see Section 4.4 for details).

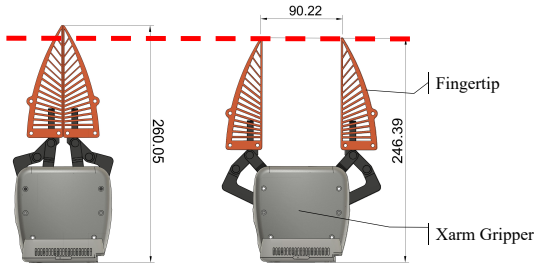

Figure 9: Our plug-in fingertip design integrated with the xArm Gripper; The effective length of the xArm Gripper changes by approximately 1 centimeter between fully closed and open positions, potentially causing misalignment when transferring demonstrations.

8.4 Other Design Optimizations

To improve the stability and durability of FastUMI, we introduce three structural enhancements:

- *Reinforced Key Mount Structure:* Increased structural integrity to reduce vibration.
- *Carbon Fiber Components:* Strengthened material properties while minimizing weight.
- *Standardized Male-Female Interface Design*: Allowed sequential connection of extension arms to adjust length without significant vibration.

Overall, the extensive hardware-related designs ensure reliable performance during data collection and simplify hardware adjustments for users. Additionally, our system configuration allows for a single standardized handheld device to be shared among multiple users, while the robot-mounted device can be adapted to various grippers or robotic arms. This decoupled arrangement preserves uniform data collection workflows and advances our goal of making FastUMI accessible to a broader user community.

8.5 Data Sub-Sampling and Synchronization

In multisensor data fusion, differing sampling rates and data patterns often hinder precise alignment. To address these challenges, we employ a *unified ROS clock* for consistent timestamping, a *multi-threaded buffering mechanism* to handle each sensor stream independently, and *synchronized sub-sampling* at the greatest common frequency. This integrated strategy ensures robust multi-modal alignment without compromising data integrity. Such measures are necessary because certain sensors (e.g., the T265 at 200 Hz and the GoPro at 60 Hz) exhibit mismatched rates, increasing the risk of misalignment and overload. In practice, each sensor's data is tagged with the unified clock and routed into a dedicated thread-safe queue to prevent data loss under high throughput. Before each recording session, these queues are reset to maintain orderly buffers. We then sub-sample both streams at 20 Hz—identified as the greatest common frequency between 200 Hz and 60 Hz—by retaining one in every three camera frames and pairing each retained frame with the temporally nearest T265 pose. This approach achieves *sub-millisecond offsets*, well within half the T265's 1/200 s interval, minimizing interpolation errors and ensuring consistently synchronized data for downstream learning tasks.

8.6 Accumulated Drift Correction for T265

Although the T265 provides robust pose estimates, it can accumulate drift during substantial motion (e.g., sudden accelerations). To address this, we employ two main strategies: *1) Reinitialization.* The simplest remedy is to restart the T265 in a stationary, predefined reference pose. This action resets the device's internal state and restores accurate pose tracking. *2) Loop Closure.* Another strategy leverages a visually distinct reference region—a blue 3D-printed groove on the table (Figure 10 Left)—to facilitate loop closure. When the T265 revisits this area, it re-encounters previously mapped visual features, typically realigning the estimated trajectory with the initial reference (highlighted as a green dashed box in Figure 10 Right) in RVIZ under minor drift. However, if significant misalignment persists even after returning to the marked area, loop closure is deemed ineffective, and the T265 must be reinitialized to restore accurate pose tracking.

8.7 Raw Data Quality Assessment

Ensuring reliable demonstrations is crucial for downstream learning tasks; however, to our knowledge, no existing work fully quantifies what constitutes "ideal" data quality. In practice, we enforce consistency through *sensor confidence* and *trajectory smoothness* checks. The T265 provides four discrete confidence levels—Failed, Low, Medium, and High; to avoid prolonged low-confidence data, we first validate the environment by confirming that at least 95% of sample poses achieve *High* confidence. Our tests indicate that **lighting conditions** notably affect the T265's performance, with dim or low-light environments often leading to reduced confidence levels and increased drift. During actual recordings, any low-confidence pose is excluded and interpolated from neighboring frames to maintain continuity. Meanwhile, user-defined thresholds on *velocity*, *acceleration*, and *relative orientation* identify abrupt transitions, further refining data fidelity. Although these strategies cannot

Figure 10: **Left**: The blue 3D-printed groove on the table, serving as a clear visual reference to aid loop closure. **Right**: The T265's trajectory in RVIZ, illustrating alignment with the initial reference, highlighted as a green dashed box, after revisiting the blue groove.

guarantee an absolute benchmark for data quality, they help establish rigorous collection standards and minimize errors that could propagate into subsequent modeling and inference.

8.8 Trajectory Computation

Absolute TCP Trajectory: To calculate the absolute TCP trajectory, we first computed the pose of the T265 camera in the robot base frame, and then used this pose to calculate the TCP pose in the robot base frame. Specifically, at each timestamp i, the T265 provides $\left(\mathbf{p}_i, \mathbf{R}_i\right)$, describing the camera's motion relative to its initial pose. The camera's absolute pose in the robot base frame is given by:

$$\mathbf{p}_{\text{cam}}^{(i)} = \mathbf{p}_{\text{b2g}} + \mathbf{p}_i - \mathbf{R}_{\text{b2g}} \Delta_{c2g}, \tag{2}$$

$$\mathbf{R}_{\text{cam}}^{(i)} = \mathbf{R}_{\text{b2g}} \cdot \mathbf{R}_i. \tag{3}$$

The absolute TCP pose $\left(\mathbf{p}_{\text{ee}}^{(i)}, \mathbf{R}_{\text{ee}}^{(i)}\right)$ is then obtained by incorporating the camera-to-gripper offset Δ_{c2g}:

$$\mathbf{p}_{\text{ee}}^{(i)} = \mathbf{p}_{\text{cam}}^{(i)} + \mathbf{R}_{\text{cam}}^{(i)} \Delta_{c2g}, \tag{4}$$

$$\mathbf{R}_{\text{ee}}^{(i)} = \mathbf{R}_{\text{cam}}^{(i)}. \tag{5}$$

The resulting sequence $\left\{\left(\mathbf{p}_{\text{ee}}^{(i)}, \mathbf{R}_{\text{ee}}^{(i)}\right)\right\}$ yields the absolute TCP trajectory in the robot base frame.

Relative TCP Trajectory: This trajectory is formed from consecutive absolute TCP frames. For adjacent frames i and $i+1$, with absolute poses $\left(\mathbf{p}_{\text{ee}}^{(i)}, \mathbf{R}_{\text{ee}}^{(i)}\right)$ and $\left(\mathbf{p}_{\text{ee}}^{(i+1)}, \mathbf{R}_{\text{ee}}^{(i+1)}\right)$, the relative transforms are:

$$\mathbf{p}_{\text{rel}}^{(i)} = \mathbf{p}_{\text{ee}}^{(i+1)} - \mathbf{p}_{\text{ee}}^{(i)}, \tag{6}$$

$$\mathbf{R}_{\text{rel}}^{(i)} = \left(\mathbf{R}_{\text{ee}}^{(i)}\right)^{-1} \cdot \mathbf{R}_{\text{ee}}^{(i+1)}. \tag{7}$$

This formulation removes dependence on a global reference, facilitating more uniform data distributions and improving generalization when the base pose varies.

Absolute Joint Trajectory: To obtain it, inverse kinematics (IK) is solved for each absolute TCP pose $\left(\mathbf{p}_{\text{ee}}^{(i)}, \mathbf{R}_{\text{ee}}^{(i)}\right)$ using the robot's URDF, typically via an iterative solver. To maintain continuity, the solution at frame i serves as the initial guess for frame $i+1$. If the URDF only extends to the flange, the flange-to-gripper offset is accounted for in the IK computations to ensure accurate joint solutions.

8.9 Details of Dynamic Error-Compensation Algorithm

In our setup, we attach two ArUco markers to the gripper and define two hyperparameters: $d_{\max}$ and $d_{\min}$. These values represent the pixel distances between the markers at the gripper's maximum and minimum openings, respectively. For each image frame, we detect the markers and compute the pixel distance d. If only one marker is identified, we estimate d by mirroring the known marker about the

gripper's central axis; if no markers are detected, an imputed value is inserted to maintain continuity. Consequently, each frame is guaranteed a valid marker distance. Finally, the physical gripper width, denoted as W, is determined by normalizing the measured distance with respect to $d_{\max}$ and $d_{\min}$, then scaling by $G_{\max}$, which denotes the jaws' maximum physical opening:

$$W = \frac{d - d_{\min}}{d_{\max} - d_{\min}} \times G_{\max}. \tag{8}$$

8.10 Implementation of Depth-Enhanced Diffusion Policy

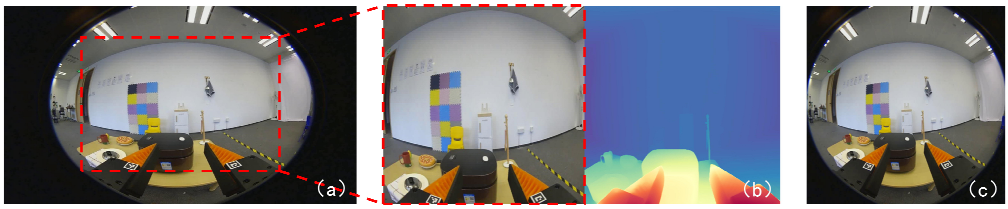

Figure 11: Generation of cropped Depth and RGB Images in FastUMI. (a) is the original image, (b) left is the cropped image for depth map generation, (b) right is the resulting depth map, and (c) is the cropped RGB input.

However, our images contain significant black margins due to the fisheye lens, which negatively impact the performance of Depth Anything V2 [26]. To address this, we preprocess the images by cropping them to retain only the rectangular regions inscribed within the circular area. Specifically, before generating depth maps, we tightly crop the 1920×1080 image to its inscribed rectangle—removing 22% from the left and right, and 17% from the top and bottom—to eliminate high-contrast black margins that could cause estimation errors, as shown in Figure 11(b). Although this cropping reduces the range observable by the fisheye camera, we focus more on the depth information in the areas related to the gripper's contact operations. For the input of RGB images, to keep the field of view as consistent as possible with the depth map while preserving the full fisheye content, we crop 13% from the left and right edges—keeping the top and bottom intact—to produce a rectangular image that fully encompasses the circular region, as shown in Figure 11(c). This dual-cropping approach ensures both high-quality depth maps and alignment with the relevant operational workspace.

During training, we employ cross-attention[23] to fuse features from the metric estimated depth and the RGB image. Specifically, the metric estimated depth serves as the query tokens, while the RGB image provides the key and value tokens. This design allows the depth information to guide attention over the RGB features, enriching them with geometric structure and spatial cues. The fused representations are then concatenated and passed to the downstream policy network for action prediction.

For real-time inference during the diffusion policy rollout, we implement Depth Anything V2 with its large pre-trained model [26], achieving an inference frequency of **20 Hz** on an RTX 4090 GPU. This improvement is achieved without the need for additional sensors or multi-view camera setups, providing a practical and efficient solution to enhance policy performance in precision-critical applications.

8.11 Implementation of Dynamic Error-Compensation Algorithm

Stage 1: Compensation Distance. Let $W(i)$ be the measured gripper width at frame i, and denote the gripper's maximum width by $W_{\max}$. We define a compensation distance $d(i)$ to counteract TCP displacement caused by non-parallel jaws. Let d_{close} be the maximum compensation distance when the gripper is fully closed, and d_{open} be the minimum distance when it is fully open. We then compute

$$d(i) = d_{\text{close}} - \frac{d_{\text{close}} - d_{\text{open}}}{W_{\max}} W(i). \tag{9}$$

As $W(i)$ decreases, the end-effector is shifted further along the negative Z-axis of TCP frame, thereby offsetting the forward motion introduced by closing gripper jaws.

Stage 2: Pose Correction. Let $\mathbf{p}_{\text{ee}}^{(i)}$ and $\mathbf{R}_{\text{ee}}^{(i)}$ be the desired TCP position and orientation at frame i, respectively. The rotation matrix $\mathbf{R}_{\text{ee}}^{(i)}$ defines the TCP coordinate frame's orientation relative to the robot's base frame. To determine the direction of displacement, we first extract the TCP frame's local Z-axis, expressed in the base coordinate frame:

$$\mathbf{z}_{\text{axis}}^{(i)} = \mathbf{R}_{\text{ee}}^{(i)} \hat{\mathbf{e}}_z, \tag{10}$$

where $\hat{\mathbf{e}}_z = [0, 0, 1]^\mathsf{T}$ is the local Z-axis of the TCP coordinate frame. The corrected TCP position $\mathbf{p}_{\text{ee}}^{\prime(i)}$ in the base coordinate frame is then computed as:

$$\mathbf{p}_{\text{ee}}^{\prime(i)} = \mathbf{p}_{\text{ee}}^{(i)} - d(i)\mathbf{z}_{\text{axis}}^{(i)}. \tag{11}$$

Finally, inverse kinematics (IK) is solved using the corrected TCP position $\mathbf{p}_{\text{ee}}^{\prime(i)}$, while maintaining the original orientation $\mathbf{R}_{\text{ee}}^{(i)}$, yielding the joint vector $\theta^{(i)}$:

$$\theta^{(i)} = \text{IK}\big(\mathbf{p}_{\text{ee}}^{\prime(i)}, \mathbf{R}_{\text{ee}}^{(i)}\big). \tag{12}$$

8.12 Dataset Acquisition Process

The dataset is collected by **five** operators using **three** FastUMI devices, ensuring diversity in user interactions and environmental contexts. Each recorded task involves a fixed target object (e.g., a specific drawer or container), while the surrounding background (e.g., table clutter, lighting) is randomized to introduce variability. During acquisition, operators utilize RVIZ for real-time visualization, enabling verification of the T265 sensor output and ensuring high-quality demonstrations. We enforce a quality-assurance protocol by continuously monitoring critical metrics (e.g., T265 tracking confidence) and discarding or re-recording sequences affected by sensor drift.

8.13 Dataset Storage and Format

All raw sensor data are initially recorded locally before undergoing post-processing. To support various imitation learning and control paradigms, we provide multiple data representations—most notably, joint trajectories and TCP trajectories. Each demonstration is stored in a dedicated HDF5 file, encapsulating both observations (e.g., images, tracked poses) and actions (e.g., gripper commands) within a unified dataset. For broader compatibility, we also provide scripts to convert HDF5 files into Zarr format, which maintains a hierarchical structure while offering greater flexibility in storage backends, chunking, compression, and parallel access. Detailed specifications of the dataset schema and file organization are provided in Section 8.17.

8.14 Data Quality

Table 4: Error analysis of trajectories for different tasks (values in mm).

Pose Tracking Module	Task	Traj 1	Traj 2	Traj 3	Traj 4	Traj 5	Traj 6	Traj 7	Traj 8	Traj 9	Traj 10
RealSense T265	Pick Cup	11	10	12	11	11	12	11	10	7	10
	Open Container	19	16	18	17	19	17	17	17	18	19
	Rearrange Coke	36	21	21	20	19	22	21	25	22	26
RoboBaton MINI	Pick Cup	17	15	16	14	13	15	15	14	16	17
	Open Container	10	11	10	11	11	12	11	12	12	12

In Table 4, we summarize the pose estimation errors of both T265 and MINI across three representative tasks: "Pick Cup," a straightforward pick-and-place action (as shown in Figure 12 Right); "Open container," which involves hinged motion and partial occlusion; and "Rearrange Coke," a more complex scenario with substantial occlusion. To establish ground-truth trajectories, four reflective markers are affixed to the handheld device and tracked by an optical motion-capture system (Figure 12 Left). Simultaneously, the device poses are recorded through our data collection pipeline. All data streams are synchronized within ROS via unified timestamps, and ten trajectories per sensor are collected for each task. The *evo* toolkit is used to compute all reported errors [27].

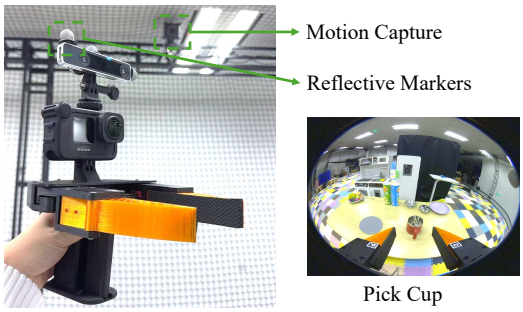

Figure 12: **Left**: Four reflective markers attached to the FastUMI handheld device, tracked by an optical motion-capture system for ground-truth trajectory collection. **Right**: Example scenarios from the "Pick Cup" task.

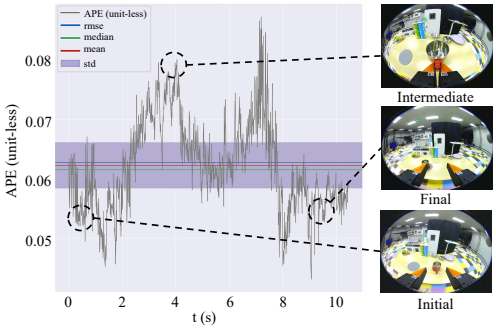

Figure 13: Representative T265 VIO error over time during the "Pick Cup" task. Error peaks appear when the gripper nears the table and occludes visual features, then recover once it returns to the original viewpoint.

In the "Pick Cup" scenario, where occlusion is minimal, T265 achieves an average positioning error of 10.5 mm, while MINI's error averages 15.2 mm. In the "Open Container," T265's error increases to 17.7 mm, reflecting the partial obstruction of its field of view, whereas MINI's error decreases to 11.2 mm. T265's performance degrades further in the "Rearrange Coke," where placing an object inside a cabinet induces significant occlusion. These findings indicate that T265 is particularly susceptible to severe visual obstruction at close range. In contrast, MINI demonstrates relatively stable cross-task performance—albeit with slightly reduced accuracy in low-occlusion scenarios. T265 offers superior localization when visual inputs are largely unobstructed, whereas MINI exhibits more consistent performance under varying levels of occlusion. We also observe that the VIO error typically remains low at the beginning and end of each trajectory but grows noticeably in the middle. Figure 13 illustrates this pattern for T265 during the "Pick Cup" task: as the gripper moves closer to the table, occlusion reduces visible features and causes two pronounced error peaks. During intermediate movement, partial visibility leads to moderate errors, though still higher than at the outset. By the final stage, returning to the original viewpoint restores abundant features, and loop-closure mechanisms recover tracking accuracy to near-initial levels.

Table 5 compares the orientation errors (in degrees) for both T265 and MINI. The Intel T265 achieves a lower mean orientation error (2.64°, max 3.69°) than the RoboBaton MINI (3.18°, max 4.42°) over ten pick-cup trials. Although the MINI slightly outperforms the T265 on Trajectories 3 and 7, the T265 demonstrates greater overall consistency.

Table 5: Orientation Estimation Errors (in degrees)

Sensor	Task	Traj 1	Traj 2	Traj 3	Traj 4	Traj 5	Traj 6	Traj 7	Traj 8	Traj 9	Traj 10
T265	Pick Cup	2.48	2.17	2.78	3.69	2.87	2.80	3.31	2.12	2.14	2.02
RoboBaton MINI	Pick Cup	2.77	3.33	2.47	4.42	4.11	3.34	2.39	2.45	3.32	3.22

8.15 Further Analyses

We further investigate the influence of camera configurations and training data size on policy inference performance. Table 6 compares different camera setups for both pick-and-place and hinged operations. For each configuration (i.e., camera model and lens type), we collected 50 demonstrations under identical scene settings, with only the target object's position randomly varied within a small range. All trajectories were obtained via direct teleoperation. The original ACT algorithm is then evaluated on object positions seen during training but under new trials, each repeated 15 times to compute the success rate. Notably, *a fisheye lens at the end-effector achieves performance comparable to multi-view setups*, potentially because its wide field of view captures richer contextual information for decision-making.

Next, to assess how the amount of training data affects generalization, we conducted an experiment on a "Pick Cup" task with 200, 400, and 800 demonstrations (Table 7). In this scenario, the cup and coaster were each placed in five distinct positions, repeated three times with different handle orientations. The original ACT model must learn not only positional information but also handle orientation to generate an appropriate grasp trajectory. As the dataset grew larger, success rates significantly improved, indicating that data abundance bolsters the model's capacity to generalize across varied object placements and orientations.

Table 6: Comparison of task performance under varying camera setups (lens type and viewpoint).

	D435i (First-Person)	**GoPro with Flat Lens** (First-Person)
Pick Bear	0%	6.67%
Open Container	0%	93.33%
	D435i (First-Person&Third-person)	**GoPro with Fisheye Lens** (First-Person)
Pick Bear	86.67%	80.00%
Open Container	100.00%	100.00%

Table 7: Success rates in the "Pick Cup" task using different training dataset sizes.

Task	**Data Size (200)**	**Data Size (400)**	**Data Size (800)**
Pick Cup	20.00%	26.67%	53.33%

We further provide a quantitative comparison of FastUMI and UMI. FastUMI weighs 732g, a modest 10% increase over UMI's 663g. Despite this, handling remains comparable, with both systems maintaining a data collection rate of 9s per task. In practice, our recent work shows that FastUMI supports prolonged and large-scale use: for instance, 10 operators continuously collected about 3000 long-horizon tasks ($\sim$20s each) over an 8-hour shift without reported fatigue or performance degradation. Table 8 outlines the pipeline of FastUMI, effectively eliminating several time-intensive steps required by UMI. Table 9 compares the SLAM success rates between UMI (using GoPro) and FastUMI (using T265) across several tasks. UMI exhibits substantially lower performance, with many trajectories either not generated or of poor quality with missing key points, whereas FastUMI consistently achieves a 100% success rate.

Table 8: Comparisons of Pipeline Between FastUMI and UMI

Collection Pipeline	**FastUMI**	**UMI**
1. Camera Intrinsic Calibration	0	Over 1200s
2. Prepare Scene	120s	120s
3. Mapping and Gripper Calibration	0	80s
4. SLAM Trajectory Generation	0	639s / 20 Trajectories.

Table 9: Comparisons of SLAM Success Rate (SR)

Task	Open Container	Rearrange Coke	Unplug Charger	Open Drawer
SLAM SR (**FastUMI**)	100%	100%	100%	100%
SLAM SR (**UMI**)	46%	46%	38%	38%

FastUMI has been deployed across a range of robots with diverse gripper configurations, with the results summarized in Table 10.

Table 10: Performance of FastUMI on Different Platforms

Platform	**xArm6 + xArm Gripper**	**Flexiv Rizon + Robotiq 2F-85**	**xArm6 + Robotiq 2F-85**	**Z1 + Custom Gripper**
Algorithm	DP / ACT	DP / ACT	DP / ACT	DP / ACT
Open Container	93.3% / 80.0%	86.7% / 86.7%	93.3% / 86.7%	86.7% / 80.0%
Unplug Charger	86.7% / 80.0%	93.3% / 80.0%	86.7% / 86.7%	73.3% / 66.7%

8.16 RealSense T265 vs. RoboBaton MINI

We present a comparative overview of the key specifications of the T265 and MINI devices. Figure 14 shows the product image of the MINI.

Table 11: Device Specifications Comparison

	T265	**MINI**
Output Frequency (Hz)	200	20
Accuracy (mm)	10	10
FOV	163°(D)	164.7°(D) 164.7°(H) 123.8°(V)
Resolution	848×800	640×480
Weight (g)	55	68
Dimensions (mm)	108×24.5×12.5	101.6×32.25×17.70
SDK	Windows/Linux ROS1	Windows/Linux HTTP/ROS2

Figure 14: The RoboBaton MINI product image.

8.17 FastUMI Dataset

Our dataset is composed of more than 10000 demonstrations from 22 daily tasks. The dataset has been split into smaller parts. Users need to merge the files after downloading to reconstruct the original dataset. Each file is named with its corresponding task name and contains no more than 50 HDF5 files. Each HDF5 file corresponds to a single episode and encapsulates both observational data and actions. Below is the hierarchical structure of the HDF5 file:

```
episode_<idx>.hdf5
├── observations/
│   ├── images/
│   │   └── <camera_name_1> (Dataset)
│   └── qpos (Dataset)
├── action (Dataset)
└── attributes/
    └── sim = False
```

The variable “sim” indicates whether the data was recorded in simulation (True) or real-world (False). The “images” stores image data from cameras as uint8 and has a shape of (num_frames, height=1920, width=1080, channels=3). The “qpos” stores position and orientation data for each timestep and has a shape of (num_timesteps, 7), where the 7 columns correspond to [Pos X, Pos Y, Pos Z, Q_X, Q_Y, Q_Z, Q_W]. The “actions” stores action data corresponding to each timestep. In this script, actions mirror the qpos data.

8.18 Related Work

8.18.1 Data Collection Methods

High-quality data is fundamental to the success of learning algorithms [28]. Here, we introduce several data collection systems and compare them to our Fast-UMI. **Teleoperated systems** represent one of the most widely adopted methodologies for data collection in imitation learning. This approach enables researchers to intuitively gather demonstration data, establishing a direct correspondence between observed visual inputs and associated actions [15]. Various control interfaces, including AR controllers [29, 30], haptic controllers [31, 32], 3D spacemouses [33], and newly explored leader-follower systems [8] are developed to build teleoperated systems. However, these systems inherently depend on real robotic arms during data collection. Additionally, hardware-specific constraints often necessitate modifications to enable cross-platform compatibility, significantly reducing efficiency. In contrast, Fast-UMI requires only a handheld device, enabling portable and flexible data collection.

An alternative data collection paradigm involves capturing multi-view **human demonstration videos**. Robots can extract actionable knowledge from these recordings by leveraging adversarial learning objectives [34], contextualized annotations [35, 36], and hybrid CNN-probabilistic parsing techniques [37, 38]. This approach circumvents the need for physical robotic platforms and facilitates the construction of reusable datasets. However, it presents several inherent limitations. Since the action data are inferred from raw videos, these actions sometimes may not precisely reflect the true actions, which hinders the formation of generalizable policies. Furthermore, the embodiment mismatch remains a persistent challenge, as discrepancies between the domain in which data is collected and the deployment environment can lead to policy failures [39]. In contrast, Fast-UMI directly collects precise action information during demonstrations and minimizes domain shift by aligning video observation of wrist-mounted cameras on both the hand-held device and the on-robot device.

Sensor-enhanced interfaces (i.e., handheld grippers) offer a promising alternative for data collection, addressing some of the aforementioned challenges. However, obtaining precise TCP pose information remains nontrivial. Existing solutions incorporate SLAM-based estimation from video streams [2], motion capture systems [14], and vision-based tracking algorithms [40]. These techniques, however, often necessitate extensive post-processing or rely on fixed infrastructure, reducing overall efficiency. In contrast, Fast-UMI employs the T265 to directly capture accurate pose data, eliminating the need for cumbersome SLAM pipelines or motion capture systems. Additionally, its wrist-mounted gopro camera records high-resolution visual data at variable frame rates, providing a rich observational dataset to support policy learning.

8.18.2 Imitation Learning

Unlike methods that heavily rely on human programming [41] and task-specific reward functions [42, 43], Imitation Learning (IL) enables robots to autonomously perform tasks by learning from expert demonstrations [44, 45, 46, 47, 48]. With the large-scale collection of robotic manipulation datasets in recent years [4, 49, 50, 51, 52, 53], IL has been widely adopted in robotic manipulation, demonstrating remarkable performance across diverse task domains. Depending on the nature of the collected data, IL algorithms can leverage real-robot demonstrations [16, 18, 17], video-based observations without explicit action labels [14, 1], or data obtained from decoupled handheld tracking devices [2, 54, 55]. Furthermore, in dexterous hand manipulation tasks, IL has been extended to learn from human hand motion demonstrations [56, 30]. The ACT algorithm applies imitation learning to absolute joint pose data collected from robotic arms and utilizes temporal ensemble techniques over fixed-length action sequences to enable smooth and autonomous dual-arm control [16]. The work [17] integrates a language modality into the ACT algorithm, fine-tuning a vision-language model to facilitate language-based interaction with dual-arm robots. DP generates actions in the robotic action space through a conditional denoising process, offering advantages such as expressing multimodal action distributions, handling high-dimensional output spaces, and providing stable training [20]. UMI demonstrates that UMI-like data can be effectively used to train diffusion policy, and yield promising results [2]. In our work, we validate our system's performance using data collected through ACT [16] and DP [20]. We further analyze the characteristics of the data collected by Fast-UMI and evaluate its impact on

these algorithms. Based on this analysis, we implement a series of optimizations and adaptations, enhancing the performance of these algorithms when applied to Fast-UMI data.