

EgoEngine: From Egocentric Human Videos to High-Fidelity Dexterous Robot Demonstrations

Yangcen Liu, Shuo Cheng, Xinchun Yin^{1†}, Woo Chul Shin^{1†}, Alfred Cueva^{1†}, Yiran Yang²,
Zhenyang Chen¹, Chuye Zhang¹, Danfei Xu¹

¹Georgia Institute of Technology ²Tsinghua University

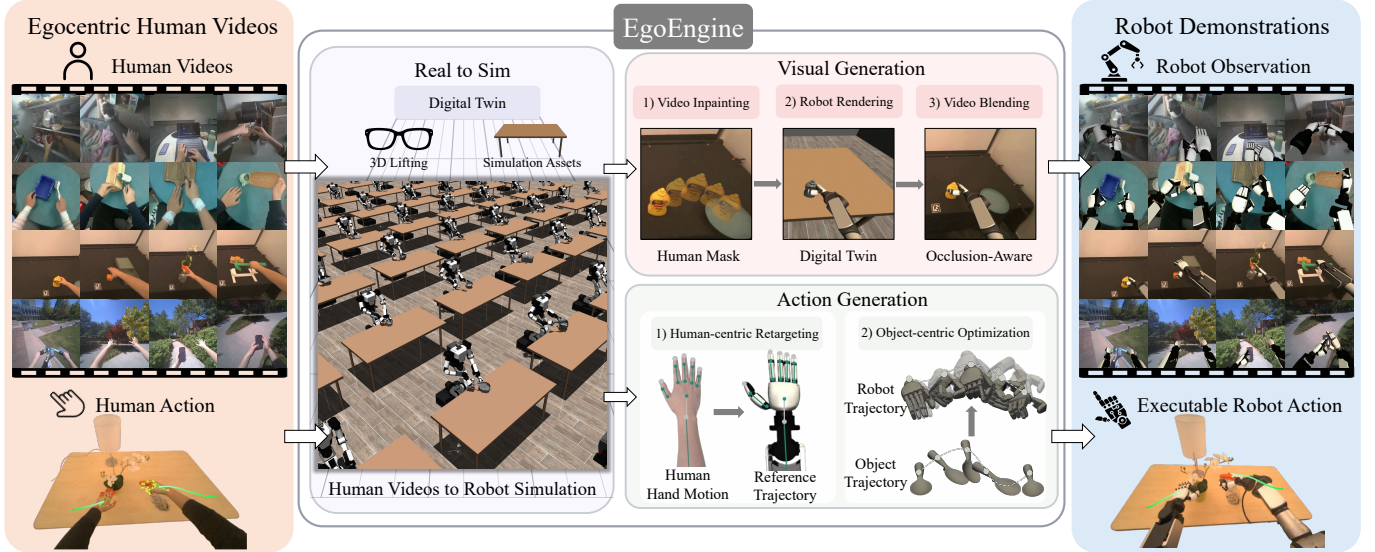


Fig. 1: **EgoEngine** is a scalable data engine that converts egocentric human videos into robot demonstrations. Given an egocentric human video, EgoEngine creates a digital twin and jointly produces (1) a high-fidelity, temporally consistent **robot observation video** and (2) executable **action trajectories** aligned with the video. The generated demonstrations serve as training data that can be distilled into downstream visuomotor policies for **zero-shot** execution.

Abstract—Dexterous manipulation is limited by the cost of collecting large-scale robot demonstrations. Egocentric human videos offer a scalable source of diverse manipulation behaviors, but directly using them for robot learning requires bridging two gaps: the visual gap between human and robot observations, and the action gap between human motion and robot-executable action. We propose EgoEngine, a scalable framework for transforming egocentric human manipulation videos into high-fidelity robot data. Given an egocentric RGB video, EgoEngine produces: (i) a high-fidelity robot observation video replacing human with robot while preserving scene context and temporal alignment, and (ii) a task-aligned, executable robot action trajectory under feasibility constraints. Experiments in simulation and on real robots show that EgoEngine enables scalable conversion of human videos into robot data and, to our knowledge, demonstrates the first zero-shot visuomotor dexterous policy learning from egocentric human videos without real-robot demonstrations. Project website: <https://egoengine.github.io>.

I. INTRODUCTION

Scaling robot learning hinges on access to large volumes of high-quality data that capture diverse and physically grounded interactions between robots and environments [31, 17]. Collecting such data on real robots remains costly and time-

consuming, especially for dexterous manipulation. Existing data collection pipelines commonly rely on teleoperation, which is expensive to scale due to limited hardware availability, complex interfaces, and high-DoF contact-rich control [60, 52, 36]. As a consequence, many dexterous policies are trained on narrow robot demonstration distributions, motivating scalable non-robot data sources that can reduce or eliminate dependence on robot demonstration collection. In contrast, egocentric human videos are abundant and naturally capture contact-rich dexterous manipulation across diverse scenes, objects, and interaction styles [7, 36, 12, 67]. Wearable interfaces such as Aria glasses [5] further make scalable high-fidelity collection efficient for robot learning [16, 42].

However, human videos are not robot demonstrations [24, 35, 68]. The challenge is twofold: visually, human arms and hands occlude the scene and differ substantially from the robot embodiment; on the action side, differences in morphology, kinematics, actuation, and contact dynamics make directly retargeted robot trajectories physically infeasible. Although several works [16, 48] use human videos for pretraining or co-training, they still rely primarily on robot teleoperation

demonstrations for policy learning. The central challenge is therefore to turn human videos into robot demonstrations that preserve the task while providing robot observations and executable robot actions [23, 19].

In this work, we introduce **EgoEngine**, a scalable framework to convert egocentric human videos to high-fidelity robot demonstrations that simultaneously addresses the human-to-robot visual gap and action gap. As shown in Fig. 1, given a human RGB video, EgoEngine generates a paired robot demonstration consisting of: (1) a **robot observation video** that depicts the robot performing the interaction in the original source scene, and (2) an **executable robot trajectory** aligned with the task object motion. By producing observation-action pairs that remain consistent with the original human video and physically executable for the robot, EgoEngine enables high-fidelity large-scale dataset generation and supports **zero-shot** visuomotor policy learning without any robot teleoperation data.

EgoEngine addresses the core challenges of human-to-robot transfer through an object-centric visual-action generation pipeline. EgoEngine first preprocesses the video to build a digital twin that aligns the scene, object, and robot embodiment. For visual generation, EgoEngine replaces the human with the robot appearance, producing robot observation videos that preserve the original scene context and object state in each frame. For action generation, EgoEngine first retargets human motion into a robot reference trajectory, then refines it in simulation to produce an executable robot trajectory supervised by reference object motion. To improve efficiency, EgoEngine uses a Monte Carlo Tree Search-style (MCTS-style) adaptive mode-switching strategy, escalating from Replay to Model Predictive Control (MPC) and Reinforcement Learning (RL) when stronger refinement is needed. After the two branches, the generated demonstrations support downstream policy training.

Overall, our contributions are threefold:

- We treat egocentric human videos as a scalable data source of robot supervision, and present EgoEngine, a two-branch data engine for generating paired robot observation-action demonstrations from egocentric human videos.
- EgoEngine formulates human-to-robot demonstration generation as jointly bridging the visual and action gaps, and identifies executable action generation as the primary factor of downstream policy performance.
- We demonstrate the quality of the generated data through visual fidelity analysis, action evaluation in simulation, and downstream policy distillation, enabling zero-shot dexterous manipulation on a real humanoid robot without using any real robot demonstrations.

II. RELATED WORK

Human-to-Robot Action Generation. Human-to-robot action generation is commonly built on teleoperation and retargeting, where human motion is mapped to a robot embodiment [37, 18, 4, 59, 38]. Recent glove-based [57] and

hand-as-interface systems [52] improve dexterous data action quality, but remain expensive to scale and require skilled teleoperators. However, offline retargeted trajectories can be infeasible under embodiment and dynamics mismatch [24]. Some methods address this limitation with structured motion planning [15, 29, 47], but require task-specific design. Other Real2Sim2Real methods use RL [66, 22, 28, 26, 8] or MPC [32, 54] in simulation to refine trajectories by incorporating object-centric objectives. EgoEngine follows this line and focuses on executable trajectory generation by adaptively balancing solver capability and efficiency.

Human-to-Robot Video Generation. To bridge the visual gap, early methods like EgoMimic [16] mainly reduce the visual gap with lightweight arm masks, while others generate pseudo latent visual features [24]. Most methods follow an inpainting-rendering-blending pipeline to replace the human with the robot appearance in videos [19, 20, 62, 21]. Recent motion- and physics-conditioned video generation and editing methods show promise for controllable robot video synthesis [53, 6, 65, 64, 55, 14, 34], but still struggle with physically consistent high-DoF dexterous interactions. EgoEngine follows the controllable inpainting-rendering-blending pipeline and further grounds it in a digital twin for better physical consistency.

Learning from Human Videos. A series of works have explored learning robot policies from human demonstrations [48, 16, 44], aiming to bridge the visual and action gaps between humans and robots. Some methods use human videos for pretraining or high-level policy planning [56, 48], while later works formulate the problem as domain adaptation or co-training across human and robot embodiments [16, 24, 35, 68]. However, these methods still rely on robot demonstrations as the target domain, limiting their ability to learn zero-shot from human videos alone. Recent methods [9, 23] explore zero-shot policy learning from human videos, but without explicit action refinement, they are mainly demonstrated on simpler manipulation settings or lower-DoF grippers. Instead, EgoEngine turns human videos into demonstrations for zero-shot dexterous manipulation.

III. EGOENGINE FRAMEWORK

As illustrated in Fig. 1, EgoEngine converts egocentric human videos into robot-executable demonstrations for policy learning. Given a human video, EgoEngine first reconstructs an object-centric digital twin comprising camera geometry, depth, object trajectory, and human/object masks. It then runs two parallel branches: the action branch converts human motion into executable robot actions, and the visual branch converts human egocentric frames into robot-view observations. Together, they produce paired robot demonstrations $(\tilde{o}_t, \tilde{a}_t)$ for training a downstream visuomotor policy.

A. Human Video to Simulation

For each human video, we reconstruct an object-centric digital twin as follows. Human demonstrations $\tau^{(h)}$ are captured using Aria Gen2 glasses [5], which provide synchronized RGB

frames and per-frame 3D hand poses with 21 hand keypoints. We estimate absolute depth maps from the RGB frames with FoundationStereo [51]. We then use SAM2 [39] for two types of masks: hand-keypoint prompts provide human arm–hand masks for removing the demonstrator, while a first-frame point prompt tracks the task-object mask throughout the clip. Given the RGBD frames, tracked object masks, and object mesh, FoundationPose [50] estimates a temporally consistent 6D object trajectory $\{T_o^t\}_{t=1}^T$. The resulting camera geometry, depth, masks, hand poses, object mesh, and object trajectory together define the digital twin consumed by the action and visual branches.

B. Action Generation

1) *Human-Centric Retargeting*: We first retarget the human hand motion to a reference robot trajectory. Given the fingertip positions and orientations $\{(p_{\text{tip},k}^t, R_{\text{tip},k}^t)\}_{k=1}^5$ along with the wrist orientation R_{wrist}^t from the human video at time t , we solve an inverse-kinematics problem with MINK [63]:

$$q_t^* = \arg \min_{q \in \mathcal{Q}} \mathcal{L}_{\text{tip}}(q; t) + \lambda_w \mathcal{L}_{\text{wrist}}(q; t), \quad (1)$$

where $\mathcal{L}_{\text{tip}}$ and $\mathcal{L}_{\text{wrist}}$ are L2 losses aligning the robot fingertips with the human fingertip poses and the robot wrist with the human wrist orientation, and $\mathcal{Q}$ is the feasible configuration space subject to joint limits and self-collision constraints. The resulting trajectory $\tau^{\text{ref}} = \{q_t^*\}_{t=1}^T$ imitates the human motion on the robot and serves as the initial motion prior for action refinement.

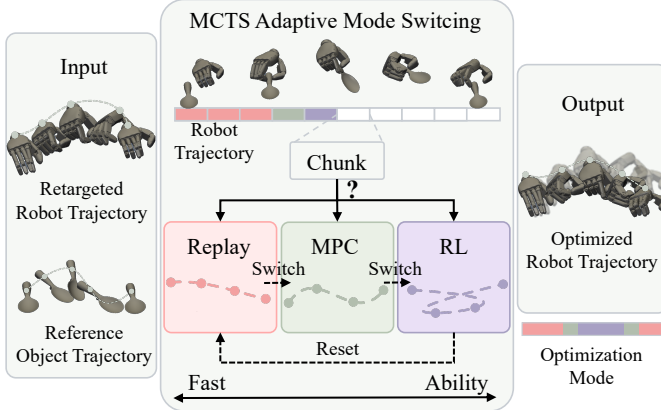


Fig. 2: **Adaptive MCTS-style switching optimization.** EgoEngine progressively selects among Replay, MPC, and RL to solve each trajectory chunk with the cheapest feasible solver.

2) *Object-Centric Trajectory Optimization*: After retargeting, EgoEngine faces a long-horizon, high-DoF trajectory optimization problem: the reference trajectory preserves the human motion prior, but must be converted into executable robot actions that reproduce the demonstrated object motion. Retargeting alone is insufficient for two reasons: (1) replaying it often fails under the human-to-robot embodiment gap, including kinematic mismatch and contact-dynamics discrepancies; and (2) human videos provide an observable proprioceptive trajectory rather than action commands, leaving a proprio-to-action gap for applying forces and sustaining contacts. We

refine the reference trajectory in simulation with an object-centric objective, using object motion extracted from human videos as the task-level target for executable robot actions.

Let T_o^t denote the 6D object pose tracked from the human video (Sec. III-A), and let $\hat{T}_o^t$ denote the corresponding object pose in simulation under the robot’s executed control. We define the object pose tracking error as:

$$e^t = \left(\lambda_p d_p \left(\text{trans}(\hat{T}_o^t), \text{trans}(T_o^t) \right)^2 + \lambda_R d_R \left(\text{rot}(\hat{T}_o^t), \text{rot}(T_o^t) \right)^2 \right)^{1/2}. \quad (2)$$

where $d_p(\cdot, \cdot)$ is the Euclidean distance in $\mathbb{R}^3$ and $d_R(\cdot, \cdot)$ is the geodesic distance on $\text{SO}(3)$. We use a threshold object-tracking objective with early termination: the episode terminates if e^t exceeds a threshold C . Within the valid regime ($e^t \leq C$), the reward is $r_{\text{obj}}^t = C - e^t$, so lower error gives higher reward while rewards remain non-negative until termination. This object-centric reward is combined with auxiliary terms for contact, action smoothness, and human-mimetic regularization (see Appendix for details).

Solver modes. EgoEngine decomposes the long-horizon trajectory into temporal chunks and optimizes each chunk progressively. For each chunk, EgoEngine uses three solver modes with increasing optimization capability. *Replay* directly executes the reference robot trajectory. *MPC* searches over short-horizon action samples around the reference trajectory, providing local correction at a moderate cost. *RL* trains a hand residual policy for difficult chunks. The policy observes the hand state, object pose, and reference retargeted command, and predicts a residual $\delta a_t \sim \pi_\phi(\cdot | s_t)$ added to the reference command, $a_t = a_t^{\text{base}} + \delta a_t$. We optimize π_ϕ with PPO [40, 41], consistent with residual RL and hybrid control formulations [1].

MCTS-style Adaptive Mode Switching. Unlike prior dexterous RL methods that learn universal or chained policies for long-horizon manipulation [2], often requiring extensive reward engineering [11, 27] or large-scale simulation [58], EgoEngine uses a Monte Carlo Tree Search (MCTS)-style mode-switching strategy [3]. Here, MCTS-style refers to a lightweight, heuristic-based progressive search over solver modes, rather than a full MCTS algorithm. Instead of applying the strongest solver to the full trajectory, in each chunk, EgoEngine starts from Replay and escalates to MPC or RL only when the current mode cannot produce a feasible or sufficiently improved rollout, as shown in Fig. 2. Replay is retained when the reference trajectory satisfies the object-centric criterion; otherwise, EgoEngine invokes MPC for local correction and falls back to RL for stronger refinement when MPC remains insufficient. To avoid local minima from optimizing each chunk in isolation, EgoEngine uses a two-chunk optimization window, jointly solving the current and next chunks but executing only the current chunk once both are feasible. This design improves efficiency by avoiding unnecessary refinement and eliminating the need to maintain an RL policy over the full trajectory.

C. Visual Generation

Human removal. The raw egocentric frame $I_t^{(h)}$ shows the human arms and hands performing the task; these regions must be removed before compositing in a robot rendering. We mask the arm-hand regions using the SAM2 masks from Sec. III-A and fill them with Inpaint-Anything v2 [61], recovering the scene and object content occluded by the demonstrator. This yields a demonstrator-free frame $\bar{I}_t$ retaining the scene and target object, onto which the robot is composited.

Occlusion-aware blending. Given the robot trajectory from the action branch, we render the robot at the egocentric viewpoint to obtain R_t , and recover an occlusion-aware mask via two-pass differential rendering. Keeping object geometry opaque in both passes, we render the scene with the robot fully transparent (I_{bg}^t) and with the robot opaque (I_{rob}^t); The mask is computed per pixel p from the thresholded RGB difference:

$$\tilde{M}_r^t(p) = \mathbf{1}[\|I_{rob}^t(p) - I_{bg}^t(p)\| > 0]. \quad (3)$$

Since objects are present in both passes, this implicitly removes the occluded robot pixels, yielding $\tilde{M}_r^t$ as the visible robot mask. The final generated observation is blended with:

$$\tilde{o}_t^{(r)} = \tilde{M}_r^t \odot R_t + (1 - \tilde{M}_r^t) \odot \bar{I}_t. \quad (4)$$

D. Policy Distillation

After the action and visual branches, each human video is converted into a robot demonstration consisting of synchronized robot observation and action. Aggregating these across videos yields a synthetic robot dataset $\mathcal{D}_{robot} = \{(\tilde{o}, \tilde{a})\}$, where $\tilde{o}$ denotes the generated robot observation from the visual branch with proprioception, and $\tilde{a}$ denotes the corresponding action from the action branch. We train a visuomotor policy π_θ with HPT [49] on this dataset with an ℓ_2 action regression loss:

$$\min_{\theta} \mathbb{E}_{(\tilde{o}, \tilde{a}) \sim \mathcal{D}_{robot}} [\|\pi_\theta(\tilde{o}) - \tilde{a}\|_2^2]. \quad (5)$$

The policy distills both branches into a closed-loop controller mapping robot observation to action.

IV. EXPERIMENTS

A. Experiment Setup

We structure our evaluation around three questions. First, **are the generated robot observations visually aligned with real robot observations?** We compare synthesized and real robot observations in appearance and representation space (Sec. IV-B). Second, **are the generated robot actions executable and task-aligned?** We test whether they reproduce demonstrated object interactions in simulation on TACO and Aria (Sec. IV-C). Third, **do the generated observation-action pairs support zero-shot policy learning?** We train policies from EgoEngine-generated demonstrations and evaluate them on the real robot (Sec. IV-D).

Datasets and settings. We use two data sources for evaluation. TACO [25] provides 2,500 video sequences for visual and action evaluation. We collect the Aria dataset with Aria Gen2 Glasses [5], containing 200 real-world egocentric human

videos across four tasks. For comparison, we also collect 200 real-robot teleoperation demonstrations, which are not used by EgoEngine. In simulation, we use a bimanual RB-Y1 with two 7-DoF arms and two 12-DoF XHands. On the real robot, we use a single-arm RB-Y1 with one XHand.

Tasks. In simulation, we evaluate on 16 selected demonstration pairs from TACO that are compatible with our robot embodiment and digital-twin reconstruction pipeline. In the real world, we evaluate on four Aria tasks: *Mustard* (placing a mustard bottle onto a target plate), *Drawer* (opening a drawer and placing a cube inside), *Hammer* (picking up a hammer and striking a nail), and *Flower* (holding a water bottle to a flower). Together, these tasks cover diverse interaction patterns, including pick-and-place, tool use, and fine contact.

Metrics. We use metrics matched to each evaluation question. For visual fidelity, we report qualitative comparisons and feature-space alignment with real robot observations. For action fidelity, we report success rate, object-tracking reward, and generation cost. For downstream policy performance, we report real-robot success rate under fixed rollout trials.

B. Visual Fidelity of Generated Robot Observations

We evaluate visual fidelity by measuring how well EgoEngine synthesizes robot egocentric observations that match real robot observations. We compare EgoEngine with several representative human-to-robot video editing baselines: (a) the original **human videos**, (b) **EgoMimic** [16], which masks the human region toward a robot-like color, (c) **VACE** [14], built on WAN2.1 [46], a video generation model conditioned on video, text, and image template prompts, and (d) **Phantom** [19], which directly blends a rendered robot into the scene without modeling occlusion or physical contact.

EgoEngine generates robot observations that are physically consistent and closely aligned with real robot observations. We validate the conclusion through both qualitative comparisons and feature-space analysis. Qualitative comparisons in Fig. 3 show that EgoEngine produces more physically consistent robot observations than the baselines, especially around robot-object contact and visibility ordering. These examples evaluate appearance realism, physical consistency, and contact/occlusion synthesis under representative tasks. We further analyze feature-space alignment with real robot observations using three fixed pretrained encoders: ResNet18 [10] (used as the policy encoder), VGG16 [43], and DINOv2 [30]. We extract per-frame features and compare the distributions of synthesized observations and real robot observations. As shown in Fig. 4, EgoEngine features are closer to the real-robot feature distribution than the original human videos in KDE [33] visualization. Tab. I quantifies this: EgoEngine reduces Fréchet Distance (FD) from the last layer across encoders versus baselines, except on DINOv2, where it matches Phantom (473.1 vs. 470.6).

C. Action Fidelity of Generated Robot Trajectories

We evaluate whether EgoEngine can convert retargeted human motion into executable robot trajectories. We compare

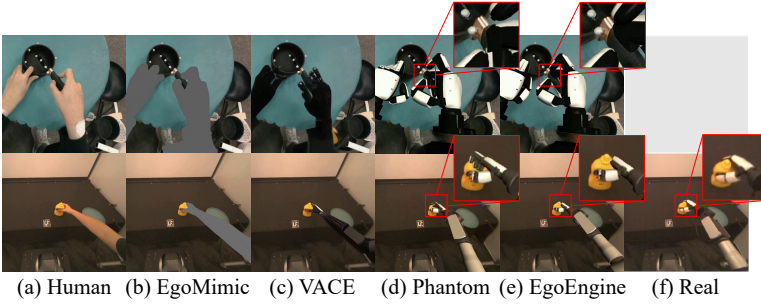


Fig. 3: Two samples from TACO and Aria Dataset. Qualitative visual comparison between (a) original human video, (b) EgoMimic inpainting, (c) VACE (WAN2.1), (d) Phantom, (e) EgoEngine (Ours), (f) Real teleoperation videos. Compared to (d), EgoEngine produces more physically consistent robot observations with the digital twin.

EgoEngine with three action-generation modes: (1) **Mink (Replay)** [63], which directly executes the retargeted reference trajectory, (2) **Spider (MPC)** [32], which performs local trajectory optimization; and (3) **H2S2R (RL)** [26], which learns strong residual refinement. We report success rate (*SR*), normalized rollout progress before early termination (*Step*), and normalized object-tracking reward relative to the perfect reference trajectory (*Reward*). We also measure generation cost with a device-agnostic metric, the average simulation steps required to optimize a single successful trajectory timestep (*Cost*).

EgoEngine generates executable, task-aligned robot trajectories. As shown in Tab. II, Replay is fast but often fails under embodiment mismatch and contact-rich dynamics. MPC improves over Replay on some TACO trajectories, especially for pushing or reorientation motions, but remains unreliable for grasp-centric stages in the Aria tasks. For Aria, these two only succeed for some Drawer demos. RL achieves stronger raw performance, indicating that difficult contact-rich interactions require stronger trajectory refinement. In contrast, EgoEngine preserves strong trajectory quality across both TACO and Aria, showing that it can convert retargeted human motion into executable and task-aligned robot trajectories.

EgoEngine improves the quality-efficiency tradeoff through MCTS-style adaptive mode switching, especially for long-horizon tasks. As shown in Tab. II, EgoEngine preserves strong trajectory quality while reducing generation cost compared with full RL refinement. The efficiency gain comes from assigning solver capacity only where it is needed: Fig. 6 shows that easier chunks are mainly handled by Replay and MPC, while difficult contact-rich chunks fall back to RL. MPC is selected less frequently than Replay and RL, suggesting that local correction is useful for a subset of chunks but insufficient for many contact-rich stages. In Fig. 5, EgoEngine improves Aria generation efficiency by 22.0%, from 2.36 demos/hour with RL to 2.88 demos/hour on a single RTX 4090 without parallelization. On TACO, a long-horizon bimanual dataset with harder and more complex tasks, the average trajectory length is 327.5 timesteps, which is $2.39\times$

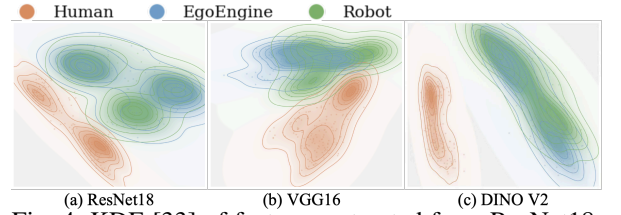


Fig. 4: KDE [33] of features extracted from ResNet18, VGG16, and DINOv2.

Method (FD↓)	ResNet18	VGG16	DINOv2
Human Video	764.5	670.2	602.9
EgoMimic	830.5	812.1	579.6
VACE	713.6	745.3	488.0
Phantom	620.0	650.8	470.6
EgoEngine	614.7	644.2	473.1

TABLE I: Fréchet Distance (FD) of feature similarity comparison across methods.

that of Aria; EgoEngine therefore achieves larger efficiency gains by avoiding full-trajectory refinement when cheaper solvers are sufficient. Together, these results show that EgoEngine preserves strong trajectory quality while substantially improving generation efficiency, especially as task horizon and complexity increase.

D. Downstream Policy Distillation

For each Aria task, we train the visuomotor policy from EgoEngine-generated data and evaluate it on the real robot. Tab. III compares EgoEngine with (1) **Human videos**, (2) **Phantom** [19], which performs video editing during both training and inference, and (3) **Real robot** teleoperation demonstrations. All policies share the same architecture, demonstration count, and evaluation protocol.

EgoEngine-generated demonstrations support zero-shot real-robot policy learning from egocentric human videos. Human videos and Phantom videos

Method	SR ↑
Human Videos	0.03
+Visual branch	0.05
+Action branch	0.43
EgoEngine	0.51

achieve near-zero success and mostly fail due to grasping pose, indicating that visual conversion alone is insufficient for dexterous policy learning. In contrast, EgoEngine achieves non-trivial zero-shot performance across tasks and matches or exceeds real robot demonstrations on 2 of 4 tasks. Fig. 7 (a, b) provide two example behaviors supporting this result. For *Hammer*, Real Robot reaches a low SR, as unintended early contacts often disturb the hammer before grasping. EgoEngine instead refines the grasp with a slight wrist rotation, leading to a more stable hammer grasp. The learned policy can recognize task state, recover from intermediate failures, and adapt to object pose variations, suggesting that EgoEngine-generated demonstrations provide usable supervision for real-robot visuomotor policy learning.

TABLE IV: Ablation of the visual and action branches on downstream success rate (averaged over four Aria tasks).

Method	TACO				Aria			
	SR $\uparrow$	Step $\uparrow$	Reward $\uparrow$	Cost $\downarrow$	SR $\uparrow$	Step $\uparrow$	Reward $\uparrow$	Cost $\downarrow$
Mink [63]	0.17	0.29	0.29	1.00	0.10	0.66	0.62	1.00
Spider [32]	0.25	0.42	0.39	7,923	0.20	0.69	0.65	4,382
H2S2R [26]	0.83	0.86	0.70	73,675	0.90	0.94	0.85	20,237
EgoEngine	0.83	0.84	0.67	34,842	0.90	0.91	0.83	16,560

TABLE II: Simulation results of action generation on TACO (16 demonstration pairs) and Aria (4 tasks). Metrics: success rate (SR), rollout completion step ratio (Step), object-tracking reward ratio vs. perfect trajectories (Reward), and average simulation steps per trajectory timestep (Cost).

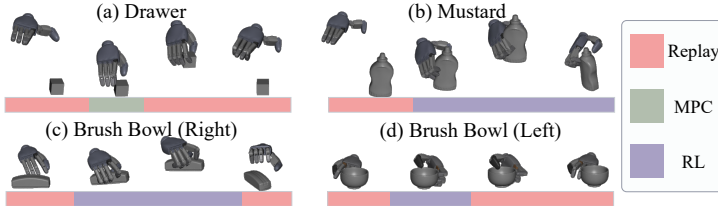


Fig. 6: Visualization of Adaptive MCTS-style mode switching across trajectory chunks on four demonstrations from the Aria and TACO datasets.

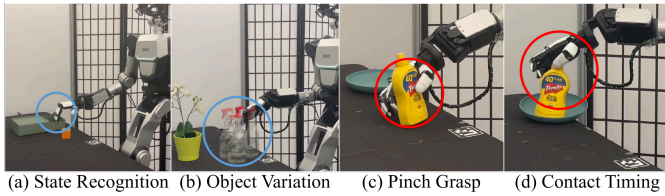


Fig. 7: Behavioral strengths and failure modes of EgoEngine. The policy handles (a) state recognition/recovery and (b) object variation; failures are mainly (c) unstable pinch grasps and (d) contact-timing mismatch.

Executable action generation provides the primary improvement, while visual generation provides an additional gain. We further perform an ablation study over four settings on all four Aria tasks: Human videos, visual branch only (+Visual branch), action branch only (+Action branch), and full EgoEngine, to isolate how visual and action generation contribute to policy performance. Tab. IV shows that removing the action branch causes the largest performance drop, leaving only a few successful trials on Drawer. This highlights the importance of action generation: without action refinement, the retargeted proprioceptive trajectory either fails to apply sufficient contact forces for grasping or produces incorrect object orientations after grasping, resulting in poor object interactions for policy learning. Removing the visual branch still preserves a substantial SR, with only a small drop from full EgoEngine. This trend is consistent with other works [62], where policies can tolerate moderate embodiment appearance mismatch and the visual encoder could be object-centric. Fig. 7(c,d) further shows that the remaining failures are mainly action-side errors, such as unstable grasps for Mustard and mistimed Drawer contacts, where the contact reward encourages premature contact or late release.

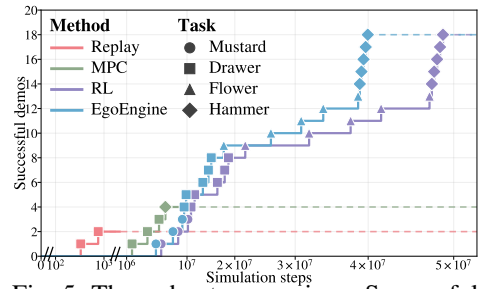


Fig. 5: Throughput comparison: Successful demo number generated over simulation steps on 20 Aria demos across 4 tasks.

Method	Mustard	Drawer	Flower	Hammer
Human Video	0.00	0.10	0.00	0.00
Phantom [19]	0.00	0.05	0.00	0.00
Real Robot	0.80	0.80	0.70	0.25
EgoEngine	0.40	0.35	0.70	0.60

TABLE III: Policy success rate (SR) on four Aria tasks, comparing EgoEngine with Human Videos (direct retargeting), Phantom, and real robot teleoperation.

V. CONCLUSION

We present EgoEngine, a scalable data engine that converts egocentric human videos into paired robot demonstrations, consisting of executable action trajectories and aligned egocentric robot observations. By jointly grounding action generation and visual synthesis through object-centric reconstruction and physically feasible optimization, EgoEngine turns egocentric human videos into training data suitable for dexterous robot learning. Across simulation, visual comparison, and real-robot policy distillation, our results show that EgoEngine produces high-quality demonstrations that enable zero-shot policy learning on the real robot.

VI. LIMITATIONS

EgoEngine still has limitations along three axes. (1) **Quality.** EgoEngine already generates demonstrations that support zero-shot downstream policy learning, but the visual branch currently uses blending-based synthesis rather than fully learned photorealism, while action generation can still suffer from contact modeling errors and the sim-to-real gap. (2) **Scalability.** EgoEngine scales with human video collection, but building the digital twin remains a bottleneck. Obtaining high-quality object assets, estimating object states under severe occlusion, and handling deformable objects remain challenging. (3) **Efficiency.** Simulation-based trajectory optimization is still slow at very large scale, though trajectories run in parallel; future work could use pretrained models to speed it up.

REFERENCES

- [1] Lars Ankile, Anthony Simeonov, Idan Shenfeld, Marcel Torne, and Pulkit Agrawal. From imitation to refinement – residual rl for precise assembly, 2024. URL <https://arxiv.org/abs/2407.16677>.
- [2] Yuanpei Chen, Chen Wang, Li Fei-Fei, and C. Karen Liu. Sequential dexterity: Chaining dexterous policies for long-horizon manipulation, 2023. URL <https://arxiv.org/abs/2309.00987>.
- [3] Shuo Cheng and Danfei Xu. League: Guided skill learning and abstraction for long-horizon manipulation, 2023. URL <https://arxiv.org/abs/2210.12631>.
- [4] Xuxin Cheng, Jialong Li, Shiqi Yang, Ge Yang, and Xiaolong Wang. Open-television: Teleoperation with immersive active visual feedback, 2024. URL <https://arxiv.org/abs/2407.01512>.
- [5] Jakob Engel, Kiran Somasundaram, Michael Goesele, Albert Sun, Alexander Gamino, Andrew Turner, Arjang Talattof, Arnie Yuan, Bilal Souti, Brigid Meredith, Cheng Peng, Chris Sweeney, Cole Wilson, Dan Barnes, Daniel DeTone, David Caruso, Derek Valleroy, Dinesh Ginjupalli, Duncan Frost, Edward Miller, Elias Mueggler, Evgeniy Oleinik, Fan Zhang, Guruprasad Somasundaram, Gustavo Solaira, Harry Lanaras, Henry Howard-Jenkins, Huixuan Tang, Hyo Jin Kim, Jaime Rivera, Ji Luo, Jing Dong, Julian Straub, Kevin Bailey, Kevin Eickenhoff, Lingni Ma, Luis Pesqueira, Mark Schwesinger, Maurizio Monge, Nan Yang, Nick Charron, Nikhil Raina, Omkar Parkhi, Peter Borschowa, Pierre Moulon, Prince Gupta, Raul Mur-Artal, Robbie Pennington, Sachin Kulkarni, Sagar Miglani, Santosh Gondi, Saransh Solanki, Sean Diener, Shangyi Cheng, Simon Green, Steve Saarinen, Suvam Patra, Tassos Mourikis, Thomas Whelan, Tripti Singh, Vasileios Balntas, Vijay Baiyya, Wilson Dreewes, Xiaqing Pan, Yang Lou, Yipu Zhao, Yusuf Mansour, Yuyang Zou, Zhaoyang Lv, Zijian Wang, Mingfei Yan, Carl Ren, Renzo De Nardi, and Richard Newcombe. Project aria: A new tool for egocentric multi-modal ai research, 2023. URL <https://arxiv.org/abs/2308.13561>.
- [6] Daniel Geng, Charles Herrmann, Junhwa Hur, Forrester Cole, Serena Zhang, Tobias Pfaff, Tatiana Lopez-Guevara, Carl Doersch, Yusuf Aytar, Michael Rubinstein, et al. Motion prompting: Controlling video generation with motion trajectories. *arXiv preprint arXiv:2412.02700*, 2024.
- [7] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18995–19012, 2022.
- [8] Zhaoyuan Gu, Yipu Chen, Zimeng Chai, Alfred Cueva, Thong Nguyen, Yifan Wu, Huishu Xue, Minji Kim, Isaac Legene, Fukang Liu, Matthew Kim, Ayan Barula, Yongxin Chen, and Ye Zhao. Refine-dp: Diffusion policy fine-tuning for humanoid loco-manipulation via reinforcement learning, 2026. URL <https://arxiv.org/abs/2603.13707>.
- [9] Yunhai Han, Zhenyang Chen, Kyle A Williams, and Harish Ravichandar. Learning prehensile dexterity by imitating and emulating state-only observations, 2024. URL <https://arxiv.org/abs/2404.05582>.
- [10] Kaiping He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, pages 770–778, 2016.
- [11] Tairan He, Zi Wang, Haoru Xue, Qingwei Ben, Zhengyi Luo, Wenli Xiao, Ye Yuan, Xingye Da, Fernando Castañeda, Shankar Sastry, Changliu Liu, Guanya Shi, Linxi Fan, and Yuke Zhu. Viral: Visual sim-to-real at scale for humanoid loco-manipulation, 2025. URL <https://arxiv.org/abs/2511.15200>.
- [12] Ryan Hoque, Peide Huang, David J. Yoon, Mouli Siva-purapu, and Jian Zhang. Egodex: Learning dexterous manipulation from large-scale egocentric video, 2026. URL <https://arxiv.org/abs/2505.11709>.
- [13] Balasubramanian, Sivakumar and Melendez-Calderon, Alejandro and Roby-Brami, Agnes and Burdet, Etienne. On the analysis of movement smoothness. *Journal of NeuroEngineering and Rehabilitation*, 12(1):112, 2015. doi: 10.1186/s12984-015-0090-9. URL <https://doi.org/10.1186/s12984-015-0090-9>.
- [14] Zeyinzi Jiang, Zhen Han, Chaojie Mao, Jingfeng Zhang, Yulin Pan, and Yu Liu. Vace: All-in-one video creation and editing, 2025. URL <https://arxiv.org/abs/2503.07598>.
- [15] Zhenyu Jiang, Yuqi Xie, Kevin Lin, Zhenjia Xu, Weikang Wan, Ajay Mandlekar, Linxi Fan, and Yuke Zhu. Dexmimicgen: Automated data generation for bimanual dexterous manipulation via imitation learning, 2025. URL <https://arxiv.org/abs/2410.24185>.
- [16] Simar Kareer, Dhruv Patel, Ryan Punamiya, Pranay Mathur, Shuo Cheng, Chen Wang, Judy Hoffman, and Danfei Xu. Egomimic: Scaling imitation learning via egocentric video. *arXiv preprint arXiv:2410.24221*, 2024.
- [17] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.
- [18] Chuizheng Kong, Yunho Cho, Wonsuhk Jung, Idris Wibowo, Parth Shinde, Sundhar Vinodh-Sangeetha, Long Kiu Chung, Zhenyang Chen, Andrew Mattei, Advait Nidumukkala, Alexander Elias, Danfei Xu, Taylor Higgins, and Shreyas Kousik. A closed-form geometric retargeting solver for upper body humanoid robot teleoperation, 2026. URL <https://arxiv.org/abs/2602.01632>.
- [19] Marion Lepert, Jiaying Fang, and Jeannette Bohg. Phan-

- tom: Training robots without robots using only human videos. In *Conference on Robot Learning (CoRL)*, Seoul, Korea, 2025.
- [20] Marion Lepert, Jiaying Fang, and Jeannette Bohg. Masquerade: Learning from in-the-wild human videos using data-editing. In *2026 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2026.
- [21] Guangrun Li, Yaoxu Lyu, Zhuoyang Liu, Chengkai Hou, Yinda Xu, Jieyu Zhang, and Shanghang Zhang. H2r: A human-to-robot data augmentation for robot pre-training from videos. *arXiv preprint arXiv:2505.11920*, 2025.
- [22] Toru Lin, Kartik Sachdev, Linxi Fan, Jitendra Malik, and Yuke Zhu. Sim-to-real reinforcement learning for vision-based dexterous manipulation on humanoids, 2025. URL <https://arxiv.org/abs/2502.20396>.
- [23] Vincent Liu, Ademi Adeniji, Haotian Zhan, Raunag Bhirangi, Pieter Abbeel, and Lerrel Pinto. Egozero: Robot learning from smart glasses. *arXiv preprint arXiv:2505.20290*, 2025.
- [24] Yangcen Liu, Woo Chul Shin, Yunhai Han, Zhenyang Chen, Harish Ravichandar, and Danfei Xu. Immimic: Cross-domain imitation from human videos via mapping and interpolation, 2025. URL <https://arxiv.org/abs/2509.10952>.
- [25] Yun Liu, Haolin Yang, Xu Si, Ling Liu, Zipeng Li, Yuxiang Zhang, Yebin Liu, and Li Yi. Taco: Benchmarking generalizable bimanual tool-action-object understanding. *arXiv preprint arXiv:2401.08399*, 2024.
- [26] Tyler Ga Wei Lum, Olivia Y Lee, C Karen Liu, and Jeannette Bohg. Crossing the human-robot embodiment gap with sim-to-real rl using one human demonstration. *arXiv preprint arXiv:2504.12609*, 2025.
- [27] Yecheng Jason Ma, William Liang, Guanzhi Wang, De-An Huang, Osbert Bastani, Dinesh Jayaraman, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Eureka: Human-level reward design via coding large language models, 2024. URL <https://arxiv.org/abs/2310.12931>.
- [28] Zhao Mandi, Yifan Hou, Dieter Fox, Yashraj Narang, Ajay Mandlekar, and Shuran Song. Dexmachina: Functional retargeting for bimanual dexterous manipulation, 2025. URL <https://arxiv.org/abs/2505.24853>.
- [29] Juncheng Mu, Sizhe Yang, Yiming Bao, Hojin Bae, Tianming Wei, Linning Xu, Boyi Li, Huazhe Xu, and Jiangmiao Pang. Deximit: Learning bimanual dexterous manipulation from monocular human videos, 2026. URL <https://arxiv.org/abs/2602.10105>.
- [30] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research (TMLR)*, 2024.
- [31] Abby O’Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya 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.
- [32] Chaoyi Pan, Changhao Wang, Haozhi Qi, Zixi Liu, Homanga Bharadhwaj, Akash Sharma, Tingfan Wu, Guanya Shi, Jitendra Malik, and Francois Hogan. Spider: Scalable physics-informed dexterous retargeting, 2026. URL <https://arxiv.org/abs/2511.09484>.
- [33] Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.
- [34] Shivansh Patel, Shraddha Mohan, Hanlin Mai, Unnat Jain, Svetlana Lazebnik, and Yunzhu Li. Robotic manipulation by imitating generated videos without physical demonstrations, 2025. URL <https://arxiv.org/abs/2507.00990>.
- [35] Ryan Punamiya, Dhruv Patel, Patcharapong Aphiwetsa, Pranav Kuppili, Lawrence Y. Zhu, Simar Kareer, Judy Hoffman, and Danfei Xu. Egobridge: Domain adaptation for generalizable imitation from egocentric human data, 2025. URL <https://arxiv.org/abs/2509.19626>.
- [36] Ryan Punamiya, Simar Kareer, Zeyi Liu, Josh Citron, Ri-Zhao Qiu, Xiongyi Cai, Alexey Gavryushin, Jiaqi Chen, Davide Liconti, Lawrence Y. Zhu, Patcharapong Aphiwetsa, Baoyu Li, Aniketh Cheluva, Pranav Kuppili, Yangcen Liu, Dhruv Patel, Aidan Gao, Hye-Young Chung, Ryan Co, Renee Zbizika, Jeff Liu, Xiaomeng Xu, Haoyu Xiong, Geng Chen, Sebastiano Olani, Chenyu Yang, Xi Wang, James Fort, Richard Newcombe, Josh Gao, Jason Chong, Garrett Matsuda, Aseem Doriwala, Marc Pollefeys, Robert Katschmann, Xiaolong Wang, Shuran Song, Judy Hoffman, and Danfei Xu. Egoverse: An egocentric human dataset for robot learning from around the world, 2026. URL <https://arxiv.org/abs/2604.07607>.
- [37] Yuzhe Qin, Wei Yang, Binghao Huang, Karl Van Wyk, Hao Su, Xiaolong Wang, Yu-Wei Chao, and Dieter Fox. Anyteleop: A general vision-based dexterous robot arm-hand teleoperation system. In *Robotics: Science and Systems*, 2023.
- [38] Ri-Zhao Qiu, Shiqi Yang, Xuxin Cheng, Chaitanya Chawla, Jialong Li, Tairan He, Ge Yan, David J. Yoon, Ryan Hoque, Lars Paulsen, Ge Yang, Jian Zhang, Sha Yi, Guanya Shi, and Xiaolong Wang. Humanoid policy human policy, 2025. URL <https://arxiv.org/abs/2503.13441>.
- [39] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. SAM 2: Segment anything in images and videos. In *International Conference on Learning Representations (ICLR)*, 2025.
- [40] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL <https://arxiv.org/abs/1707.06347>.

- [41] Clemens Schwarke, Mayank Mittal, Nikita Rudin, David Hoeller, and Marco Hutter. Rsl-rl: A learning library for robotics research, 2025. URL <https://arxiv.org/abs/2509.10771>.
- [42] Junyao Shi, Zhuolun Zhao, Tianyou Wang, Ian Pedroza, Amy Luo, Jie Wang, Jason Ma, and Dinesh Jayaraman. Zeromimic: Distilling robotic manipulation skills from web videos. *arXiv preprint arXiv:2503.23877*, 2025.
- [43] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015.
- [44] Tony Tao, Mohan Kumar Srirama, Jason Jingzhou Liu, Kenneth Shaw, and Deepak Pathak. Dexwild: Dexterous human interactions for in-the-wild robot policies, 2025. URL <https://arxiv.org/abs/2505.07813>.
- [45] SAM 3D Team, Xingyu Chen, Fu-Jen Chu, Pierre Gleize, Kevin J Liang, Alexander Sax, Hao Tang, Weiyao Wang, Michelle Guo, Thibaut Hardin, Xiang Li, Aohan Lin, Jiawei Liu, Ziqi Ma, Anushka Sagar, Bowen Song, Xiaodong Wang, Jianing Yang, Bowen Zhang, Piotr Dollár, Georgia Gkioxari, Matt Feiszli, and Jitendra Malik. Sam 3d: 3dfy anything in images, 2025. URL <https://arxiv.org/abs/2511.16624>.
- [46] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwu Yu, Haiming Zhao, Jianxiao Yang, Jianyuan Zeng, Jiayu Wang, Jingfeng Zhang, Jingren Zhou, Jinkai Wang, Jixuan Chen, Kai Zhu, Kang Zhao, Keyu Yan, Lianghua Huang, Mengyang Feng, Ningyi Zhang, Pandeng Li, Pingyu Wu, Ruihang Chu, Ruili Feng, Shiwei Zhang, Siyang Sun, Tao Fang, Tianxing Wang, Tianyi Gui, Tingyu Weng, Tong Shen, Wei Lin, Wei Wang, Wei Wang, Wenmeng Zhou, Wenten Wang, Wenting Shen, Wenyuan Yu, Xianzhong Shi, Xiaoming Huang, Xin Xu, Yan Kou, Yangyu Lv, Yifei Li, Yijing Liu, Yiming Wang, Yingya Zhang, Yitong Huang, Yong Li, You Wu, Yu Liu, Yulin Pan, Yun Zheng, Yuntao Hong, Yupeng Shi, Yutong Feng, Zeyinzi Jiang, Zhen Han, Zhi-Fan Wu, and Ziyu Liu. Wan: Open and advanced large-scale video generative models, 2025. URL <https://arxiv.org/abs/2503.20314>.
- [47] Weikang Wan, Jiawei Fu, Xiaodi Yuan, Yifeng Zhu, and Hao Su. Lodestar: Long-horizon dexterity via synthetic data augmentation from human demonstrations, 2025. URL <https://arxiv.org/abs/2508.17547>.
- [48] Chen Wang, Linxi Fan, Jiankai Sun, Ruohan Zhang, Li Fei-Fei, Danfei Xu, Yuke Zhu, and Anima Anandkumar. Mimicplay: Long-horizon imitation learning by watching human play, 2023. URL <https://arxiv.org/abs/2302.12422>.
- [49] Lirui Wang, Xinlei Chen, Jialiang Zhao, and Kaiming He. Scaling proprioceptive-visual learning with heterogeneous pre-trained transformers, 2024. URL <https://arxiv.org/abs/2409.20537>.
- [50] Bowen Wen, Wei Yang, Jan Kautz, and Stan Birchfield. Foundationpose: Unified 6d pose estimation and tracking of novel objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17868–17879, 2024.
- [51] Bowen Wen, Matthew Trepte, Joseph Aribido, Jan Kautz, Orazio Gallo, and Stan Birchfield. Foundationstereo: Zero-shot stereo matching, 2025. URL <https://arxiv.org/abs/2501.09898>.
- [52] Mengda Xu, Han Zhang, Yifan Hou, Zhenjia Xu, Linxi Fan, Manuela Veloso, and Shuran Song. Dexumi: Using human hand as the universal manipulation interface for dexterous manipulation. *arXiv preprint arXiv:2505.21864*, 2025.
- [53] Tianshuo Xu, Zhifei Chen, Leyi Wu, Hao Lu, Yuying Chen, Lihui Jiang, Bingbing Liu, and Yingcong Chen. Motion dreamer: Realizing physically coherent video generation through scene-aware motion reasoning. *arXiv preprint arXiv:2412.00547*, 2024.
- [54] Lujie Yang, Xiaoyu Huang, Zhen Wu, Angjoo Kanazawa, Pieter Abbeel, Carmelo Sferrazza, C. Karen Liu, Rocky Duan, and Guanya Shi. Omniretarget: Interaction-preserving data generation for humanoid whole-body loco-manipulation and scene interaction, 2025. URL <https://arxiv.org/abs/2509.26633>.
- [55] Pei Yang, Hai Ci, Yiren Song, and Mike Zheng Shou. X-humanoid: Robotize human videos to generate humanoid videos at scale, 2025. URL <https://arxiv.org/abs/2512.04537>.
- [56] Ruihan Yang, Qinxi Yu, Yecheng Wu, Rui Yan, Borui Li, An-Chieh Cheng, Xueyan Zou, Yunhao Fang, Xuxin Cheng, Ri-Zhao Qiu, Hongxu Yin, Sifei Liu, Song Han, Yao Lu, and Xiaolong Wang. Egovla: Learning vision-language-action models from egocentric human videos, 2025. URL <https://arxiv.org/abs/2507.12440>.
- [57] Jessica Yin, Haozhi Qi, Youngsun Wi, Sayantan Kundu, Mike Lambeta, William Yang, Changhao Wang, Tingfan Wu, Jitendra Malik, and Tess Hellebrekers. Osmo: Open-source tactile glove for human-to-robot skill transfer, 2025. URL <https://arxiv.org/abs/2512.08920>.
- [58] Patrick Yin, Tyler Westenbroek, Zhengyu Zhang, Joshua Tran, Ignacio Dagnino, Eeshani Shilamkar, Numfor Mbiziwo-Tiapo, Simran Bagaria, Xinlei Liu, Galen Mullins, Andrey Kolobov, and Abhishek Gupta. Emergent dexterity via diverse resets and large-scale reinforcement learning, 2026. URL <https://arxiv.org/abs/2603.15789>.
- [59] Zhao-Heng Yin, Changhao Wang, Luis Pineda, Krishna Bodduluri, Tingfan Wu, Pieter Abbeel, and Mustafa Mukadam. Geometric retargeting: A principled, ultrafast neural hand retargeting algorithm, 2025. URL <https://arxiv.org/abs/2503.07541>.
- [60] Zhao-Heng Yin, Changhao Wang, Luis Pineda, Francois Hogan, Krishna Bodduluri, Akash Sharma, Patrick Lancaster, Ishita Prasad, Mrinal Kalakrishnan, Jitendra Malik, Mike Lambeta, Tingfan Wu, Pieter Abbeel, and Mustafa Mukadam. Dexteritygen: Foundation controller for unprecedented dexterity, 2025. URL <https://arxiv.org/>

abs/2502.04307.

- [61] Tao Yu, Runseng Feng, Ruoyu Feng, Jinming Liu, Xin Jin, Wenjun Zeng, and Zhibo Chen. Inpaint anything: Segment anything meets image inpainting, 2023. URL <https://arxiv.org/abs/2304.06790>.
- [62] Chengbo Yuan, Rui Zhou, Mengzhen Liu, Yingdong Hu, Shengjie Wang, Li Yi, Chuan Wen, Shanghang Zhang, and Yang Gao. Motiontrans: Human vr data enable motion-level learning for robotic manipulation policies, 2025. URL <https://arxiv.org/abs/2509.17759>.
- [63] Kevin Zakka. Mink: Python inverse kinematics based on MuJoCo, February 2026. URL <https://github.com/kevinzakka/mink>.
- [64] Ke Zhang, Cihan Xiao, Yiqun Mei, Jiacong Xu, and Vishal M Patel. Think before you diffuse: Llms-guided physics-aware video generation. *arXiv preprint arXiv:2505.21653*, 2025.
- [65] Tianyuan Zhang, Hong-Xing Yu, Rundi Wu, Brandon Y Feng, Changxi Zheng, Noah Snavely, Jiajun Wu, and William T Freeman. Physdreamer: Physics-based interaction with 3d objects via video generation. In *European Conference on Computer Vision*, pages 388–406. Springer, 2024.
- [66] Shuqi Zhao, Xinghao Zhu, Yuxin Chen, Chenran Li, Xiang Zhang, Mingyu Ding, and Masayoshi Tomizuka. Dexh2r: Task-oriented dexterous manipulation from human to robots. *arXiv preprint arXiv:2411.04428*, 2024.
- [67] Ruijie Zheng, Dantong Niu, Yuqi Xie, Jing Wang, Mengda Xu, Yunfan Jiang, Fernando Castañeda, Fengyuan Hu, You Liang Tan, Letian Fu, Trevor Darrell, Furong Huang, Yuke Zhu, Danfei Xu, and Linxi Fan. Egoscale: Scaling dexterous manipulation with diverse egocentric human data, 2026. URL <https://arxiv.org/abs/2602.16710>.
- [68] Lawrence Y. Zhu, Pranav Kuppili, Ryan Punamiya, Patcharapong Aphiwetsa, Dhruv Patel, Simar Kareer, Sehoon Ha, and Danfei Xu. Emma: Scaling mobile manipulation via egocentric human data, 2025. URL <https://arxiv.org/abs/2509.04443>.

APPENDIX

VII. HARDWARE AND DATA PREPROCESS

The overall data collection and robot system is illustrated in Fig. VII.1. A humanoid robot RB-Y1 equipped with XHand dexterous hands is used in our experiments to minimize the action gap. We use a bimanual setup with two arms and two XHands in simulation, and a single-arm setup with one XHand for the real-robot experiments. We collect human demonstration videos with Aria Gen2 to establish a comprehensive dataset for our study. To minimize the visual gap between human and robot demonstrations, we mount the same glasses on the robot for egocentric view perception.

Aria Gen2 Glasses. Aria Gen2 is a next-generation research device in the Project Aria program [5]. It serves as an egocentric perception platform to record human demonstrations from the wearer’s viewpoint, and is then used for robot egocentric perception, providing state-of-the-art hand tracking, SLAM, and depth estimation results.

RB-Y1 Humanoid Robot. RB-Y1 is a full-body humanoid robot that we use as the target embodiment for executing the retargeted demonstrations. It provides a whole-body manipulation platform with a torso and two arms, enabling bimanual interaction with everyday objects while maintaining stable body motion. In our system, we command the robot through a standard low-level controller interface (e.g., joint-space position/velocity control) and execute trajectories produced by our retargeting module.

XHand. XHand is a 12 DoF dexterous robotic hand, with 3 DoF on each of the thumb and index fingers, and 2 DoF on each of the remaining fingers. We mount it on the RB-Y1 end-effector to enable better fine-grained object interaction.

A. Camera Calibration

For the Aria dataset, we use AprilTag-based calibration to accurately estimate camera-to-robot extrinsics during data collection and deployment. For the TACO dataset, we directly estimate a pseudo robot base from the object position. In limited-data real-world dexterous manipulation, such calibration is a practical component for building a high-fidelity and executable human-to-robot transfer pipeline. By explicitly aligning the observation and robot frames, it improves sample efficiency and reduces optimization effort spent compensating for viewpoint mismatch. We therefore treat calibration as a standard system design choice, rather than a substantive assumption of the proposed method.

Aria: Glasses to Tag to Robot. In our real robot experiments with Aria dataset, we use an AprilTag placed on the workspace table in Fig. VII.1 as a common anchor to relate the Aria Gen2 coordinate system used by human recordings to the robot base frame used for action alignment. Because the Aria worn by the human and the Aria mounted on the robot are physically different frames, we denote them as aria_h and aria_r , respectively, and write the rigid transformation from the robot-mounted Aria frame to the robot base frame as ${}^{\text{base}}\mathbf{T}_{\text{aria}_r}$.

For human recordings, Aria Gen2 provides hand pose and object pose trajectories in the Aria coordinate system, denoted

as ${}^{\text{aria}_h}\mathbf{T}_{\text{hand}}^t$ and ${}^{\text{aria}_h}\mathbf{T}_o^t$. Since the AprilTag is visible in the Aria stream, we detect the tag and estimate its pose in the Aria frame, ${}^{\text{aria}_h}\mathbf{T}_{\text{tag}}$. We then convert all trajectories into the tag coordinate system:

$${}^{\text{tag}}\mathbf{T}_x^t = ({}^{\text{aria}_h}\mathbf{T}_{\text{tag}})^{-1} {}^{\text{aria}_h}\mathbf{T}_x^t, \quad x \in \{o, \text{hand}\}. \quad (\text{VII.1})$$

For robot data generation, Aria glasses are rigidly mounted on the robot’s head with a fixed extrinsic relative to the head link. Because the head moves with the robot body, the transform from the Aria frame to the robot base depends on the robot configuration and is obtained through forward kinematics. The robot-mounted Aria observes the same AprilTag, yielding ${}^{\text{aria}_r}\mathbf{T}_{\text{tag}}$, and forward kinematics provides the pose of the Aria device in the robot base frame, ${}^{\text{base}}\mathbf{T}_{\text{aria}_r}(q_t)$, where q_t is the robot configuration. Combining these yields the tag pose in the robot base frame:

$${}^{\text{base}}\mathbf{T}_{\text{tag}} = {}^{\text{base}}\mathbf{T}_{\text{aria}_r}(q_t) {}^{\text{aria}_r}\mathbf{T}_{\text{tag}}. \quad (\text{VII.2})$$

With ${}^{\text{base}}\mathbf{T}_{\text{tag}}$ and the human trajectories expressed in the tag frame, we obtain their robot-base representation:

$${}^{\text{base}}\mathbf{T}_x^t = {}^{\text{base}}\mathbf{T}_{\text{tag}} {}^{\text{tag}}\mathbf{T}_x^t, \quad x \in \{o, \text{hand}\}. \quad (\text{VII.3})$$

As the dataset scales up, we can avoid per-sequence AprilTag calibration by relying on a fixed workcell setup and the robot’s kinematic calibration, making explicit tag-based alignment unnecessary in large-scale collection.

TACO: Objects to Robot. For TACO, we define the robot base from the scene geometry by taking the center of the tool and target objects and applying a fixed 0.6m offset. Since TACO is specified in an object-centric manner, this rule provides a consistent way to instantiate the robot pose across episodes while preserving executability. We set the table height to 0.72 m, matching the real-world setup, so that the resulting configuration remains aligned with the physical system.

B. Robot Teleoperation



Fig. VII.2: Robot teleoperation setup. We use Meta Quest for visual teleoperation.

We collect real robot teleoperation demonstration data for reference and sample-efficiency comparison in the main paper. In Sec. XI, we compare the sample efficiency of EgoEngine-generated data with real-robot teleoperation data.

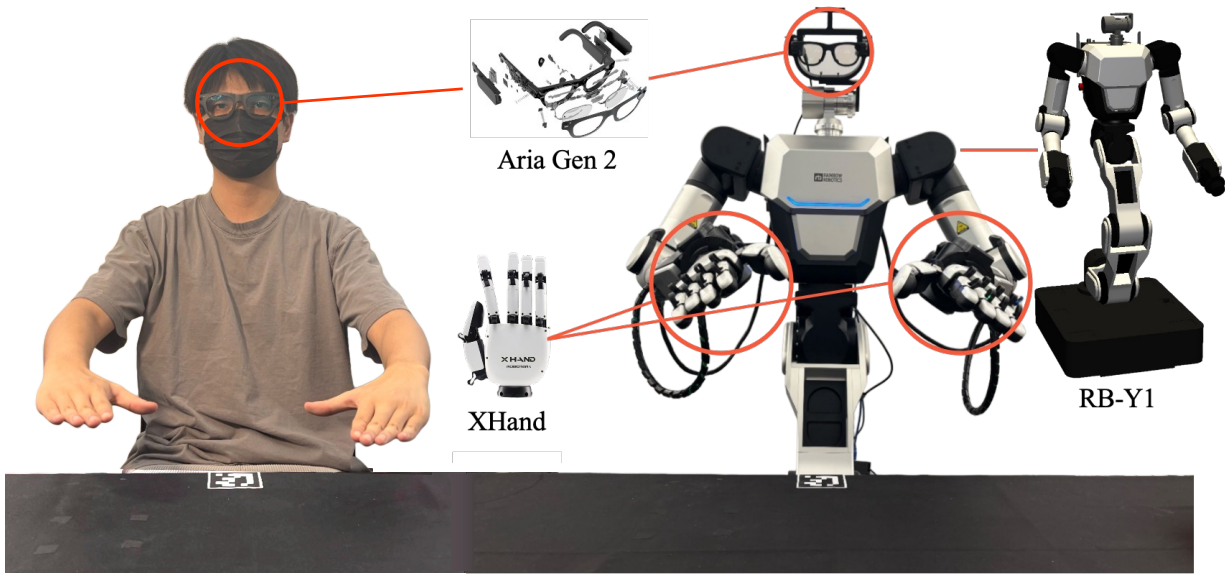


Fig. VII.1: **System overview.** We use Aria Gen2 for egocentric perception for both human and robot. Then we retarget demonstrations to the RB-Y1 humanoid equipped with XHands (bimanual in simulation, single-arm with one XHand on the real robot) for dexterous manipulation.

We use the state-of-the-art teleoperation system for the RB-Y1 robot, SEW [18], for whole-body arm retargeting, and Open-TeleVision (OpenTV) [4] for dexterous hand teleoperation, as shown in Fig. VII.2. For the upper body, SEW provides an analytical retargeting method, which we use to map the operator’s **S**houlder, **E**lbow, and **W**rist motion to the robot arm in real time. The human motion observations are streamed from the Aria glasses to the Meta Quest, and the resulting retargeted commands are used for robot whole-body control. For the hand, following OpenTV, we retarget the operator’s **H**and motion to the XHand by using the vectors from the palm to the five fingertips as the hand action representation. We represent the RB-Y1 arm as a Cartesian wrist pose and the XHand as joint positions. This design is sufficient for our data collection pipeline and provides a practical interface for recording teleoperated demonstrations.

C. Digital Twin Process

With an RGB video collected from Aria Gen2 glasses, we first apply FoundationStereo [51] to estimate per-frame depth maps. Given the resulting RGB-D observations, we then apply FoundationPose [50] together with the ground-truth object mesh to recover the object 6D pose trajectory over time.

For Aria-collected data, we align the reconstructed scene to the simulator using an AprilTag attached in the real scene, which establishes the transformation between the observation frame and the simulator frame for digital twin reconstruction. For TACO data, since no AprilTag is available, we instead estimate the robot base frame using a fixed offset heuristic and reconstruct the scene under this approximated alignment.

After initialization, the object pose trajectory estimated by FoundationPose is propagated into the simulator frame to drive

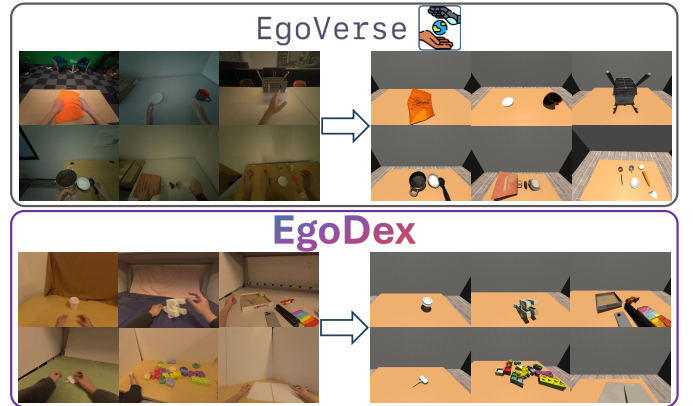
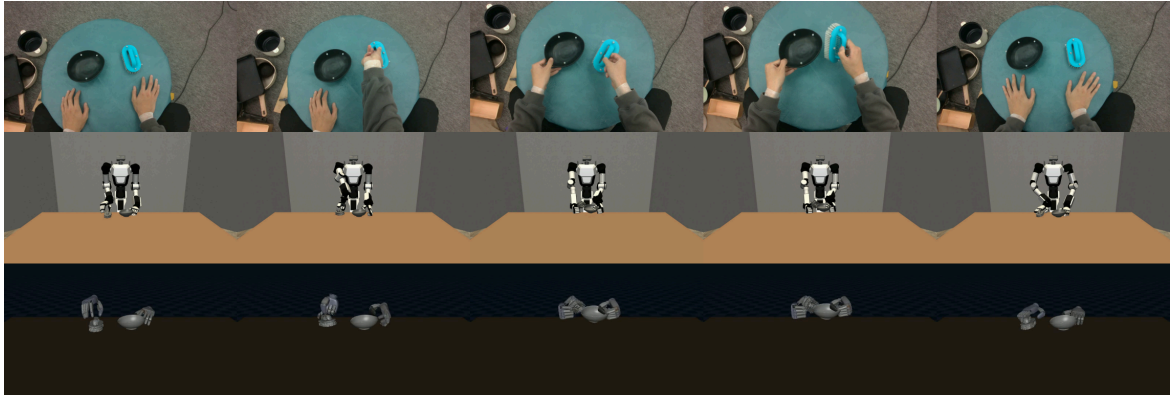


Fig. VII.3: Digital twin reconstruction examples for EgoDex and EgoVerse datasets. The system converts egocentric manipulation videos into simulation scenes covering diverse objects, layouts, and viewpoints.

digital twin reconstruction across the full sequence. Some qualitative examples of reconstructed digital twins on TACO tasks are shown in Fig. VII.4, including brush bowl, cut with spatula, skim off to plate, and smear on box. Notably, because TACO does not provide AprilTag-based calibration, these examples rely on the approximated base-frame alignment together with purely vision-based object pose estimation.

D. Scalability of Digital Twin Reconstruction

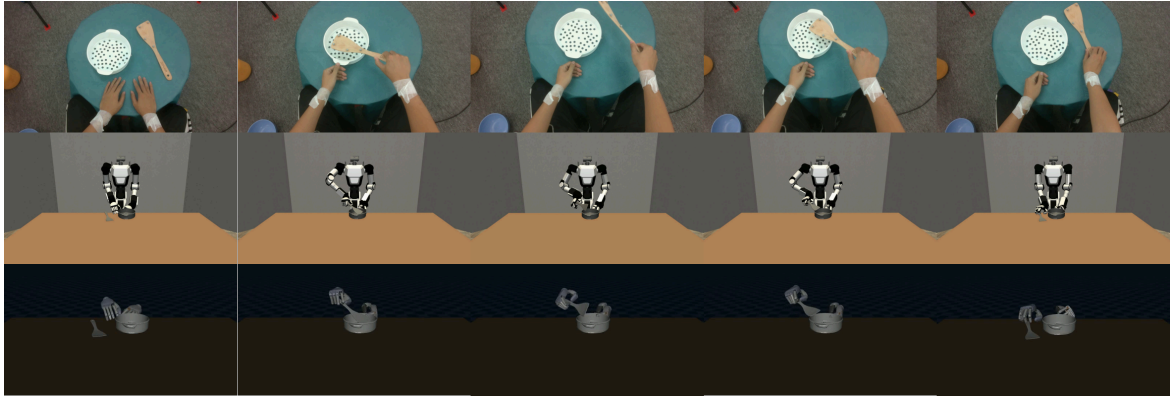
EgoEngine relies on a digital twin to bridge egocentric human videos and robot-executable demonstrations. Given an input egocentric video, the digital twin reconstruction module builds a simulation scene that preserves the task-relevant geometry, object layout, and camera viewpoint of the original interaction. Importantly, while digital twin reconstruction cur-



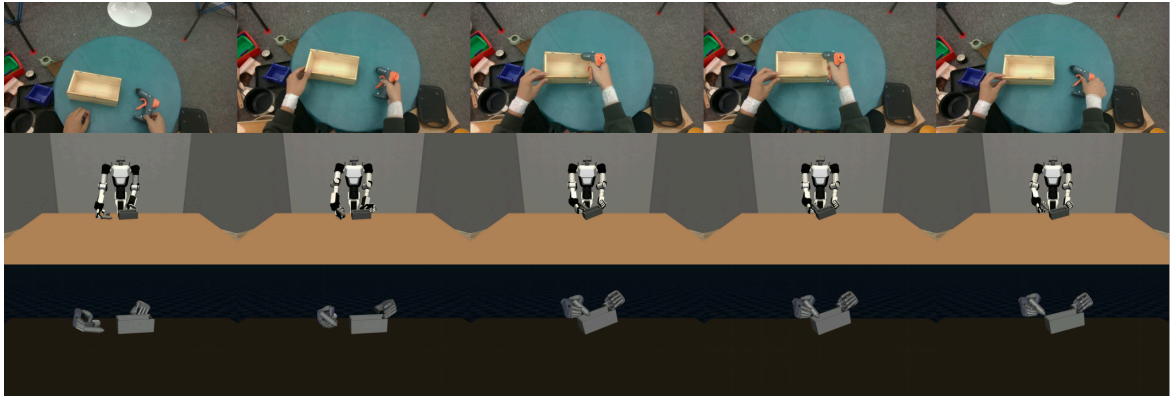
(a) Brush Bowl with Brush.



(b) Cut Bowl with Spatula.



(c) Skim off with Spatula on Plate.



(d) Smear on Box with Glue Gun.

Fig. VII.4: Digital twin reconstruction from sample TACO datasets. Starting from TACO human demonstrations, we first recover the human hand pose together with the task-relevant scene entities, including the action, active object, and other passive objects in the scene (first row). We then perform MINK-based retargeting to convert the recovered human hand motion into robot-executable hand trajectories (second row). Finally, we reconstruct a digital twin that preserves the task scene, where the action branch uses a floating Cartesian wrist/base control abstraction with the same XHand for interaction and downstream RL training (third row). (a) Brush bowl. (b) Cut with spatula. (c) Skim off to plate. (d) Smear on box.

rently remains a practical requirement of EgoEngine, this stage can be increasingly automated: recent 3D foundation models such as SAM3D [45] can be used to segment and reconstruct task-relevant objects from egocentric observations, reducing the need for manual scene modeling or object annotation. This makes digital twin construction a scalable system component rather than a fixed manual bottleneck. The reconstructed digital twin then provides the common grounding space for both branches of EgoEngine: the visual branch renders the robot into the aligned scene, while the action branch optimizes robot trajectories against the object-centric task motion.

To demonstrate the scalability of this reconstruction stage, we apply the digital twin reconstruction pipeline to egocentric manipulation videos from EgoVerse [36] and EgoDex [12]. These datasets contain diverse objects, viewpoints, and contact-rich hand-object interactions, making them suitable testbeds for assessing whether the digital twin construction process can generalize beyond manually curated scenes. Our results show that EgoEngine can recover task-relevant scene structure and object assets with sufficient fidelity to support downstream robot visual generation and action optimization.

EgoDex is a large-scale dataset and benchmark for egocentric dexterous manipulation collected with ARKit on Apple Vision Pro. The dataset has 829 hours of 30 Hz 1080p egocentric video with paired 3D pose annotations for the head, upper body, and hands as well as natural language annotation. It consists entirely of active tabletop manipulation across 194 diverse tasks.

EgoVerse is a collaborative platform for human data-driven robot learning that unifies data collection, processing, and access under a shared framework, enabling contributions from individual researchers, academic labs, and industry partners. The current release includes 1,362 hours (80k episodes) of human demonstrations spanning 1,965 tasks, 240 scenes, and 2,087 unique demonstrators, with standardized formats, manipulation-relevant annotations, and tooling for downstream learning.

In Fig. A.3, we show 12 examples from EgoDex and EgoVerse to illustrate the potential of scaling digital twin reconstruction to diverse egocentric manipulation videos. These examples cover different objects, layouts, viewpoints, and hand-object interactions, showing that the reconstruction stage is not limited to manually prepared scenes or a fixed set of task objects. By preserving task-relevant geometry and egocentric scene structure, these digital twins provide the shared representation needed by EgoEngine’s visual and action branches. This demonstrates the potential of scaling EgoEngine to large egocentric video datasets, where diverse human manipulation videos can be automatically lifted into simulation for robot demonstration generation.

VIII. QUALITATIVE RESULTS FOR VISUAL BRANCH

We provide additional qualitative results for the visual branch of EgoEngine in Fig. VIII.1. The examples cover six manipulation tasks from both the TACO and Aria datasets, and compare EgoEngine with representative human-to-robot

video generation baselines, including EgoMimic, VACE, and Phantom.

Across different tasks and viewpoints, EgoEngine produces more coherent robot-centric videos while better preserving the original scene layout, object appearance, and task-relevant interactions. In particular, compared with methods that primarily rely on direct video editing or appearance transfer, EgoEngine explicitly leverages robot rendering and scene-aware composition, which leads to more stable robot morphology, fewer visual artifacts around the manipulated objects, and better consistency between the generated robot motion and the underlying human demonstration.

These qualitative results further demonstrate that the visual branch can effectively bridge the embodiment gap between egocentric human videos and robot demonstrations. By replacing human hands with physically grounded robot embodiments while maintaining the original task context, EgoEngine generates visual demonstrations that are more suitable for downstream robot policy learning.

IX. TRAJECTORY OPTIMIZATION

A. Mode Switching

The adaptive switching in the action branch operates sequentially over fixed temporal chunks of $H = 20$ control steps, escalating complexity from Replay to MPC and RL only when lower-cost solvers fail to meet the object-centric tracking feasibility criteria. The search space can be viewed as a MCTS-style implicit tree over chunk-level mode choices, where each node corresponds to the simulator state at a chunk boundary and each branch selects one mode from Replay, MPC, and RL. Instead of using learned value estimates or full tree backup, EgoEngine applies a heuristic greedy selection from low-cost to high-cost modes.

At each chunk boundary, EgoEngine re-plans from the current simulator state and checks whether each candidate mode can satisfy the feasibility threshold over a two-chunk horizon, covering both the current chunk and the next chunk. If the rollout is feasible, EgoEngine executes only the current chunk and re-evaluates the solver mode at the next chunk boundary. Otherwise, it escalates from Replay to MPC and then to RL when stronger correction is needed. This chunk-wise strategy avoids using expensive solvers on easy segments and returns to cheaper modes once the trajectory becomes feasible again. We visualize this adaptive process in Fig. IX.1. By filtering out redundant RL refinement over extended sequences, EgoEngine improves generation efficiency while maintaining trajectory feasibility, especially for long-horizon contact-rich tasks.

B. Reward (Objective) Design

Object tracking reward and early termination. Let T_o^t be the object pose estimated from the human demonstration, and let $\hat{T}_o^t$ be the corresponding object pose in simulation. We define the translation and rotation errors as

$$e_p^t = \left\| \text{trans}(\hat{T}_o^t) - \text{trans}(T_o^t) \right\|_2, \quad (\text{IX.1})$$

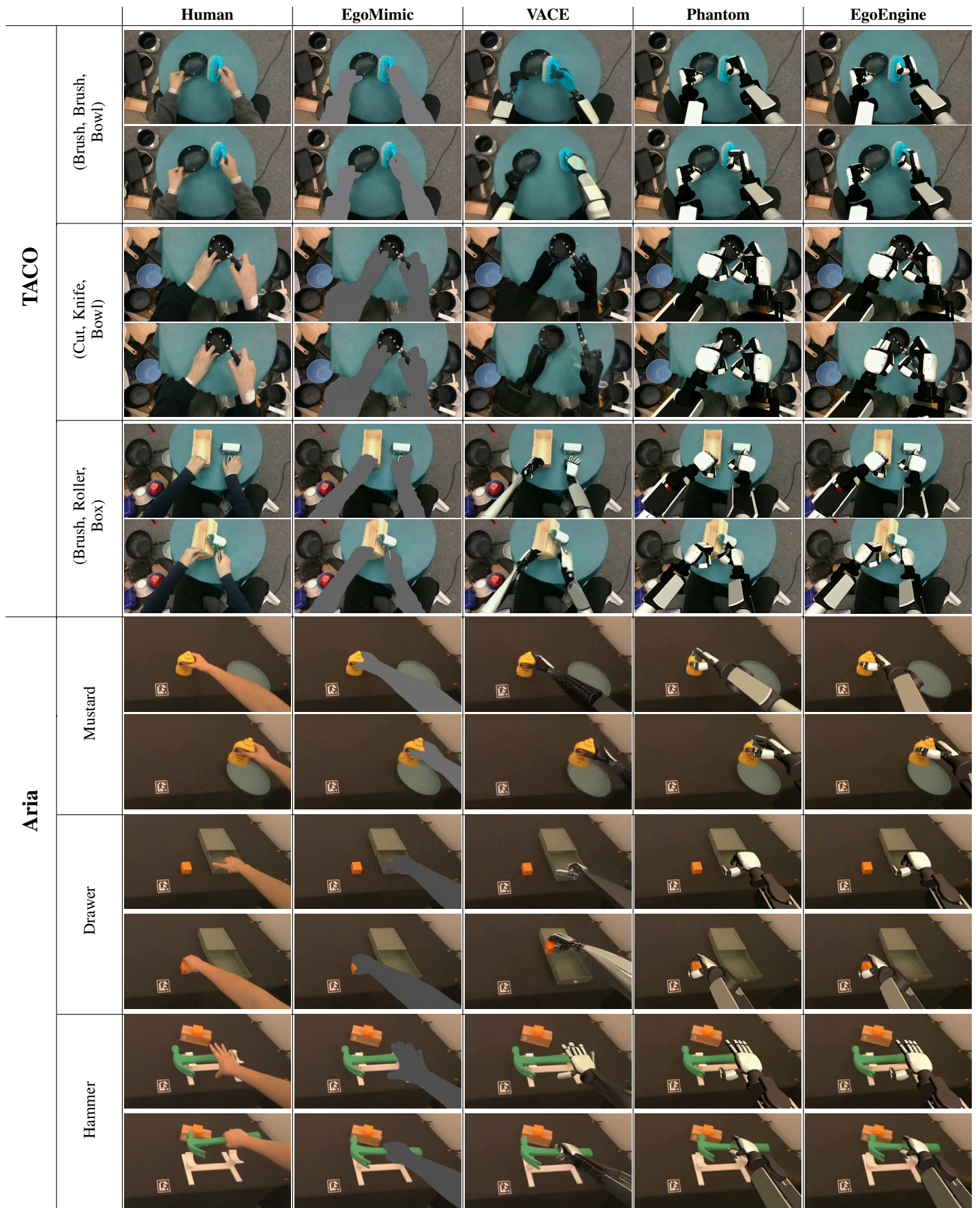


Fig. VIII.1: Visualization for different human-to-robot video generation methods. The figure shows six tasks from the TACO and Aria datasets for EgoMimic, VACE, Phantom, and EgoEngine.

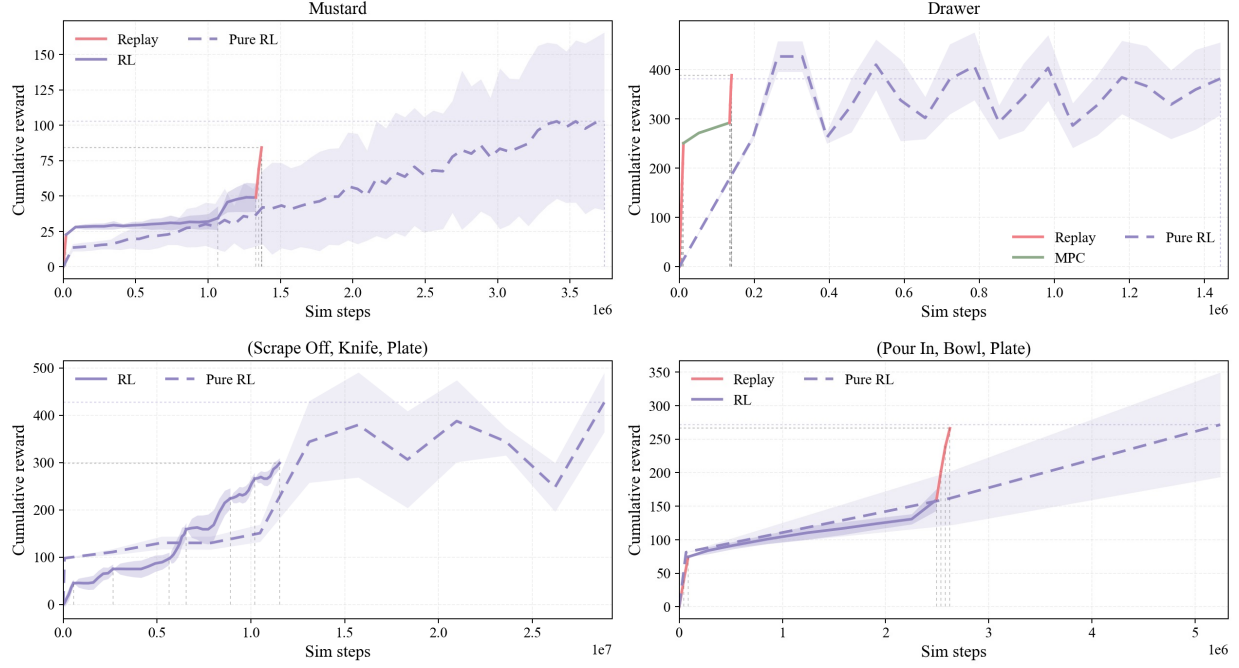


Fig. IX.1: Reward curves comparing EgoEngine with Pure RL on four long-horizon tasks: the Aria tasks Mustard and Drawer, and the TACO tasks (Scrape Off, Knife, Plate) and (Pour, Bowl, Plate). The cumulative reward is accumulated until task success, and the vertical dashed lines indicate chunk boundaries used by EgoEngine. EgoEngine combines replay, MPC, and RL-based refinement to solve each chunk, reaching task completion with substantially fewer simulation steps than Pure RL. Mustard is a challenging right-hand task, Drawer is a relatively simple left-hand task, and the two TACO tasks involve longer-horizon contact-rich manipulation.

and

$$e_R^t = d_R(\text{rot}(\hat{T}_o^t), \text{rot}(T_o^t)), \quad (\text{IX.2})$$

where $d_R(\cdot, \cdot)$ denotes the geodesic distance on $\text{SO}(3)$, defined as $d_R(R_1, R_2) = \arccos((\text{tr}(R_1^\top R_2) - 1)/2)$. The object tracking reward is then written as

$$r_{\text{obj}}^t = C - \sqrt{\lambda_R (e_R^t)^2 + \lambda_p (e_p^t)^2}, \quad (\text{IX.3})$$

where λ_R and λ_p balance rotation and translation tracking. The constant C is selected according to the early termination boundary. Specifically, if

$$\sqrt{\lambda_R (e_R^t)^2 + \lambda_p (e_p^t)^2} > C, \quad (\text{IX.4})$$

the rollout is terminated early. Thus, the same object tracking score defines both the dense reward and the feasibility boundary.

Human-mimic reward. Since action optimization uses a floating Cartesian wrist base abstraction with the same XHand joints, we regularize both the floating base pose and the finger joints toward the retargeted reference:

$$r_{\text{human}}^t = -\left(\beta_x \|x_t - x_t^{\text{retar}}\|_2^2 + \beta_R d_R(R_t, R_t^{\text{retar}})^2 + \beta_q \|q_t - q_t^{\text{retar}}\|_2^2\right). \quad (\text{IX.5})$$

where (x_t, R_t) is the floating hand base pose, q_t is the finger joint vector, and $(x_t^{\text{retar}}, R_t^{\text{retar}}, q_t^{\text{retar}})$ is the retargeted reference, and β_x , β_R , and β_q are weighting coefficients. This

reward acts as a soft prior that keeps the optimized motion close to the human-retargeted trajectory while still allowing contact-driven corrections.

Action smoothness reward. We penalize abrupt action changes between consecutive control steps:

$$r_{\text{smooth}}^t = -\|a_t - a_{t-1}\|_2^2. \quad (\text{IX.6})$$

This encourages temporally smooth control and reduces unstable contacts caused by high-frequency action oscillations.

Contact reward. For dexterous manipulation, we use a sparse binary contact bonus. Let $\mathcal{C}_{\text{thumb}}^t$ indicate whether the thumb is in contact with the manipulated object, and let $\mathcal{C}_{\text{other}}^t$ indicate whether at least one of the other four fingers is in contact with the object. The contact reward is

$$r_{\text{contact}}^t = c_{\text{contact}} \cdot \mathbf{1}[\mathcal{C}_{\text{thumb}}^t \wedge \mathcal{C}_{\text{other}}^t], \quad (\text{IX.7})$$

where c_{contact} is a constant bonus. This reward encourages stable opposition-style grasping: the thumb and at least one non-thumb finger must contact the object simultaneously to receive the bonus.

Lifting reward. For tasks that require lifting the manipulated object, we additionally encourage upward object motion along the vertical axis. Let z_o^t denote the z -coordinate of the object in simulation at timestep t , and let z_o^0 denote its initial height. The lifting reward is defined as

$$r_{\text{lift}}^t = \lambda_z (z_o^t - z_o^0), \quad (\text{IX.8})$$

where λ_z controls the strength of the lifting objective. This term provides a simple dense signal for raising the object and is only applied when vertical object displacement is relevant to the task.

We use different optimization configurations for simulation-only evaluation and real-robot transfer. For TACO, no domain randomization is applied, and the human-mimic constraint reward and action-smoothness reward are disabled. The feasibility thresholds and reward scales are set for direct simulation evaluation, where object trajectories are directly available. For the Aria real-robot tasks, the object trajectories are extracted from egocentric human videos rather than ground-truth simulator states, and therefore contain reconstruction, tracking, and calibration errors. We consequently use looser feasibility thresholds, the human-mimic constraint reward on the floating base pose, action-smoothness regularization, and stronger perturbations to make the optimized trajectories robust to real-world transfer.

Representative reward parameters. Across all tasks, we share the tracking weights, contact bonus, residual noise schedule, chunk length, and episode length. Since different tasks exhibit different object-motion ranges and contact geometries, we only adjust the object position and rotation thresholds that define the early termination boundary, while keeping all reward coefficients fixed. These thresholds are set from the reference object motion rather than searched as reward parameters.

For example, a static blade pressing on a plate (Cut, Knife, Plate) uses a tight rotation threshold (0.9 rad) to prevent the optimizer from satisfying the objective by tilting the blade away from the surface. Tasks with larger demonstrated rotations use looser rotation thresholds: Scrape Off (Knife, Plate) and Brush (Brush, Bowl) use 1.2 rad, while Pour (Bowl, Plate), which lifts and tilts the bowl by roughly 1.13 rad, uses 1.5 rad together with a tighter position threshold (0.12 m). This limited threshold adjustment preserves a shared optimization objective while allowing EgoEngine to handle tasks with different contact physics and object-motion profiles.

Domain randomization. For the four Aria real-robot tasks, we apply domain randomization during trajectory optimization and policy-based refinement to account for real-world modeling errors, calibration errors, and noisy object trajectories extracted from egocentric human videos. Compared with the simulation-only setting, we use looser feasibility thresholds, including an object position threshold of 0.08 m and an object rotation threshold of 2.5 rad, so that the optimizer can tolerate imperfect object-trajectory estimates instead of overfitting to noisy references. We set the contact reward scale to 2.0 to encourage stable object interaction. For the human-mimic constraint, we use base-position and base-rotation reward scales of 0.2 and 1.0, respectively, while setting the joint reward scale to 0.0 to avoid over-constraining noisy retargeted finger motions. We also use an action-smoothness reward scale of 0.8 to reduce high-frequency action oscillations during real-robot transfer. For robustness, we use 8 randomized rollouts with position, rotation, and joint noise scales of 0.045, 1.0, and 0.8,

respectively. We additionally randomize pair margins within $[-0.005, 0.005]$, workspace xy offsets within $[-0.015, 0.015]$, and object mass scales within $[0.8, 1.2]$. These perturbations are specific to the Aria real-robot setting and are used to keep the generated actions robust while preserving alignment with the demonstrated task.

C. Evaluation

We use **SR** (Success Rate), **Step**, **Reward**, and **Cost** for action branch evaluation in the main paper. For evaluation, we only use the object-tracking reward, which is the same score used for the object-centric objective and early termination:

$$r_{\text{eval}}^t = C - \sqrt{\lambda_R(e_R^t)^2 + \lambda_P(e_P^t)^2}. \quad (\text{IX.9})$$

A rollout is terminated if the corresponding tracking error exceeds the boundary C . No auxiliary terms, including human-mimic regularization, action smoothness, or contact bonus, are used during evaluation. These auxiliary rewards are only used during optimization to improve stability and contact quality.

For a reference trajectory of length T_i , let τ_i denote the number of valid rollout steps before success or early termination. To calculate **SR**, a trajectory is counted as successful only if it completes the full reference horizon without violating the object-tracking boundary. Given N evaluated trajectories, we define:

$$\text{SR} = \frac{1}{N} \sum_{i=1}^N \mathbb{1}[\tau_i = T_i], \quad (\text{IX.10})$$

and the normalized rollout completion **Step** ratio as:

$$\text{Step} = \frac{1}{N} \sum_{i=1}^N \frac{\tau_i}{T_i}. \quad (\text{IX.11})$$

Thus, it measures the fraction of the reference trajectory completed before early termination, rather than the raw number of simulator steps.

We report **Reward** as the object-tracking reward ratio compared with the perfect reference trajectory. For trajectory i , the accumulated evaluation reward is $R_i = \sum_{t=1}^{\tau_i} r_{\text{eval}}^t$. Since a perfect trajectory obtains reward C at every timestep, its maximum accumulated reward is CT_i . We define

$$\text{Reward} = \frac{1}{N} \sum_{i=1}^N \frac{R_i}{CT_i}. \quad (\text{IX.12})$$

This metric jointly reflects tracking accuracy and rollout progress, because early termination reduces the accumulated reward.

Cost measures the average simulation steps required to generate one successful trajectory timestep. For successful trajectories, we define

$$\text{Cost} = \frac{\sum_i M_i}{\sum_i T_i}, \quad (\text{IX.13})$$

where M_i is the number of simulator steps used to generate trajectory i , and the summation is taken only over successful trajectories with $\tau_i = T_i$. Replay has the lowest cost because

it directly executes the retargeted reference trajectory, while MPC and RL require additional simulation rollouts for local search or policy refinement.

X. POLICY ARCHITECTURE AND TRAINING DETAILS

We train the policy solely on synthetic robot demonstration data generated by EgoEngine. No real-robot teleoperation data or co-training scheme is used in this stage.

A. Observation Encoder

Our policy is based on HPT [49], taking RGB observations and proprioceptive states as input, and directly outputs actions with a flow-matching head.

Visual stem. Given an RGB image $\mathbf{I} \in \mathbb{R}^{H \times W \times 3}$, we first apply ImageNet normalization and encode it with a ResNet-18 backbone truncated before global average pooling. This produces a spatial feature map, which is flattened and linearly projected to the transformer hidden dimension. A set of learnable query tokens attends to the visual features through a single cross-attention block, producing a fixed number of visual tokens.

Proprioceptive stem. The proprioceptive observation vector $\mathbf{q} \in \mathbb{R}^{d_q}$, including robot state information such as joint positions and end-effector pose, is first normalized and then projected with a linear layer. A set of learnable query tokens extracts a fixed number of proprioceptive tokens through a cross-attention block.

Token fusion encoder. Tokens from all observation stems are concatenated along the sequence dimension. We prepend a small set of learnable context tokens and process the resulting sequence with a transformer encoder. The output context tokens serve as the compact observation representation for action prediction.

B. Flow-Matching Action Decoder

We predict an action horizon of length T with a flow-matching transformer decoder conditioned on the encoder context tokens.

During training, given a ground-truth action sequence $a_1 = \mathbf{a}$ and Gaussian noise $a_0 \sim \mathcal{N}(0, I)$, we sample a continuous interpolation time $\tau \in (0, 1]$ and construct

$$x_\tau = \tau a_0 + (1 - \tau)a_1. \quad (\text{X.1})$$

The decoder takes x_τ together with a time embedding of τ as input and predicts the velocity field $v_\theta(x_\tau, \tau)$, which is regressed against the target velocity $\frac{dx_\tau}{d\tau} = a_0 - a_1$ that points from the clean action sequence toward the noise.

The decoder is implemented as a transformer with alternating self-attention and cross-attention blocks, where the action tokens attend to the encoder context tokens. A final linear layer maps decoder outputs to the action space.

At inference time, we initialize the action sequence from Gaussian noise at $\tau = 1$ and integrate the learned velocity field in the decreasing- τ direction from $\tau = 1$ to $\tau = 0$ using fixed-step Euler updates. We use 10 inference steps in all experiments.

C. Real Robot Experiments Setup

We evaluate EgoEngine on four real-robot manipulation tasks: Mustard, Drawer, Hammer, and Flower, as shown in Fig. X.1. Each task introduces randomized initial object poses to evaluate robustness under realistic execution variations.

Mustard. The Mustard task is a pick-and-place task. The robot needs to grasp a mustard bottle from its initial pose and place it onto the target plate. The initial position of the mustard bottle is randomized within a $10 \text{ cm} \times 10 \text{ cm}$ square region, with an orientation perturbation of $\pm 30^\circ$ around the vertical axis. The object is initially placed on the table surface. A trial is considered successful only if the mustard bottle is fully placed on the target plate and remains stable after release.

Drawer. The Drawer task requires the robot to first open the drawer and then place a cube inside it. The cube initialization follows the same randomization setting as the Mustard task, with a $10 \text{ cm} \times 10 \text{ cm}$ positional offset range and a $\pm 30^\circ$ orientation perturbation. A trial is considered successful only if the drawer is opened and the cube is placed inside the drawer.

Hammer. The Hammer task requires the robot to pick up a hammer from its holder and use it to strike a target nail. The nail position is randomized within a 5 cm offset range, and the initial hammer position is randomized with a lateral offset of up to 5 cm. A trial is considered successful only if the robot successfully lifts the hammer from the holder and strikes the target nail.

Flower. The Flower task requires the robot to pick up a water bottle and aim it toward a flower pot. The water bottle is randomized within a $10 \text{ cm} \times 10 \text{ cm}$ square region, with an orientation perturbation of $\pm 30^\circ$ around the vertical axis. The flower pot position is randomized within a 5 cm offset range, with arbitrary in-plane rotation. A trial is considered successful only if the robot picks up the water bottle and finally aligns it toward the flower pot.

XI. DATASET QUALITY ANALYSIS

A. Scalability and Sample Efficiency

We further study whether EgoEngine-generated demonstrations can improve sample efficiency compared with real robot demonstrations under limited data budgets. Specifically, we train downstream policies with increasing numbers of demonstrations from each data source and evaluate how quickly the success rate improves as more demonstrations are added. This experiment tests whether EgoEngine can provide effective supervision without requiring the same amount of real-robot teleoperation data.

We compare two training sources. **Real Robot** uses real teleoperation demonstrations only, and serves as the reference for how performance scales with additional robot data. **EgoEngine** uses robot demonstrations generated from egocentric human videos, evaluating whether the visual and action branches can produce training data that is useful for downstream policy learning. As shown in Fig. XI.1, EgoEngine is not uniformly better than real robot demonstrations, especially at larger data budgets where real teleoperation can provide

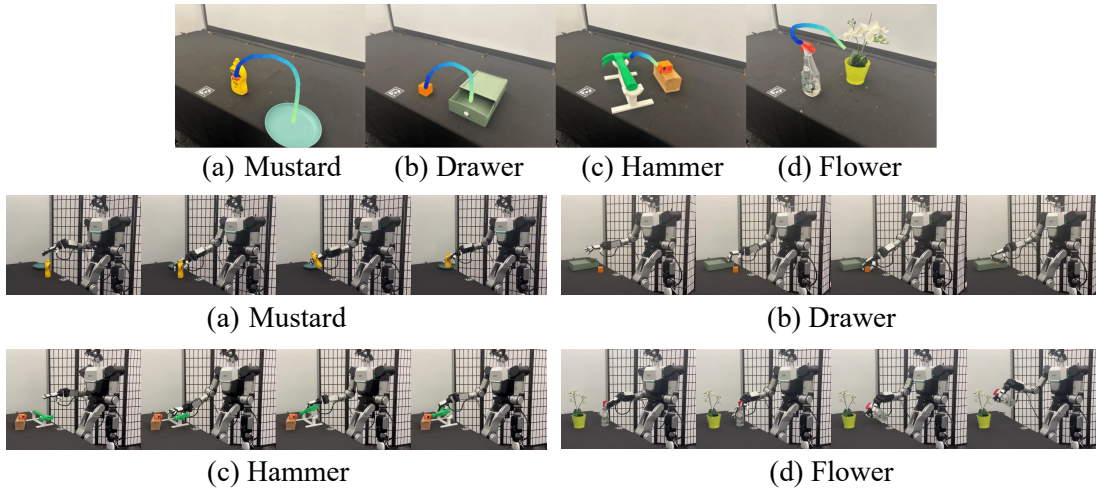


Fig. X.1: Real-robot task flow visualizations. (a) **Mustard**: pick and place the mustard bottle onto the target plate. (b) **Drawer**: open the drawer and place the cube inside. (c) **Hammer**: lift the hammer and strike the target nail. (d) **Flower**: grasp the water bottle and aim it toward the flower pot. Each task is evaluated with randomized object or target poses.

stronger task-specific supervision. However, EgoEngine still shows a clear scaling trend as the number of generated demonstrations increases, and reaches non-trivial performance with only a small number of demonstrations.

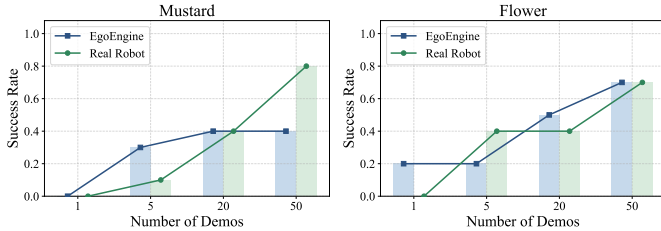


Fig. XI.1: Sample-efficiency comparison on Mustard and Flower under different demonstration budgets.

On Flower, EgoEngine approaches the real-robot trend at larger data budgets and achieves positive success even with one demonstration. On Mustard, EgoEngine improves steadily in the low-data regime, although real-robot teleoperation remains stronger at the largest budget. These results suggest that EgoEngine-generated demonstrations are not a replacement for high-quality real robot data in all cases, but provide a sample-efficient source of supervision that can reduce the dependence on collecting large numbers of real-robot demonstrations for each task.

B. Motion Pattern Analysis

Smoothness. We quantify action smoothness using Spectral Arc Length (SPARC) [13], a frequency-domain metric defined on the trajectory speed profile. Given a speed sequence $s_{1:T}$, SPARC maps the trajectory to a scalar smoothness score:

$$\text{SPARC}(s_{1:T}) \triangleq - \int_0^{\omega_c} \sqrt{\left(\frac{1}{\omega_c}\right)^2 + \left(\frac{d}{d\omega} \frac{S(\omega)}{S(0)}\right)^2} d\omega, \quad (\text{XI.1})$$

where $S(\omega)$ is the magnitude spectrum of $s_{1:T}$. The cutoff frequency ω_c is chosen adaptively using an amplitude threshold $\bar{S}$ and an upper bound $\omega_c^{\max}$. We use zero-padding with factor $K = 4$, $\omega_c^{\max} = 15$, and $\bar{S} = 0.05$. Higher values indicate smoother trajectories.

The results are reported in Tab. XI.1. EgoEngine datasets exhibit smoother action trajectories than real-robot demonstrations on average. This is expected because EgoEngine starts from human motions, which are naturally smoother than visual teleoperation trajectories, and further applies action-smoothness regularization during real-robot action generation. In contrast, visual teleoperation is affected by hand detection errors, sensing latency, and intermittent corrective motions, which can introduce high-frequency action changes. Thus, beyond task success, EgoEngine also improves the motion quality of generated demonstrations by combining smooth human action priors with explicit smoothness constraints.

Motion Pattern

Robustness. We qualitatively compare the grasping behaviors induced by teleoperation and EgoEngine across four tasks by visualizing the grasp stage of representative trajectories. The comparison reveals that EgoEngine is particularly effective when the task can benefit from smooth human-like motion priors and contact-aware correction, while real teleoperation

Task	SPARC $\uparrow$	
	Real	EgoEngine
Mustard	-8.68	-4.88
Drawer	-10.40	-7.49
Hammer	-3.21	-3.25
Flower	-4.66	-3.88
All	-6.60	-4.81

TABLE XI.1: Action quality comparison between real-robot demonstrations and EgoEngine datasets across four manipulation tasks, measured by trajectory smoothness.

when the task can benefit from smooth human-like motion priors and contact-aware correction, while real teleoperation

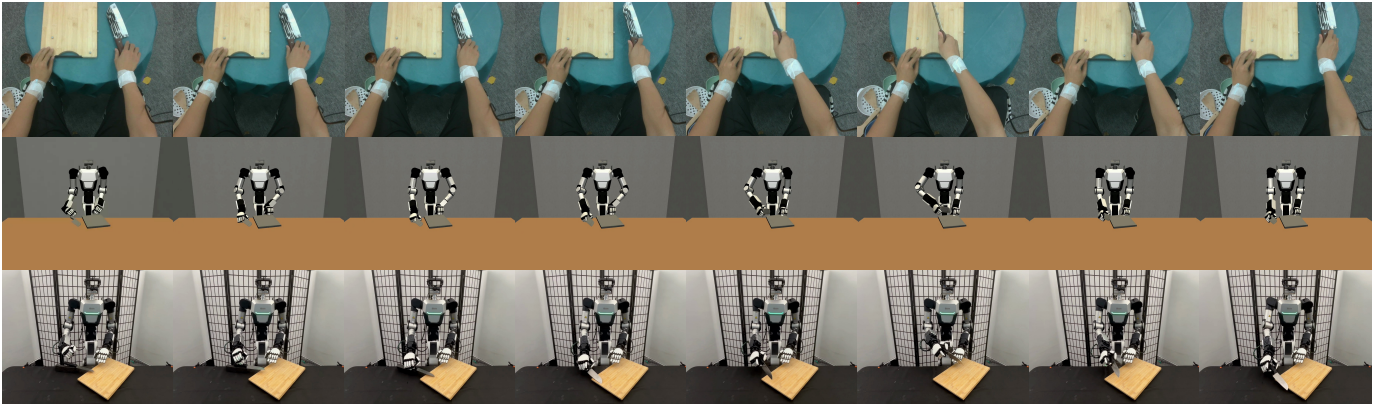


Fig. XI.2: Visualization of (Cut, Knife, Plate) task from TACO dataset. We show the original video, simulation, and real-robot rollout after action refinement. For safety, we add an offset for the real robot rollout.

remains stronger for some precise pinch-grasp cases. For tasks such as Mustard and Drawer, which require stable pinch grasps during object pickup, teleoperated demonstrations can still provide more reliable finger placement. In these cases, EgoEngine occasionally exhibits a small wrist-orientation offset at the pickup stage, which leads to unstable finger contact and less reliable grasps, even though the generated trajectories are robust in simulation under substantial domain randomization.

In contrast, for Flower and Hammer, which rely more on power grasping, EgoEngine performs comparably to or better than teleoperation. For Flower, EgoEngine produces grasps on the water bottle that are nearly as stable as those from teleoperated demonstrations. For Hammer, EgoEngine performs noticeably better, consistently adopting a more effective pre-grasp strategy by rotating the wrist toward the thumb side and opening the thumb and index fingers widely before contact, which leads to more stable hammer acquisition. These results suggest that EgoEngine is currently more reliable for manipulation patterns that benefit from smooth human motion and power-grasp structure, while precision pinch grasps remain a key failure mode.

Complex Tasks Beyond Visual Teleoperation. We further demonstrate that EgoEngine can use human videos to generate robot actions for complex manipulation tasks that are difficult to collect through visual teleoperation, as shown in Fig. XI.2. We take a long-horizon TACO video demonstration of cutting on a plate with a knife, reconstruct the task in simulation, and transfer the generated action trajectory back to the real robot with a similar knife and cutting board. This task is substantially more difficult than simple pick-and-place: the robot must grasp the knife handle, establish a stable tool grasp, perform in-hand reorientation, and maintain a useful cutting pose while interacting with the plate. Such behaviors are extremely difficult to collect through visual teleoperation because the operator does not directly observe the full robot hand state and must control a high-dimensional dexterous hand through noisy visual hand tracking, sensing latency, and limited real-time dexterity. As a result, precise tool acquisition

and in-hand adjustment, such as grasping the knife handle and reorienting it for cutting, are nearly infeasible to perform reliably through teleoperation. EgoEngine addresses this difficulty through a Real-to-Sim-to-Real pipeline: the human video provides a task-level motion prior, the action branch converts it into an executable robot trajectory in simulation, and the resulting trajectory is replayed on the real robot. The successful real-robot rollout shows that EgoEngine can transfer complex human tool-use behaviors into physically plausible and task-directed robot motions, highlighting an advantage over visual teleoperation in settings where direct robot data collection is prohibitively difficult.