Flash ALOHA: Learning Fast and Precise Manipulation

John Hua Yao Qi Wu Yihuai Gao Chelsea Finn Zipeng Fu
Stanford University
Project Website

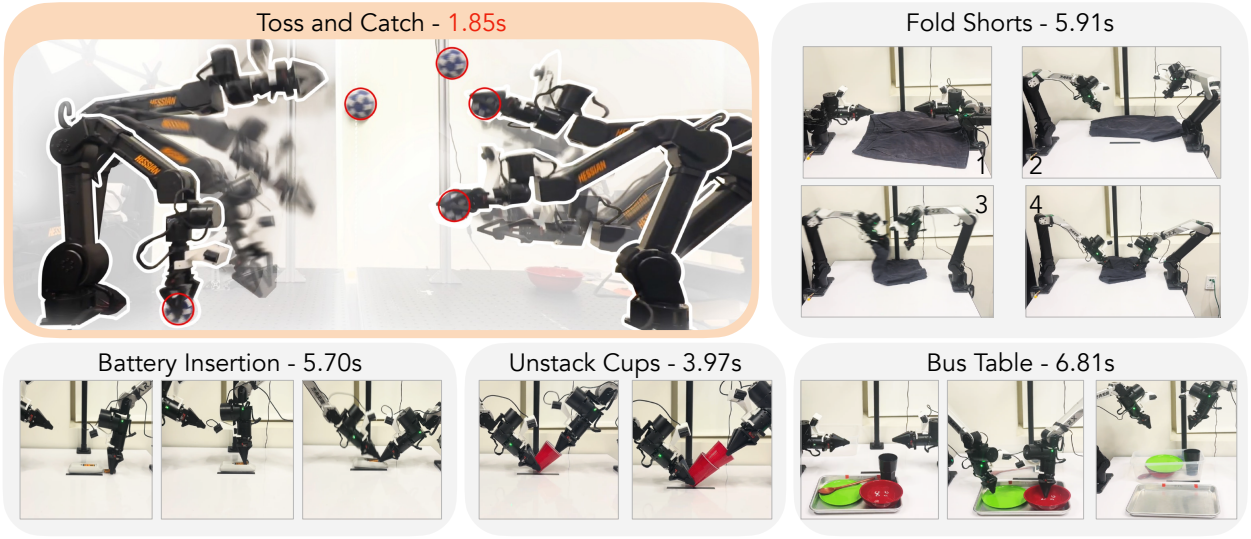


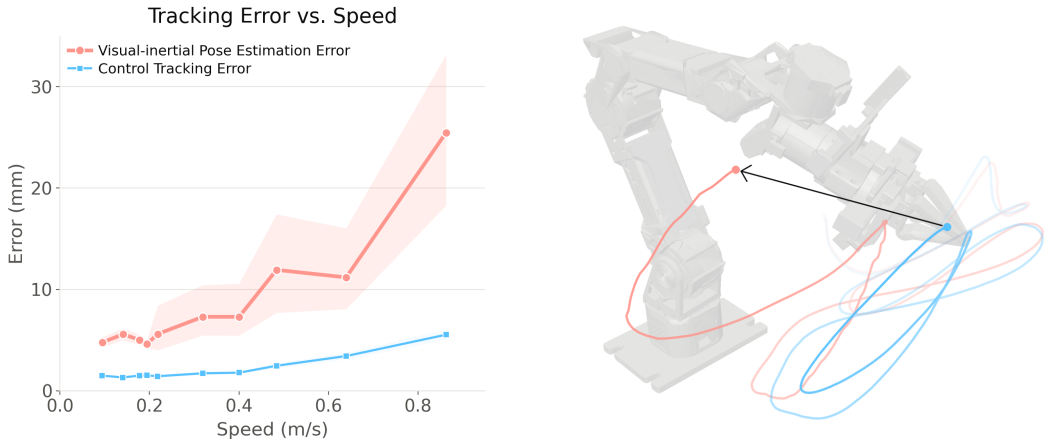
Fig. 1: ***Flash ALOHA***. We present a full-stack system for learning precise and high-speed manipulation policies, which can autonomously do bimanual toss and catch in under 2 seconds, bus a full dining set under 7, and fold a pair of shorts under 6.

***Abstract*— Learning from human demonstrations has enabled robots to acquire a wide range of manipulation skills, but learned policies typically execute far slower than ordinary humans. This speed gap is mainly due to lack of an interface for precisely collecting demonstration data at high speed, and the difficulty in training policies that can robustly execute high-speed motions. In this paper, we present Flash ALOHA, a practical system for learning fast and precise robotic manipulation. Our system uses kinesthetic teaching to intuitively collect accurate and near-human-speed demonstrations on a backdrivable bimanual platform, yielding natural and fast trajectories. Our learning pipeline consists of two components: a lightweight smoothing algorithm to address tracking instabilities produced by action chunking, and for visual alignment we mask out the operator visible in the camera views. Given 50 demonstrations for each task, Flash ALOHA autonomously completes bimanual tasks such as ball tossing and catching, folding shorts, battery insertion, and bussing tables at over 80% success rates while close to human speed.**

I. Introduction

Learning from human demonstrations has proven to be a powerful paradigm for teaching robots a wide array of complex skills. We have seen robots learn everything from fine-grained manipulation tasks such as threading a shoelace and lighting a match to mobile manipulation tasks such as sauteing and serving shrimp [1]–[3]. However, despite these successes, a significant gap remains between robotic and human performance: *speed*. While a person can perform many everyday chores with fluid and rapid motions, policies learned from demonstration typically execute with a slow and deliberate pace that feels distinctly non-human. This disparity is more than just an aesthetic issue; it is a fundamental barrier to deploying robots in practical human-centric environments where efficiency and responsiveness are important. For time-sensitive tasks like cooking and serving food, or collaborative tasks requiring synchronization with humans, the value of a robotic assistant diminishes greatly if it performs these tasks at a fraction of human speed.

The root of this speed gap lies in two challenges. First, there is a lack of accessible systems for collecting demonstration data that capture the dynamics of human-speed motion. Teleoperation interfaces based on puppeteering [1], 3D mice [4], or VR controllers [5] are not designed for high-speed interaction, due to latency, unintuitive control, and limited sensory feedback for human operators, causing heavy mental load for human operators. This forces human operators to move slowly and carefully, resulting in datasets of sluggish trajectories. The policy, in turn, learns to imitate this slow behavior. Handheld devices [6] offer the benefit of more intuitive data collection, and promise scalability by avoiding



(a) Error vs. end-effector speed.

(b) Visual-inertial drift during a fast free-space motion.

Fig. 2: ***Visual-Inertial Pose Estimation vs. Control Tracking.*** (a) Average error in 1 s windows over replayed action trajectories, binned by end-effector speed. (b) End effector trajectory of a recording test from the robot's encoders (blue) and a wrist-mounted visual-inertial tracker (red), a drift of over 10 cm. Per-task policy rollouts and setup details can be found in Appendix I, II.

demonstrations on real robots. However, such devices commonly rely on visual and inertial information for localization, and completely fail in pose estimation for high speed motions (Fig. 2b). State of the art visual-inertial tracking systems degrade quickly with speed, seen in Fig. 2a.

Second, even after assuming near perfect recording, learning a policy that can reliably execute these motions is a significant hurdle. Using recorded positions as actions as in handheld devices [6] assumes perfectly reaching every commanded position on time, requiring high torques at high speeds. Discontinuities from action chunking create catastrophic instability. Additionally, position data may not always include all necessary information for forceful manipulation, such as when pressing hard on near stationary objects. Visually, having the operator in view of the camera when directly moving the robot also can confuse the policy.

To address these issues, we introduce Flash ALOHA, a practical system for directly learning near human speed bimanual manipulation from human demonstrations. During data collection, the end effectors of lightweight robot arms are attached with two scissor handles for human operators to backdrive the arms under gravity and friction compensation. This method of direct physical guidance provides operators with rich real-time sensory feedback in vision, haptics, and sound, enabling high speed, precise, and contact rich data collection. The arms' motor encoders provide precise position readings even under aggressive motion, avoiding pose estimation issues. To ensure that position trajectories produce effective policies, operators at collection time may directly replay recorded trajectories on the arms between episodes, catching any tracking issues that may arise due to missing action data.

We present a learning pipeline for high-speed manipulation data. Our approach incorporates two key techniques. First, to ensure smooth execution, we introduce test-time smoothing, a post-hoc real-time processing for smooth transitions between action chunks. This method helps mitigate jerkiness at high speed that can arise from action chunking, allowing the robot to perform fast motions without sacrificing stability. Second, to address the visual domain shift from a human operator being present with hands holding the robot end-effectors, we employ a masking technique to align the visual domains between collection and deployment. By removing regions with human hands appearing in cameras via an analytical model, we align the visual input distributions. We find that while performance remains high without visual alignment, masking further improves performance.

The synergy between our intuitive hardware for near-human-speed data collection and our robust learning pipeline allows Flash ALOHA to learn fast, complex tasks from a small number of demonstrations. With only 50 demonstrations per task, our system learns to autonomously perform fine-grained tasks such as tossing a ball from one arm and catching it with the other, folding chino shorts, bussing a table, inserting a battery into a remote, and unstacking plastic cups, achieving over 80% success rates. Critically, these tasks are completed at speeds that are close to average human speeds, reducing the execution time by over 50% compared to that of teleoperation systems, and represent a significant step towards creating robots that can operate efficiently in the real world.

II. Related Work

Learning from Demonstrations. Learning from demonstrations, or imitation learning, is a cornerstone of modern robotics, enabling robots to acquire complex skills by mimicking expert demonstrations [7]. The efficacy of this paradigm is heavily dependent on the interface used for data collection. A wide spectrum of interfaces has been explored, ranging from processing human videos to various forms of interactive teleoperation. While human videos offer immense scale and diversity, they introduce significant embodiment and observation gaps that complicate direct policy learning [8]–[10]. Teleop methods aim to reduce this gap. Common low-cost interfaces include keyboards [11], 3D space mice [12], game controllers [13] and phones [14], though they can be unintuitive for controlling 6-DoF motion. More immersive interfaces, such as Virtual Reality controllers, provide more direct 6-DoF end-effector control but can suffer from latency, a lack of haptic feedback, and a lack of feedback on kinematic constraints [5], [15]–[17]. Other approaches shadow human motion in real time or offline using real-time pose estimation [18]–[21], exoskeletons [22], [23] and vision-based tracking [6], [24], [25], capturing natural kinematics but still facing the human-to-robot correspondence problem. To achieve a more direct mapping, leader-follower puppeteering systems provide secondary leader devices that the operator manipulates to control the follower robots providing intuitive joint-space control [1], [2], [26]–[30]. In contrast to these methods, Flash ALOHA utilizes kinesthetic teaching where human operators directly hold and guide the robot motion, eliminating the cost and complexity of a duplicate leader setup for puppeteering and bypassing the latency or embodiment gap inherent in other interfaces.

Kinesthetic Teaching. Kinesthetic teaching, where an operator physically guides a robot's end-effector through a task, is one of the most direct and intuitive methods for programming robot behaviors [7]. By moving the robot by hand, the demonstrator can implicitly convey not only the desired trajectory but also information about contact points and forces. This method requires the robot to be either passively compliant or actively backdrivable, often necessitating gravity compensation to make the robot feel "weightless" and reduce operator fatigue. Early work demonstrated the effectiveness of this approach for learning simple trajectories and dynamic movement primitives [31]–[37]. However, a primary challenge has been the difficulty of recording fast, dynamic motions due to robot inertia and motor friction, which can make the robot feel sluggish and heavy to the operator. Furthermore, the physical presence of the human operator in the workspace during demonstration can create a visual domain shift, as the policy is later deployed in a scene without the operator's hands and arms. Flash ALOHA addresses the challenge of collecting high-speed data by using a lightweight backdrivable bimanual platform with low-gear-ratio actuators with real-time gravity compensation along with an ergonomic scissor handle design, significantly reducing operator effort. Optional motor friction compensation and gripper gear upsizing further reduce operator fatigue and improve trajectory quality. Together with our masking and real-time smoothing, Flash ALOHA can capture and learn end-to-end policies for precise and fast skills from fluid, near-human-speed motions that are difficult to obtain and execute with kinesthetic teaching systems.

High-Speed Manipulation. Achieving human-level speed in manipulation is a long-standing goal in robotics, crucial for applications where efficiency is paramount. Traditional approaches to high-speed robotics often rely on precise analytical models of robot and object dynamics, using techniques like trajectory optimization to generate feasible, time-optimal paths. These methods have enabled impressive feats, such as industrial robots performing rapid pick-and-place operations or research platforms batting [38], playing table tennis [39]–[43], moving objects and avoiding collisions at speed [44], tossing [45], dynamically manipulating deformable objects [46]–[48], and catching objects in mid-air [49]. However, these approaches typically require extensive, task-specific engineering and can be brittle to unmodeled contact dynamics or perturbations. Learning-based methods offer a promising alternative by acquiring dynamic skills directly from data. However, a primary bottleneck has been the lack of systems capable of collecting demonstration data that captures near-human-speed motion [50], [51].

Furthermore, even with a suitable dataset, learning a policy to robustly execute fast motions is a significant hurdle [6], [52], [53]. Fast movements are less forgiving, where latency discrepancies can cause out-of-sync actions, and discontinuous commands from a policy can lead to instability.

Orthogonal to collecting high speed human data, reinforcement learning may also be used to collect high speed data while preserving effectiveness. Such approaches typically explicitly model speed as an objective [51], [52], [54]. Other speedup approaches also include using different action representations that provide analytical representations of trajectories for speedup [55]. Flash ALOHA is complementary to these approaches, by allowing operators to directly train high speed policies without refinement.

III. Hardware Setup & Data Collection

We set up our hardware with the consideration of using kinesthetic teaching for fast and precise data collection. We prioritize comfort for human operators, pursuing

maximal achievable speed using our hardware setup.

Shown in Fig. 3, Flash ALOHA consists of a bimanual setup of two different types of 6-DoF robot arms, HESSIAN and ARX, each with one parallel jaw gripper. We leverage off-the-shelf robot arms using planetary actuators, and human operators can back-drive the robot arms during kinesthetic teaching with relatively low internal friction in the joints, compared to others like harmonic actuators. We only use the HESSIAN setup for the highly dynamic Toss & Catch task since this task requires high joint velocity limits for high-speed trajectory following and very low joint friction for high-speed data collection.

We implement real-time gravity compensation based on the joint positions and masses of each robot link through torque control at 100Hz [56]. We retrofit the end effector of a robot arm with scissor handles for human operators to hold and backdrive the robots arms for completing manipulation tasks. One operator handle is attached on each jaw of the gripper. The smaller handle is for the thumb, and the other handle is for the other four fingers. Fingers are held in place with an adjustable strap, providing human operators with firm and intuitive grasps of the end effectors of the robot arms. The long finger handle allows operators to pull downwards and sideways, while hand contact with the bottom of the arm assists with upwards acceleration. The whole setup consists of three RGB cameras capturing images at 50Hz, with one wrist camera mounted on each arm facing towards the region between the gripper jaws, and one top-down camera capturing the workspace from above.

During data collection, we record three 360 x 640 RGB images I_1, I_2, I_3 from the three cameras, joint positions q, and joint velocities $\dot{q}$ at 50Hz. Through kinesthetic teaching where the human operators directly hold and guide the robots to do different tasks, human operators obtain various real-time sensory feedbacks, including 3D vision, haptics through the handles, and contact sound without any delays that exist in common teleoperation platforms.

IV. Data Processing, Policy Learning & Deployment

Given a dataset of kinesthetic teaching data for fast and precise manipulation, directly applying imitation learning algorithms fail for two reasons. First, a human operator is present with hands holding the robot end-effectors during data collection, while the robots are alone during the autonomous execution phase after deployment. This creates a visual distribution shift between data collection and deployment. Second, fast movements are less forgiving, as small errors in execution can compound into unrecoverable failures and non-smooth actions can cause instability. The high-speed nature of the manipulation data collected presents a hurdle for learning a reliable policy.

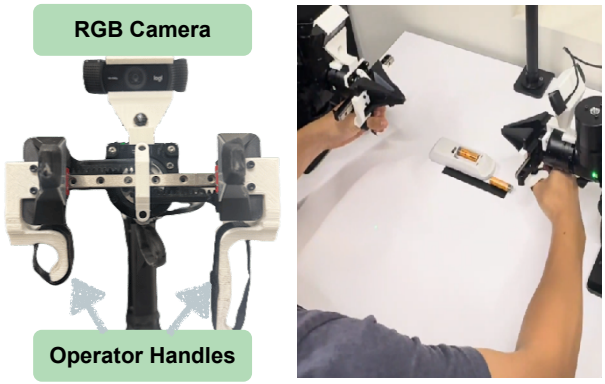


Fig. 3: ***Hardware Setup.*** Our printed handles allow for easy operation and data collection.

To address the distribution shift between data collection and deployment, we employ a masking technique to align the visual domains for training and testing, illustrated in Fig. 4. By removing regions with human hands and arms appearing in cameras using an analytical model, we align the visual input distributions, preventing the policy from being distracted by the presence or absence of the human operator. And to ensure smooth execution of fast manipulation, we introduce test-time smoothing, a post-hoc real-time processing for smooth transitions between action chunks. Illustrated in Fig. 5, this method helps mitigate the jerkiness at high speed that can arise from action chunking [3], allowing the robot to perform fast motions without sacrificing stability.

Masking in Image Space. During training, we pre-process the collected data, aiming to mask out the appearances of hands and arms of the human operators. Given that the joint positions of both arms and robot description files are known, we can compute the poses of end-effectors of both robot arms in their respective base frames using forward kinematics (FK). We also know the transformation between the base frames of both arms through camera calibration. Thus, we can transform the end-effector pose of one arm into the base frame of the other arm. Since the relative pose of the handles for the human hands is fixed with respect to the end-effector pose, we deterministically calculate the center pose of the left hand in the right camera, and similarly for the right hand in the left camera, and mask out a small square region perpendicular to the end-effector pointing direction near the handles by setting the image RGB values for that region to be zero. For the top camera, we similarly mask out two rectangular regions for both hands and arms based on the end-effector poses. We provide an illustration for masking in Fig. 4. During deployment, we perform masking in real time for every

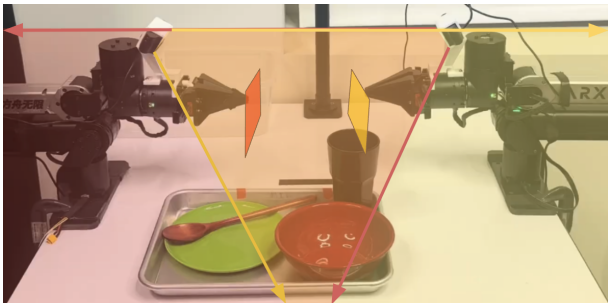


Fig. 4: ***Masking.*** Our masking method adjusts in real time based on the position of the end effectors of both arms.

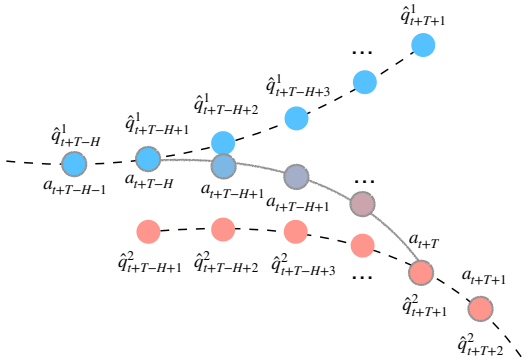


Fig. 5: ***Test-Time Smoothing.*** Given a previous chunk of target joint positions $\hat{q}^1$ and a later chunk $\hat{q}^2$, we apply smoothing to obtain the final commanded actions a.

frame. Formally, at each time step t, we perform the masking operation M and get three processed images $I'_{1,t}, I'_{2,t}, I'_{3,t}$, where

$$I'_{1,t}, I'_{2,t}, I'_{3,t} = M(I_{1,t}, I_{2,t}, I_{3,t}, q_t, \text{FK}).$$

Policy Learning. Unlike teleoperation where the action commands for the robots are explicitly sent through devices like leader arms or VR controllers, we only capture the visual and proprioceptive readings from the robots. We aim to train policies that can predict future joint positions which the robots should achieve given current images and proprioception. We follow the recent progress in imitation learning by using action chunking [1], which mitigates compounding error problems that is common in behavior cloning methods and improves overall performance. Given a policy π and an action chunk size of T, we train the policy by maximizing the probability

$$\pi([q_{t+1}, ..., q_{t+T}]|I'_{1,t}, I'_{2,t}, I'_{3,t}, q_t),$$

where the predicted future joint positions will serve as the target joint positions for a low-level PD controller to execute. We additionally tune an action lookahead where we skip the first n actions at every chunk boundary, resulting from action delay at execution. In this paper, we use ACT [1] as the policy representation, but we expect other policy representations for imitation learning with action chunks would also work.

Test-Time Smoothing. When moving at increased speeds, we observe that discontinuities between action chunks [3] become larger and more visible, affecting policy performance. We introduce a simple post-hoc test-time smoothing technique. Illustrated in Fig. 5, we tackle the discontinuity between two adjacent chunks by blending the last few actions of a chunk with the first few actions of the later chunk where these two action chunks have a small overlap in corresponding timesteps. For a smoothing window size of H smaller than the action chunk size T, an action chunk $[\hat{q}^1_{t+1}, ..., \hat{q}^1_{t+T}]$ is first predicted at timestep t, and another one $[\hat{q}^2_{t+T-H+1}, ..., \hat{q}^2_{t+2T-H}]$ is later predicted at timestep $t+T-H$. At timestep t', the action command $a_{t'}$ is the raw predicted target joint position $\hat{q}_{t'+1}$ at regions where there is no chunk overlap. Within the overlapping regions, we weigh the two corresponding target joint positions by using a sigmoid coefficient $w(t') = \text{sigmoid}\big((t' - (t+T-H) - H/2)/\sigma\big)$ where σ is a time scale, and the final blended action command is

$$a_{t'} = (1 - w(t')) * \hat{q}^1_{t'+1} + w(t') * \hat{q}^2_{t'+1}.$$

By adjusting the smoothing horizon and time scale of the sigmoid coefficient, we can adjust how sharply and reactively the robot will change to follow a later action chunk from a previous obsolete action chunk.

V. Experiments

In our experiments, we aim to determine the extent to which our masking strategy as well as our smoothing method benefit the overall success rate of policies. Additionally, we study the effects of several comparative approaches in these two domains. Through user trials, we verify the intuitive understanding that our directly operated data collection platform enables faster training of human operators. All main tasks are shown in Fig. 1.

A. Tasks

In ***Unstack Cups***, the left arm reaches out and grasps the bottom of two stacked red solo cups. Then, the right arm subsequently grasps the lip of the top cup without contacting the bottom, and pulls. The grippers must carefully balance their grasps; if the left grips too hard, the right can't tilt and grab. If the right over-grips, it pulls the entire stack. The cups' initial starting positions are randomized along a fifteen centimeter piece of tape. This task is collected with a timer of four seconds.

Battery Insertion. During execution, one arm picks up a battery and places it at an oblique angle into a remote control, taking care not to touch the spring. It then pushes one end of the battery into the spring. The other gripper then raises and forcefully pushes down

Masking																		
	Unstack Cups				Battery Insertion						Bus Table				Fold Shorts			
Method	Grab	Pull	Whole	Time(s)	Pick	Place	Prepare	Push	Whole	Time(s)	C&S	B&P	Whole	Time(s)	Up	Right	Whole	Time(s)
No Mask	90	**90**	**90**	4.12	95	95	85	75	75	6.44	**100**	**100**	**100**	6.97	**100**	90	90	6.19
Inpainting	**100**	85	85	**3.92**	95	95	70	65	65	**5.65**	**100**	**100**	**100**	**6.57**	**100**	80	80	6.20
Ours	**100**	**90**	**90**	3.97	**100**	**100**	**90**	**85**	**85**	5.70	**100**	**100**	**100**	6.81	**100**	**100**	**100**	**5.94**

TABLE I: ***Our masking improves performance.*** Our masking strategy almost uniformly beats non-masked and diffusion inpainted in policies, enabling tasks such as battery insertion.

on the battery, slotting it into place. The battery and the remote are randomized along the tape. This task is collected in under seven seconds.

Bus Table involves removing common dining ware from a pan and placing them into a tray. An arm first grasps a nearby tray and brings it closer to the pan. Then, both arms engage, alternating picking up and placing four items from the pan into the tray: a cup, a bowl, a spoon, and a plate. The position of the pan is randomized along the tape. This task is collected with a timer of six seconds.

Fold Shorts involves folding a pair of shorts placed flat. The arms must reach and grasp the edge of the bottom trunk of the shorts and toss the trunk over from bottom to top flush against the top cover. Then, the left gripper must grasp the edge of the shorts along the waist, and toss it from left to right while the right arm creates a crease. The shorts are randomly rotated by up to thirty degrees in their initial position, and this task is collected against a timer of five seconds.

Toss & Catch. The left arm first approaches and picks up a small soft ball on the table. Then it swings forward and up dynamically in high speed while releasing the ball. The right arm approaches the tossed ball-in-air swiftly with closing grippers to catch the ball. The task is only two seconds long.

B. Comparisons

We compare our method to *Inpainting*, *No Mask*, *Temporal Ensembling* (TE), *No Smooth*, *Puppeteering*, *Puppet Double Freq*, and *Puppet Skip Actions*.

Inpainting [57]. A policy trained on diffusion inpainted video, where the operator hands are segmented and replaced with the background by a pretrained segmentation [58] and pretrained diffusion network [59].

No Mask. A policy trained with the raw video.

Temporal Ensembling [1]. A post-process smoothing method where an action chunk is predicted every timestep, and exponentially weighted and averaged with past action chunks.

No Smooth. No smoothing method applied.

Puppeteering [1]. The operator collects and deploys policies on a teleoperated setup.

Puppet Double Freq. On the puppeteering setup, the policy is deployed with double the controller frequency.

Puppet Skip Actions. On the puppeteering setup, the policy acts on every other predicted action per chunk.

C. Masking Aligns Data Collection and Deployment

We evaluate the efficacy of masking, which aligns the visual distribution of the policy between collection and deployment time. Table I details the results of our experimental trials comparing policies trained with masking, no masking, and on diffusion inpainted video. We note an increase in performance on most tasks with masking compared to simply training with hands included and inpainted video.

The policies trained on inpainted video suffer significant performance losses compared to the base video. We hypothesize that this is due to two factors. First, as can be seen in Fig. 6, the shadows of the arms are still visible against the reflective surface of the table. Another factor is the noisy diffusing of frames due to lack of context. These visual artifacts not visible at test time may confuse the policy. Given a more controlled environment, this method's performance may improve.

D. Test-Time Smoothing Improves Performance

In Table II, we see performance comparisons between non-smoothed, test-time smoothed, and temporal ensembling-smoothed trajectories. In all tasks, test-time smoothing achieves the best performance, and by a wide margin in tasks ***Unstack Cups*** and ***Battery Insertion***. The resulting instabilities in the high speed non-smoothed policy rollouts result in imprecise grasps and pushes, and while the temporal-ensembling smoothed policies have mixed results, their execution speeds drop in half dramatically from collection time. This is due to the increased processing time necessary to compute inference every timestep.

A failure mode of temporal ensembling is its tendency to choose an averaged path when forced to change course [3]. We hypothesize that, at higher speeds, the tracking of robot controllers often become less reliable which forces policies to react and frequently change course to adjust for error. Thus, even if temporal ensembling were run on faster hardware and had no inference delay, it would perform even more poorly at speed due to this phenomenon. This also partially explains the presence of discontinuities in the first place, even in relatively

Method	Smoothing																	
	Unstack Cups				Battery Insertion						Bus Table				Fold Shorts			
	Grab	Pull	Whole	Time(s)	Pick	Place	Prepare	Push	Whole	Time(s)	C&S	B&P	Whole	Time(s)	Up	Right	Whole	Time(s)
No Smooth	**100**	65	65	4.23	**100**	85	75	75	75	**5.29**	**100**	80	80	**6.70**	95	80	80	6.16
TE	95	35	35	7.75	**100**	**100**	85	55	55	10.18	**100**	**100**	**100**	12.23	**100**	90	90	11.60
Ours	**100**	**90**	**90**	**3.97**	**100**	**100**	**90**	**85**	**85**	5.70	**100**	**100**	**100**	6.81	**100**	**100**	**100**	**5.94**

TABLE II: ***Smoothing improves performance.*** Test time smoothing outperforms or matches other methods in success rate in every category.

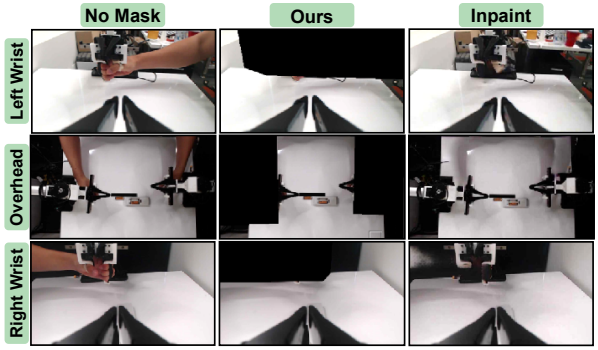


Fig. 6: **Masking Comparisons.** We show the same initial starting frame of ***Battery Insertion***. Our masking method completely aligns the visual distributions of training and test times. Inpainting via diffusion creates unwanted artifacts.

uni-modal data with little strategy-switching.

E. Speed Comparisons

In this experiment, we investigate the effectiveness of training a policy on slow data, collected with puppeteering on an ALOHA 2 (setup in Appendix III). We artificially increase the policy execution speed in two different ways across three different tasks: ***Bus Table***, ***Unstack Cups***, and ***Battery Insertion***. We are unable to complete the ***Toss & Catch*** task with puppeteering due to the difficulty of catching a dynamic object through teleoperation. The results are visible in Table III, where we see sharp drops in performance after speeding up data collected through puppeteering to match the speed of ours. Refer to Appendix IV for chunk boundary jump details.

Puppet Double Freq simply doubles the controller frequency at execution time. This results in policies that appear out of control, with large jumps at chunk boundaries. With actions that jump larger distances, such as in ***Bus Table***, this effect becomes more noticeable. Despite still moving slower than our method, double-timed policies from puppeteering have significantly larger discontinuities than in our method. We hypothesize that these discontinuities lead to deviations between collection and test time, further worsening performance.

Puppet Skip Actions is running the controller frequency at the same rate, but skipping every other action. This method introduces less discontinuity, but creates new failure modes such as oscillating above the battery instead of pushing it in, and missing grasps. This is due to the policy oftentimes skipping critical steps, which also reduces overall performance.

F. User Study

We conduct a user study with seven volunteers with little to no teleoperation experience, having them operate our setup and puppeteering. For the study, we choose ***Battery Insertion***, a task requiring precision.

In the study, in both setups, users spend three minutes familiarizing themselves with the objects in the scene and the robotic arms. Next, they are shown an expert demonstration and given basic tips on operation procedures. Finally, they are timed in ten trials, attempting to insert the battery into the remote as quickly as possible.

We observe in Fig. 7 an ability for users to quickly learn how to operate with our hardware and at speeds up to three times as fast as puppeteering. This primarily results from the operator sitting closer to the environment, giving more direct visual feedback, and haptic sensations from operating the actual robot arms. This results in more natural data collection, even for beginners.

VI. Conclusion

In this paper, we present Flash ALOHA, a full-stack system for learning fast and precise manipulation. Our system presents a hardware setup for collecting high-speed demonstration, and a learning pipeline for smooth autonomous execution of fast and precise movements. Currently, the masking strategy that we use in the paper over-conservatively masks regions near human hands and arms, erasing potentially useful visual cues near human hands in the image space. Additionally, the whole system incorporates vision and proprioception as only two modalities, preventing the system from learning dexterous manipulation requiring contact-rich perception and control. In the future we would like to research adaptive masking strategies and including tactile sensing for better performance.

Acknowledgments

We'd like to thank Ajay Sridhar, Yanjiang Guo, Brian Kim, Lars Ankile, Marcel Torne, Aneesh Muppidi, Ke Wang, Yuejiang Liu, Emily Jin, and Moo Jin Kim for helpful insights, discussions, and help filming. This

Speed Comparisons																		
	Unstack Cups				Battery Insertion						Bus Table				Toss & Catch			
Method	Grab	Pull	Whole	Time(s)	Pick	Place	Prep	Push	Whole	Time(s)	C&S	B&P	Whole	Time(s)	Pick & Toss	Catch	Whole	Time(s)
Puppeteering	85	75	75	8.84	**100**	90	60	55	55	17.87	**100**	90	90	16.05	–	–	–	–
Puppet Double Freq	80	65	65	5.47	80	65	35	15	15	12.90	95	85	85	9.30	–	–	–	–
Puppet Skip Actions	85	50	50	5.58	95	95	55	30	30	10.30	95	85	85	9.29	–	–	–	–
Ours	**100**	**90**	**90**	**3.97**	**100**	**100**	**90**	**85**	**85**	**5.70**	**100**	**100**	**100**	**6.81**	**100**	**80**	**80**	**1.85**

TABLE III: ***Speed Comparisons.*** We note a sharp drop in performance on precise tasks when applying either speed-up approach.

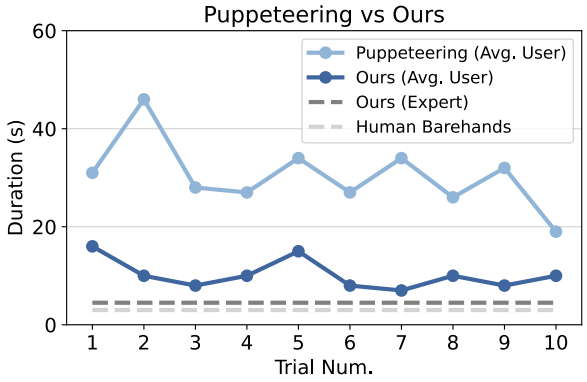


Fig. 7: ***User Study.*** Results on ***Battery Insertion*** for timing operators on both Ours and Puppeteering demonstrate a 50%+ gap in duration for both experienced and new operators.

work was supported by the Robotics and AI Institute. Zipeng Fu was supported by Pierre and Christine Lamond Fellowship. We appreciate the help from HESSIAN and ARX.

REFERENCES

[1] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," *RSS*, 2023.

[2] Z. Fu, T. Z. Zhao, and C. Finn, "Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation," in *CoRL*, 2024.

[3] K. Black, M. Y. Galliker, and S. Levine, "Real-time execution of action chunking flow policies," *arXiv preprint arXiv:2506.07339*, 2025.

[4] Y. Zhu, J. Wong, A. Mandlekar, and R. Martín-Martín, "robosuite: A modular simulation framework and benchmark for robot learning," *arXiv preprint arXiv:2009.12293*, 2020.

[5] 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," in *2018 ICRA*, 2018.

[6] 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," in *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[7] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, "Robot programming by demonstration," in *Springer handbook of robotics*. Springer, 2008, pp. 1371–1394.

[8] S. Kareer, D. Patel, R. Punamiya, P. Mathur, S. Cheng, C. Wang, J. Hoffman, and D. Xu, "Egomimic: Scaling imitation learning via egocentric video," in *2025 ICRA*, 2025.

[9] M. Lepert, J. Fang, and J. Bohg, "Phantom: Training robots without robots using only human videos," *arXiv preprint arXiv:2503.00779*, 2025.

[10] L. Y. Zhu, P. Kuppili, R. Punamiya, P. Aphiwetsa, D. Patel, S. Kareer, S. Ha, and D. Xu, "Emma: Scaling mobile manipulation via egocentric human data," *arXiv preprint arXiv:2509.04443*, 2025.

[11] H. Xiong, R. Mendonca, K. Shaw, and D. Pathak, "Adaptive mobile manipulation for articulated objects in the open world," *arXiv preprint arXiv:2401.14403*, 2024.

[12] Y. Zhu, A. Joshi, P. Stone, and Y. Zhu, "Viola: Imitation learning for vision-based manipulation with object proposal priors," in *CoRL*. PMLR, 2023, pp. 1199–1210.

[13] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert, "Learning monocular reactive uav control in cluttered natural environments," in *2013 ICRA*, 2013.

[14] A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay *et al.*, "Roboturk: A crowdsourcing platform for robotic skill learning through imitation," in *CoRL*. PMLR, 2018, pp. 879–893.

[15] X. Cheng, J. Li, S. Yang, G. Yang, and X. Wang, "Open-television: Teleoperation with immersive active visual feedback," *arXiv preprint arXiv:2407.01512*, 2024.

[16] Y. Park, J. S. Bhatia, L. Ankile, and P. Agrawal, "Dexhub and dart: Towards internet scale robot data collection," *arXiv preprint arXiv:2411.02214*, 2024.

[17] M. Seo, S. Han, K. Sim, S. H. Bang, C. Gonzalez, L. Sentis, and Y. Zhu, "Deep imitation learning for humanoid loco-manipulation through human teleoperation," in *2023 Humanoids*, 2023.

[18] A. Handa, K. Van Wyk, W. Yang, J. Liang, Y.-W. Chao, Q. Wan, S. Birchfield, N. Ratliff, and D. Fox, "Dexpilot: Vision-based teleoperation of dexterous robotic hand-arm system," in *2020 ICRA*, 2020.

[19] A. Sivakumar, K. Shaw, and D. Pathak, "Robotic telekinesis: Learning a robotic hand imitator by watching humans on youtube," *arXiv preprint arXiv:2202.10448*, 2022.

[20] Z. Fu, Q. Zhao, Q. Wu, G. Wetzstein, and C. Finn, "Humanplus: Humanoid shadowing and imitation from humans," in *CoRL*, 2024.

[21] T. He, Z. Luo, W. Xiao, C. Zhang, K. Kitani, C. Liu, and G. Shi, "Learning human-to-humanoid real-time whole-body teleoperation," *arXiv preprint arXiv:2403.04436*, 2024.

[22] H. Fang, H.-S. Fang, Y. Wang, J. Ren, J. Chen, R. Zhang, W. Wang, and C. Lu, "Airexo: Low-cost exoskeletons for learning whole-arm manipulation in the wild," in *2024 ICRA*, 2024.

[23] A. Purushottam, C. Xu, Y. Jung, and J. Ramos, "Dynamic mobile manipulation via whole-body bilateral teleoperation of a wheeled humanoid," *IEEE Robotics and Automation Letters*, 2023.

[24] 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.

[25] H. Etukuru, N. Naka, Z. Hu, S. Lee, J. Mehu, A. Edsinger, C. Paxton, S. Chintala, L. Pinto, and N. M. M. Shafiullah, "Robot utility models: General policies for zero-shot deployment in new environments," in *2025 ICRA*, 2025.

[26] R. Jenness and C. Wicker, "Master–slave manipulators and remote maintenance at the oak ridge national laboratory," Oak Ridge National Lab., Tenn.(USA), Tech. Rep., 1975.

[27] 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 IROS*, 2024.

[28] T. Buamanee, M. Kobayashi, Y. Uranishi, and H. Takemura, "Bi-act: Bilateral control-based imitation learning via action chunking with transformer," in *2024 AIM*, 2024.

[29] H. Ito, K. Yamamoto, H. Mori, and T. Ogata, "Efficient multitask learning with an embodied predictive model for door opening and entry with whole-body control," *Science Robotics*, 2022.

[30] Y. Jiang, R. Zhang, J. Wong, C. Wang, Y. Ze, H. Yin, C. Gokmen, S. Song, J. Wu, and L. Fei-Fei, "Behavior robot suite: Streamlining real-world whole-body manipulation for everyday household activities," *arXiv preprint arXiv:2503.05652*, 2025.

[31] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *International conference on machine learning*, 2016.

[32] A. Xie, F. Ebert, S. Levine, and C. Finn, "Improvisation through physical understanding: Using novel objects as tools with visual foresight," *arXiv preprint arXiv:1904.05538*, 2019.

[33] B. Akgun, M. Cakmak, J. W. Yoo, and A. L. Thomaz, "Trajectories and keyframes for kinesthetic teaching: A human-robot interaction perspective," in *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, 2012.

[34] P. Kormushev, D. N. Nenchev, S. Calinon, and D. G. Caldwell, "Upper-body kinesthetic teaching of a free-standing humanoid robot," in *2011 ICRA*, 2011.

[35] P. Kormushev, S. Calinon, and D. G. Caldwell, "Imitation learning of positional and force skills demonstrated via kinesthetic teaching and haptic input," *Advanced Robotics*, 2011.

[36] A. Weiss, J. Igelsbock, S. Calinon, A. Billard, and M. Tscheligi, "Teaching a humanoid: A user study on learning by demonstration with hoap-3," in *RO-MAN 2009-The 18th IEEE International Symposium on Robot and Human Interactive Communication*. IEEE, 2009.

[37] H. Ben Amor, E. Berger, D. Vogt, and B. Jung, "Kinesthetic bootstrapping: Teaching motor skills to humanoid robots through physical interaction," in *Annual conference on artificial intelligence*. Springer, 2009.

[38] T. Senoo, A. Namiki, and M. Ishikawa, "Ball control in high-speed batting motion using hybrid trajectory generator," in *Proceedings 2006 ICRA.*, 2006.

[39] D. B. DAmbrosio, S. Abeyruwan, L. Graesser, A. Iscen, H. B. Amor, A. Bewley, B. J. Reed, K. Reymann, L. Takayama, Y. Tassa *et al.*, "Achieving human level competitive robot table tennis," in *2025 ICRA*, 2025.

[40] D. Büchler, S. Guist, R. Calandra, V. Berenz, B. Schölkopf, and J. Peters, "Learning to play table tennis from scratch using muscular robots," *IEEE Transactions on Robotics*, 2022.

[41] K. Mülling, J. Kober, O. Kroemer, and J. Peters, "Learning to select and generalize striking movements in robot table tennis," *The International Journal of Robotics Research*, 2013.

[42] Y. Zhu, Y. Zhao, L. Jin, J. Wu, and R. Xiong, "Towards high level skill learning: Learn to return table tennis ball using monte-carlo based policy gradient method," in *2018 IEEE international conference on real-time computing and robotics (RCAR)*, 2018.

[43] C. Liu, Y. Hayakawa, and A. Nakashima, "Racket control for a table tennis robot to return a ball," *SICE Journal of Control, Measurement, and System Integration*, 2013.

[44] J. Ichnowski, Y. Avigal, Y. Liu, and K. Goldberg, "Gomp-fit: Grasp-optimized motion planning for fast inertial transport," in *2022 iICRA*. IEEE, 2022, pp. 5255–5261.

[45] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, "Tossingbot: Learning to throw arbitrary objects with residual physics," *IEEE Transactions on Robotics*, 2020.

[46] H. Zhang, J. Ichnowski, D. Seita, J. Wang, H. Huang, and K. Goldberg, "Robots of the lost arc: Self-supervised learning to dynamically manipulate fixed-endpoint cables," in *2021 ICRA*. IEEE, 2021, pp. 4560–4567.

[47] C. Chi, B. Burchfiel, E. Cousineau, S. Feng, and S. Song, "Iterative residual policy: for goal-conditioned dynamic manipulation of deformable objects," *The International Journal of Robotics Research*, 2024.

[48] H. Ha and S. Song, "Flingbot: The unreasonable effectiveness of dynamic manipulation for cloth unfolding," in *CoRL*, 2022.

[49] S. Kim, A. Shukla, and A. Billard, "Catching objects in flight," *IEEE Transactions on Robotics*, 2014.

[50] N. Masuya, H. Sato, K. Yamane, T. Kusume, S. Sakaino, and T. Tsuji, "Variable-speed teaching–playback as real-world data augmentation for imitation learning," *Advanced Robotics*, 2025.

[51] L. Guo, Z. Xue, Z. Xu, and H. Xu, "Demospeedup: Accelerating visuomotor policies via entropy-guided demonstration acceleration," *arXiv preprint arXiv:2506.05064*, 2025.

[52] N. R. Arachchige, Z. Chen, W. Jung, W. C. Shin, R. Bansal, P. Barroso, Y. H. He, Y. C. Lin, B. Joffe, S. Kousik *et al.*, "Sail: Faster-than-demonstration execution of imitation learning policies," *arXiv preprint arXiv:2506.11948*, 2025.

[53] D. D. Yuan, T. Z. Zhao, K. Burns, and C. Finn, "Speedtuning: Speeding up policy execution with lightweight reinforcement learning," in *2025 ICRA*, 2025.

[54] B. Kim, J. Pahk, C. Lee, J. Kim, J. Lee, T. T. Kim, K. Shim, J. K. Lee, and B.-T. Zhang, "Espada: Execution speedup via semantics aware demonstration data downsampling for imitation learning," *IEEE Robotics and Automation Letters*, 2026.

[55] X. Han, H. Xiong, H. Chen, C. Liu, A. Torralba, Y. Zhu, and Y. Du, "B-spline policy: Accelerating manipulation policies via b-spline action representations," *arXiv preprint arXiv:2607.09648*, 2026.

[56] K. M. Lynch and F. C. Park, *Modern robotics*. Cambridge University Press, 2017.

[57] M. Xu, H. Zhang, Y. Hou, Z. Xu, L. Fan, M. Veloso, and S. Song, "Dexumi: Using human hand as the universal manipulation interface for dexterous manipulation," *arXiv preprint arXiv:2505.21864*, 2025.

[58] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson, E. Mintun, J. Pan, K. V. Alwala, N. Carion, C.-Y. Wu, R. Girshick, P. Dollár, and C. Feichtenhofer, "Sam 2: Segment anything in images and videos," *arXiv preprint arXiv:2408.00714*, 2024.

[59] S. Zhou, C. Li, K. C. Chan, and C. C. Loy, "ProPainter: Improving propagation and transformer for video inpainting," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2023.

APPENDIX I
TRACKING ERROR DURING POLICY ROLLOUTS

Fig. 9 extends Fig. 2a with the commanded vs. achieved end-effector error of our arms during policy rollouts of each task. Fig. 10 shows the visual-inertial tracking setup.

APPENDIX II
VISUAL-INERTIAL TRACKING EXPERIMENT

Hardware. We mount an iPhone 15 Pro running the iPhUMI app on the wrist of a 6-DoF Seeed Studio reBot B601-RS arm with RobStride actuators, replacing the arm's last link with the iPhUMI phone holder and hollow gripper fingers (Fig. 10). iPhUMI records the phone camera's ARKit visual-inertial pose at about 60 Hz. It cannot stream poses live, so each recording is exported from the phone and aligned with the robot data offline. The phone stays seated in the holder throughout, and the transform X from the end effector to the phone camera is taken from the holder's CAD model.

Ground truth. The arm records its joint encoders, and forward kinematics gives the end-effector pose A_i at each sample. ARKit's world frame is gravity-aligned, but its origin and heading are set wherever tracking starts, so it differs between recordings. For each take we fit one rigid transform Y from the ARKit world frame to the robot base by least squares over the whole take, and compare A_i with the tracker's estimate of the end-effector pose

$$\hat{A}_i = Y B_i X^{-1},$$

where B_i is the ARKit camera pose.

Time alignment. The host and the phone are NTP-synchronized, and robot timestamps are mapped to Unix time through clock anchors recorded during each session. The remaining offset between the two streams, including pose latency, is found per take by cross-correlating the end effector's angular speed from both sources, which does not depend on X or Y; the peak correlation is at least 0.97 for every take. Tracker poses are then interpolated to the robot samples (linearly in position, spherically in rotation), and robot samples that fall in a gap of the tracker log are excluded rather than bridged.

Trajectories. An operator hand-guides the arm through four trajectories at 100 Hz while the phone records: a general motion, a free-space motion, a high-frequency motion and a motion with contact. The arm then replays each trajectory from its recorded joint positions at 0.25×, 0.5×, 0.75× and 1× speed, again with the phone recording. Including the hand-guided recordings, this gives 48 takes and 1,269 one-second windows, with end-effector speeds up to 1.2 m/s.

Error metric. Each take is split into 1 s windows, and each window reports its top end-effector speed and the average end-effector position error. Windows with missing tracker samples are excluded. The curves in Fig. 2a and Fig. 9 are the mean over ten equal-count speed bins, with 95% bootstrap confidence intervals. For the robot's own tracking error during the replays, we compare the commanded and achieved end-effector positions after removing one constant time lag per take, chosen to minimize the RMS error.

Recording Failures. On the 1× speed takes for each trajectory, SLAM failures are rejected where the end effector is detected moving at more than 7.0 m/s by the iPhUMI kit. Due to this, each take at 1× speed on average took 3-4 tries to record without failure, making it impractical to test iPhUMI tracking on even faster trajectories.

APPENDIX III
PUPPETEERING BASELINE SETUP

Data collection. For the puppeteering comparisons in Table III, an operator collects demonstrations by teleoperating an ALOHA 2 with its leader arms. Each task has 50 demonstrations recorded at 50 Hz, with joint positions of both arms and grippers (14 dimensions) and images from a top camera and two wrist cameras at 640×480. Table IV lists the datasets.

Policy. We train one ACT policy per task with the same architecture as ours (Table VI): ResNet-18 image backbone, no CVAE encoder, hidden dimension 512, feedforward dimension 3200, 4 encoder and 7 decoder layers with 8 heads, and a chunk size of 50 actions (1 s at 50 Hz). Policies take the three images at 320×180 and the joint positions, and are trained with batch size 256, learning rate 2×10^{-4}, backbone learning rate 10^{-5} and data augmentation.

Deployment. *Puppeteering* executes each chunk at the 50 Hz collection rate and queries a new chunk every 1 s. *Puppet Double Freq* executes the same chunks at 100 Hz and queries every 0.5 s. *Puppet Skip Actions* keeps the 50 Hz controller but executes every other action of each chunk, also querying every 0.5 s. Both sped-up variants therefore traverse each chunk in half the time of the collected data.

Task	Demos	Length	Training steps
Unstack Cups	50	12 s	200k
Battery Insertion	50	20 s	200k
Bus Table	50	18 s	200k

TABLE IV: ***Puppeteering Data.*** ALOHA 2 teleoperation datasets used for the puppeteering baselines, recorded at 50 Hz; Length is the recorded episode length.

APPENDIX IV
CHUNK BOUNDARY JUMPS

Fig. 8 measures the discontinuities between action chunks in the speed comparisons of Table III. At each

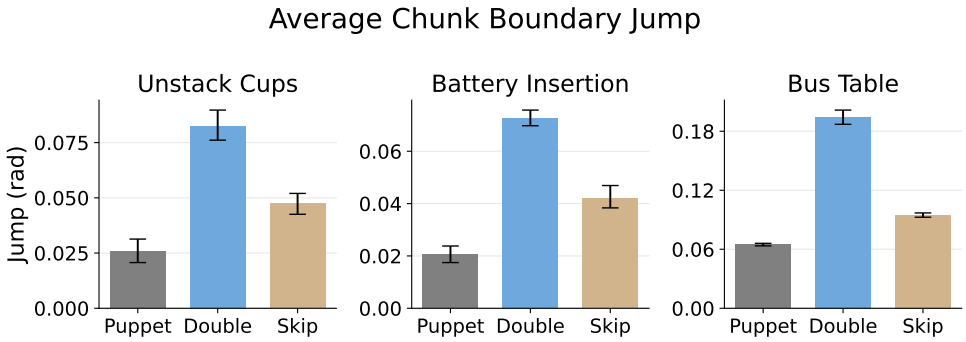


Fig. 8: ***Average Chunk Boundary Jump.*** Average change in commanded joint positions at chunk boundaries for Puppeteering (Puppet), Puppet Double Freq (Double) and Puppet Skip Actions (Skip). Bars: mean over 20 rollouts per condition (19 for Puppeteering on ***Bus Table***)

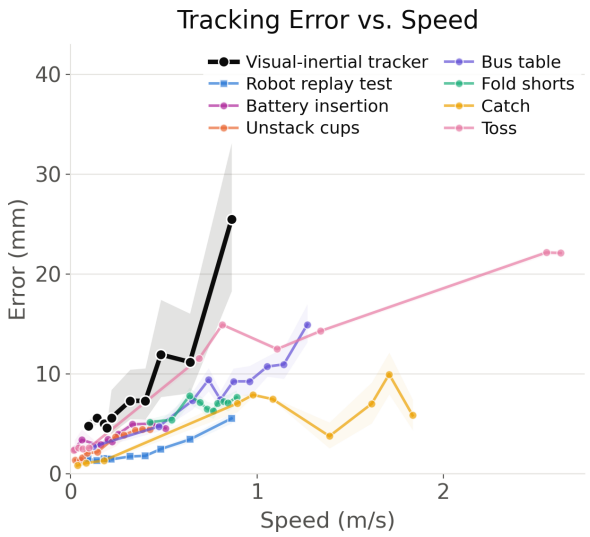


Fig. 9: ***Tracking Error During Policy Rollouts.*** Average position error in 1 s windows, binned by end-effector speed (shaded: 95% bootstrap CI). Black: a handheld visual-inertial tracker mounted on the robot wrist, against encoder ground truth. Colored: commanded vs. achieved end-effector position on our arms, during a tracking-test replay and policy rollouts of each task. Visual-inertial error grows quickly with speed and exceeds the robot's own tracking error at every speed it was tested.

policy update, we take the change in commanded joint positions between the last action of the previous chunk and the first action of the new one, as the Euclidean norm over the 12 arm joints (grippers excluded). Each rollout contributes the average jump over its chunk boundaries, and every rollout is included, failures as well. Doubling the controller frequency makes the jumps about 3–3.5× those of puppeteering, while skipping every other action makes them 1.5–2×.



Fig. 10: ***Visual-Inertial Setup.*** We add features beneficial to visual-inertial tracking, including a patterned tablecloth, bright lighting, 3-dimensional objects, and background clutter for maximum tracking performance.

APPENDIX V
POLICY TRAINING DETAILS

We train ACT [1] policies on the demonstrations. Table V lists the datasets and Table VI the ACT configuration. Images are recorded at 360×640 and downsampled to 180×320 for the policy, and training augments them with random crops covering 95% of each side, resized back to the input resolution. Policies predict absolute joint positions and gripper commands, are conditioned on joint positions only (no velocities), and are trained with seed 0 on a single GPU (NVIDIA L40S, A40 or RTX 4090), holding out 1% of the episodes for validation.

Task	Arms	Cameras	Rate	Demos	Length
Unstack Cups	ARX, 2	L, R, Top	50 Hz	50	4 s
Battery Insertion	ARX, 2	L, R, Top	50 Hz	50	7 s
Bus Table	ARX, 2	L, R, Top	50 Hz	50	6 s
Fold Shorts	ARX, 2	L, R, Top	50 Hz	50	5 s
Toss & Catch	HESSIAN, 2	L, R, Top	100/200 Hz	50	2 s

TABLE V: ***Training Data.*** Demonstrations per task, recorded by kinesthetic teaching. L and R are the wrist cameras; Length is the task duration. Toss & Catch records the tossing arm at 100 Hz and the catching arm at 200 Hz.

Policy	ACT, no CVAE encoder
Image backbone	ResNet-18
Hidden / feedforward dim.	512 / 3200
Encoder / decoder layers	4 / 7
Attention heads	8
Chunk size	50
Batch size	256
Learning rate	2×10^{-4}
Optimizer	AdamW, weight decay 10^{-4}
Training steps	200k

TABLE VI: ***ACT Configuration.***