FUNCBRIDGE: FUNCTIONAL TOOL-USE GENERALIZATION VIA KEYPOINT TRAJECTORY REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

While humans readily repurpose a book, a stone, or a shoe to drive a nail, robots trained on specific tools fail to transfer the same function to novel ones – a gap we formalize as *functional generalization*. Functionally equivalent tools share visually recognizable functional intent, such as where contact can occur and how a contact region should move to the target. However, this perceptual similarity does not directly carry over to action space, where each tool demands a different motor pattern to realize the function. To bridge this gap, we explore intermediate representations including affordance images, human video prompts, functional videos and object masks, and 2D keypoint trajectories, finding that keypoint trajectories best balance functional expressiveness and action groundability. Building on this, we present FUNCBRIDGE, a two-stage framework that decouples functional reasoning from action execution: learning to predict generalizable keypoint trajectories from action-free data, then grounding them into robot actions with limited demonstrations. Across a benchmark spanning ten tools and three functions, including hitting, sweeping, and hooking, FUNCBRIDGE consistently outperforms state-of-the-art methods on unseen tools in both simulation and the real world.

1 INTRODUCTION

In open-ended real-world environments, the right tool is rarely at hand, yet humans adapt effortlessly by repurposing available objects to accomplish diverse functions. For example, in hitting, a book, a stone, or a shoe can be used to drive a nail because functionally equivalent tools share a common functional intent: a contact region to strike with and a motion that brings it onto the target. This capacity, which we term *functional generalization*, is a hallmark of human dexterity that robots have yet to achieve. Current manipulation policies overfit to the appearance and geometry of seen tools, failing when handed a novel one that serves the same function (Chen et al., 2025; Turpin et al., 2021; Qin et al., 2023).

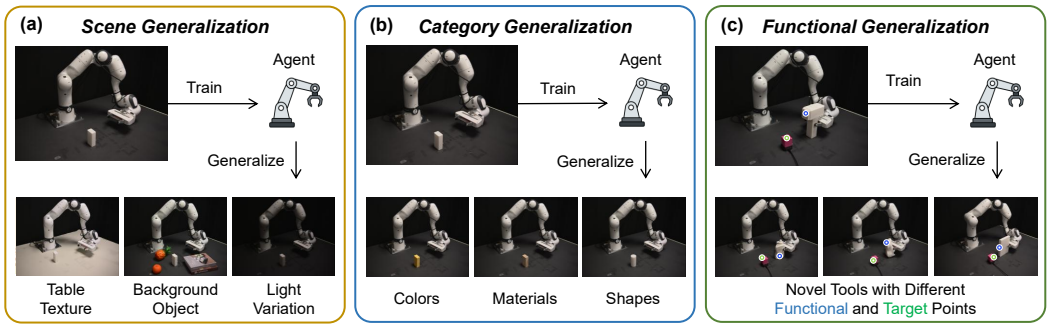


Figure 1: **Comparison of functional generalization with existing generalization settings.** (a) Scene generalization handles visual variations. (b) Category generalization transfers across object properties. (c) Functional generalization uses unseen tools for the same function.

Functional generalization is fundamentally harder than conventional generalization. As illustrated in Fig. 1, scene- and category-level generalization require tolerating visual variations while the under-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

lying motion stays the same (Goyal et al., 2023; Shridhar et al., 2023; 2022; Nair et al., 2023). Functional generalization, by contrast, demands adapting the motion to each tool's geometry: striking a target with a novel tool requires locating its contact region, aligning it, and producing an appropriate motion, even when the tool's shape and required trajectory differ from training. The core difficulty is a mismatch: functionally equivalent tools exhibit common functional intent that can be perceived from visual observations, but realizing the same function requires different motions across tools.

Bridging this perception-to-action gap requires an intermediate representation that carries functional intent across tools, satisfying two competing demands: it should be expressive enough to capture where contact can occur and how a selected contact region should move toward the target, yet abstract enough to discard tool-specific appearance that may hinder generalization to unseen tools. Dense representations like raw video entangle function-relevant cues with tool-specific appearance, while overly sparse ones discard the geometric and temporal structure needed for precise action grounding. This trade-off raises the central question of our work: *which representation best transfers functional intent from perception to action?*

To answer this question, we evaluate intermediate representations including affordance images, human video prompts, functional videos and object masks, and 2D keypoint trajectories, finding that keypoint trajectories best balance functional expressiveness and action groundability: they capture function-relevant contact and motion structure while discarding tool-specific appearance that causes overfitting. Building on this, we propose FUNCBRIDGE, a two-stage framework that decouples functional reasoning from action execution. It learns a functional keypoint predictor from large-scale action-free data and an action-grounding policy from limited action-labeled demonstrations. At inference, the predictor predicts function-relevant keypoint trajectories, which the execution policy grounds into robot actions. We further introduce a benchmark spanning ten tools and three functions, including hitting, sweeping, and hooking, on which FUNCBRIDGE consistently outperforms state-of-the-art methods on unseen tools in both simulation and the real world, achieving substantial gains in average success rate. In summary, our contributions are threefold:

- We formalize *functional generalization* in robotic tool-use and identify the perception-to-action gap as its core difficulty.
- We find that 2D keypoint trajectories best balance functional expressiveness and action groundability, and propose FUNCBRIDGE, a two-stage framework that learns functional keypoint planning from action-free data and action grounding from limited robot demonstrations.
- Across ten tools and three functions in simulation, with real-world validation, we demonstrate that FUNCBRIDGE consistently outperforms state-of-the-art methods on unseen tools.

2 RELATED WORKS

Generalization Problems in Robotic Manipulation. Generalization has been a central problem in robotic manipulation, and existing studies mainly evaluate it from three perspectives: scene-level, category-level, and cross-embodiment generalization. Scene-level generalization focuses on whether a policy can remain robust when the surrounding visual or physical contexts vary, such as changes in object layouts, lighting conditions, or camera viewpoints (Goyal et al., 2023; Shridhar et al., 2023; 2022; Nair et al., 2023). Category-level generalization studies whether robots can transfer manipulation skills to unseen objects that belong to the same semantic category while differing in geometry, size, or texture (Gao & Tedrake, 2021; Mo et al., 2021; Geng et al., 2023). More recently, cross-embodiment generalization has attracted attention, which aims to train a unified policy that can transfer across different robot platforms, such as robotic arms and humanoids (O'Neill et al., 2024; Team et al., 2024; Intelligence et al., 2025). In this work, we go beyond generalization over visual appearance and robot embodiment and focus on *functional generalization*, in which an agent is required to accomplish the same function using diverse, unseen tools.

Improving Generalization in Robot Policy Learning. To improve manipulation generalization, the most straightforward way is scaling up action-labeled robotic data and training policies end-to-end (Zitkovich et al., 2023; O'Neill et al., 2024; Team et al., 2024; Khazatsky et al., 2024; Jia et al., 2026). However, collecting large-scale robotic data is time-consuming and labor-intensive, especially for complicated tool-use manipulations. Reducing the dependence on action labels, some

works learn transferable motion priors from large-scale action-free video data (Nair et al., 2023; Wang et al., 2023; Wu et al., 2024; Ma et al., 2026; Pai et al., 2025; An et al., 2026; Hou et al., 2026). For example, MimicPlay (Wang et al., 2023) learns high-level latent plans from human play videos and grounds them into robot actions with a low-level visuomotor policy. Although videos contain rich motion and interaction cues, dense future frames do not explicitly reveal a compact abstraction of function-relevant intent. As a result, effectively learning functional priors from videos often requires extremely extensive pretraining. Another line of work improves action grounding with more structured intermediate representations, such as interaction heatmaps (Mo et al., 2021; Bahl et al., 2023; Wu et al., 2025; Li et al., 2026), task-oriented grasps (Fang et al., 2020), object-centric representations (Zhu et al., 2023; Chen et al., 2025; Chapin et al., 2025; Cao et al., 2026), or keypoint trajectories (Gao & Tedrake, 2021; Huang et al., 2025; Turpin et al., 2021; Wen et al., 2023; Cao et al., 2026). For instance, ATM (Wen et al., 2023) predicts future trajectories of arbitrary points from video and uses them to guide a low-level policy for action generation. However, these methods mainly focus on standard manipulation tasks, such as object pick-up, while overlooking tool-use scenarios that require capturing more abstract functional intent, which are systematically investigated in our work.

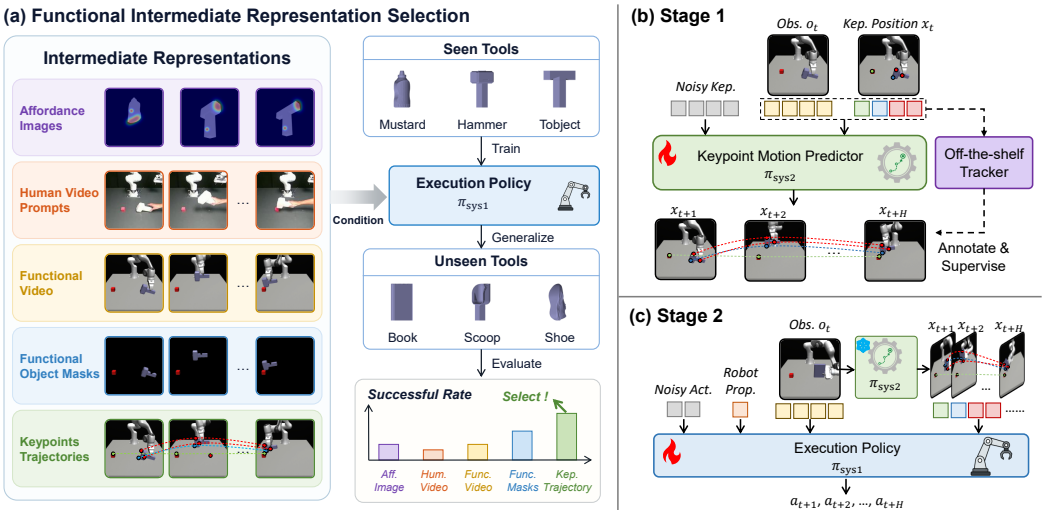


Figure 2: **Overview of the FUNCBRIDGE.** (a) Evaluation and selection of functional intermediate representations. (b) In Stage 1, the System-2 planner learns to predict future keypoint-based functional plans from action-free observations. (c) In Stage 2, the System-1 execution policy grounds the predicted functional plans into executable robot actions.

3 METHODS

3.1 PROBLEM FORMULATION

We formulate functional generalization in tool-use manipulation as a Markov Decision Process (MDP), where the state $\mathbf{s}_t$ consists of the visual observation $\mathbf{o}_t$, the proprioceptive state $\mathbf{q}_t$, and optionally a functional intermediate representation $\mathbf{x}_t$ (e.g., keypoint trajectories) that carries function-relevant cues bridging perception and action, and the action $\mathbf{a}_t$ is the motor command. A task function (e.g., hitting a target) is defined over seen tools $\mathcal{T}_{\text{seen}}$ for training and disjoint unseen tools $\mathcal{T}_{\text{unseen}}$ for evaluation, where the function stays fixed but the tool, its appearance, and the required contact positions and motions all change.

Two types of data are available for learning. The first is a large-scale action-free dataset $\mathcal{D}_U = \{(\mathbf{O}^{(i)}, \mathbf{X}^{(i)})\}_{i=1}^{N_U}$, consisting of visual observations $\mathbf{O}^{(i)} = \{\mathbf{o}_t^{(i)}\}_{t=1}^{T}$ and corresponding functional intermediate representations $\mathbf{X}^{(i)} = \{\mathbf{x}_t^{(i)}\}_{t=1}^{T}$, which is abundant and can span a broader range of tools since it requires no robot labels. The second is a smaller action-labeled robotic dataset $\mathcal{D}_L = \{(\mathbf{O}^{(i)}, \mathbf{X}^{(i)}, \mathbf{Q}^{(i)}, \mathbf{A}^{(i)})\}_{i=1}^{N_L}$, which additionally provides proprioceptive states $\mathbf{Q}^{(i)}$ and

ground-truth actions $\mathbf{A}^{(i)}$ but is costly to collect, so $|\mathcal{D}_L| \ll |\mathcal{D}_U|$. This data asymmetry, together with the perception-to-action gap discussed in Sec. 1, motivates a two-stage decomposition:

$$\underbrace{\pi_{\text{FuncBridge}}(\mathbf{a}_{t:t+H} \mid \mathbf{o}_t, \mathbf{q}_t, \mathbf{x}_t)}_{\text{FuncBridge}} = \underbrace{\pi_{\text{sys2}}(\hat{\mathbf{X}}_{t:t+H} \mid \mathbf{o}_t, \mathbf{x}_t)}_{\text{functional reasoning}} \underbrace{\pi_{\text{sys1}}(\mathbf{a}_{t:t+H} \mid \mathbf{o}_t, \mathbf{q}_t, \hat{\mathbf{X}}_{t:t+H})}_{\text{grounded execution}}. \quad (1)$$

FUNCBRIDGE learns a future predictor on $\mathcal{D}_U$ for forecasting functional intermediate representations from observations, and an execution policy on $\mathcal{D}_L$ for grounding representations into actions.

3.2 FUNCTIONAL INTERMEDIATE REPRESENTATIONS SELECTION

As discussed in Sec. 1, a desired functional intermediate representation $\mathbf{X}_{t:t+H}$ should preserve the contact and motion structure shared across tools, while remaining predictable from action-free observations and groundable into precise robot actions. We hypothesize that 2D keypoint trajectories form a suitable choice of $\mathbf{X}_{t:t+H}$. They discard tool-specific appearance that may cause overfitting, while retaining compact geometric information such as the contact point, target position, and affordance-relevant tool structure. At the same time, they provide a structured temporal description of how function-relevant points should move, which makes them easy to predict from existing vision models (Wen et al., 2023) and to ground into actions. To validate this hypothesis, we compare keypoint trajectories with four alternative choices of $\mathbf{X}_{t:t+H}$ spanning different temporal forms as shown in Fig. 2: affordance images as static conditions, human video prompts as pooled conditions, and functional videos and object masks as sequential conditions.

Affordance Image. Given an affordance image $I^{(i)}_{\text{aff}}$, we encode it with an image encoder E_{img} (He et al., 2016) and use the resulting feature as a time-invariant condition: $x^{(i)}_t = E_{\text{img}}(I^{(i)}_{\text{aff}}) \in \mathbb{R}^d$.

Human Video Prompt. Given a human video $V^{(i)}_{\text{hum}} = \{I^{(i)}_{\text{hum},t}\}_{t=1}^{T}$, we encode it with a video encoder E_{vid} (Bertasius et al., 2021) and apply temporal average pooling to obtain a sequence-level feature, which is used as a time-invariant condition: $x^{(i)}_t = \text{AvgPool}(E_{\text{vid}}(V^{(i)}_{\text{hum}})) \in \mathbb{R}^d$.

Functional Videos and Object Masks. Given a ground-truth functional execution video $V^{(i)}_{\text{func}} = \{I^{(i)}_{\text{func},t}\}_{t=1}^{T}$, we encode each frame with an image encoder E_{img}, yielding a temporally varying condition $x^{(i)}_t = E_{\text{img}}(I^{(i)}_{\text{func},t}) \in \mathbb{R}^d$. For object masks, as shown in Fig. 2, we remove the background while retaining the tool and target object in each frame, with the masked frames encoded similarly.

Keypoint Trajectory. We denote the keypoint trajectory as $P^{(i)} = \{p^{(i)}_t\}_{t=1}^{T}$, where each frame $p^{(i)}_t = \{p^{(i),k}_t\}_{k=1}^{K}$ contains K 2D keypoints. At time step t, we flatten the keypoint coordinates and encode them with a keypoint encoder E_{kp}, yielding $x^{(i)}_t = E_{\text{kp}}(\text{Flatten}(p^{(i)}_t)) \in \mathbb{R}^d$.

To evaluate these candidates, we conduct a provided-reference experiment in Fig. 2 (a), where each representation is pre-extracted for both seen and unseen tools and directly used as the condition for the execution policy π_{sys1}. The policy is trained on action-labeled data from seen tools in $\mathcal{D}_L$ and evaluated on unseen tools for the hitting function, where a rollout is successful if the designated hitting region strikes the target. Among all candidates, 2D keypoint trajectories achieve the best performance, since they go beyond static region cues and dense visual motion by providing a compact, structured description of how function-relevant points move and align over time. We therefore adopt 2D keypoint trajectories as the functional intermediate representation for FUNCBRIDGE.

3.3 FUNCBRIDGE: TWO-STAGE FUNCTIONAL REASONING AND GROUNDED EXECUTION

After selecting 2D keypoint trajectories as the functional intermediate representation, we design a two-stage framework to decouple functional reasoning from action execution. This enables FUNCBRIDGE to learn transferable functional keypoint motion from action-free demonstrations, while grounding the predicted trajectories into robot actions with limited action-labeled data.

Stage 1: Action-free Functional Reasoning. As shown in Fig. 2 (b), we train the keypoint motion predictor π_{sys2} on the action-free dataset $\mathcal{D}_U$ to predict future functional keypoint trajectories. Given the visual observation $\mathbf{o}_t$ and keypoint coordinates $\mathbf{x}_t$ at time-step t, π_{sys2} generates a future keypoint trajectory $\hat{\mathbf{X}}_{t:t+H}$ over horizon H.

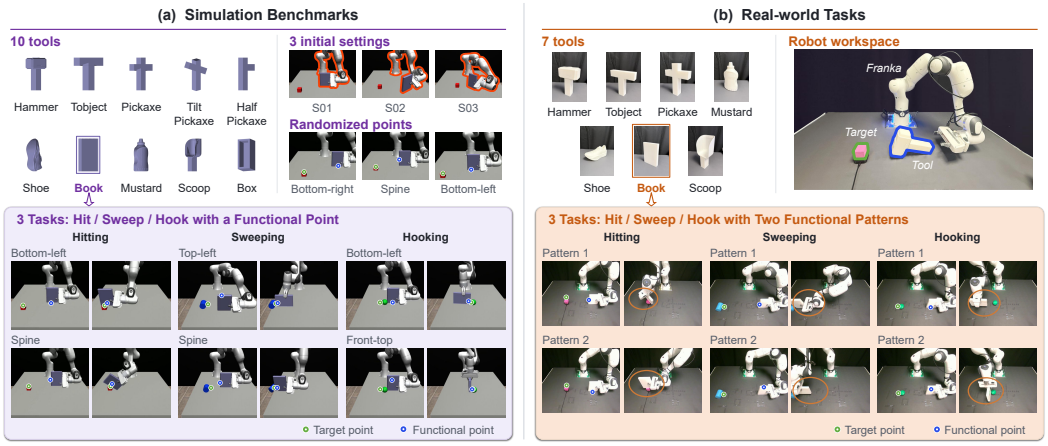


Figure 3: **Simulation Benchmark and Real-World Setting.** (a) The simulation benchmark contains 10 tools and three functions, including hitting, sweeping, and hooking, with multiple initial settings and randomized functional and target points. (b) The real-world benchmark uses a Franka robot with 7 tools and the same three functions, each evaluated under two functional patterns.

Formally, we instantiate π_{sys2} as a conditional flow-matching model in the keypoint space. Given a ground-truth keypoint trajectory $\mathbf{X}_{t:t+H}$, a Gaussian noise sample $\epsilon \sim \mathcal{N}(0, \mathbf{I})$, and an interpolation time $\tau \sim \mathcal{U}(0, 1)$, we define the interpolated keypoint trajectory as $\mathbf{X}^{\tau}_{t:t+H} = \tau \mathbf{X}_{t:t+H} + (1-\tau)\epsilon$. The keypoints motion predictor learns a conditional velocity field u_ϕ by minimizing

$$\mathcal{L}_{\text{sys2}} = \mathbb{E}_{\tau,\epsilon,\mathcal{D}_U}\left[\left\| u_\phi\left(\mathbf{X}^{\tau}_{t:t+H}, \tau \mid \mathbf{o}_t, \mathbf{x}_t\right) - \left(\mathbf{X}_{t:t+H} - \epsilon\right)\right\|_2^2\right]. \tag{2}$$

During training, our task specifies the functional point on the tool and the target point on the object. Given the functional point, we use SAM2 (Ravi et al., 2025) to segment the corresponding tool mask and sample 5 keypoints on the mask with Farthest Point Sampling (FPS), resulting in 7 keypoints in total. As shown in Fig. 2 (b), we then track these keypoints with CoTracker (Karaev et al., 2024) to obtain keypoint trajectories for training. Since this stage only requires visual observations and keypoint trajectories, System-2 can leverage action-free data to capture transferable functional intent as keypoint-based motions, without relying on large-scale robot action labels.

Stage 2: Grounded Execution Policy. In the second stage, we freeze the pretrained π_{sys2} and train the execution policy π_{sys1} on the action-labeled robotic dataset $\mathcal{D}_L$. Given the observation $\mathbf{o}_t$, robot state $\mathbf{q}_t$, and 2D keypoint coordinates $\mathbf{x}_t$, the frozen π_{sys2} first predicts future keypoint motions:

$$\hat{\mathbf{X}}_{t:t+H} = \pi_{\text{sys2}}(\mathbf{o}_t, \mathbf{x}_t). \tag{3}$$

To improve robustness to prediction errors, we apply random pixel perturbations to the $\hat{\mathbf{X}}_{t:t+H}$ during training, where each keypoint is shifted by 1–5 pixels with a probability of 0.5. The execution policy π_{sys1} then grounds the perturbed keypoint trajectories $\tilde{\mathbf{X}}_{t:t+H}$ into executable robot actions:

$$\hat{\mathbf{A}}_{t:t+H} = \pi_{\text{sys1}}(\mathbf{o}_t, \mathbf{q}_t, \tilde{\mathbf{X}}_{t:t+H}). \tag{4}$$

Similar to π_{sys2}, we instantiate π_{sys1} as a conditional flow-matching model in the action space. Given a ground-truth action chunk $\mathbf{A}_{t:t+H}$, a Gaussian noise sample $\epsilon \sim \mathcal{N}(0, \mathbf{I})$, and an interpolation time $\tau \sim \mathcal{U}(0, 1)$, we define the interpolated action trajectory as $\mathbf{A}^{\tau}_{t:t+H} = \tau \mathbf{A}_{t:t+H} + (1-\tau)\epsilon$. The action-grounded execution policy learns a conditional velocity field v_θ by minimizing:

$$\mathcal{L}_{\text{sys1}} = \mathbb{E}_{\tau,\epsilon,\mathcal{D}_L}\left[\left\| v_\theta\left(\mathbf{A}^{\tau}_{t:t+H}, \tau \mid \mathbf{o}_t, \mathbf{q}_t, \tilde{\mathbf{X}}_{t:t+H}\right) - \left(\mathbf{A}_{t:t+H} - \epsilon\right)\right\|_2^2\right]. \tag{5}$$

4 EXPERIMENTS

Functional generalization remains largely underexplored in robotic manipulation. In this work, we instantiate it across three representative tool-use functions, including hitting, sweeping, and hooking, and construct both simulation and real-world benchmarks to evaluate whether a policy can transfer each function to unseen tools with diverse geometries. Specifically, we test four hypotheses underlying the central claims of this work:

- **H1**: Functional generalization benefits from an explicit, function-aware intermediate representation rather than direct perception-to-action mapping or generic motion cues.
- **H2**: Among candidate intermediate representations, 2D keypoint trajectories best balance functional expressiveness and action groundability.
- **H3**: Functional generalization can be achieved with moderate action-labeled tool diversity.
- **H4**: Functional reasoning is scalable, which can benefit from more diverse action-free data and demonstrate greater autonomy through VLM-based functional point selection.

4.1 EXPERIMENTAL SETUP

Simulation Benchmark. As shown in Fig. 3, our simulation benchmark covers 10 tools and three functions, including hitting, sweeping, and hooking, with three initial settings per tool. For each tool-setting-function combination, we collect 30 demonstrations using function-specific MPPI planners, totaling 2,700 demonstrations. In the main setting, all methods are trained on action-labeled data from four tools (hammer, tobject, pickaxe, and mustard) and evaluated on three unseen tools held out from action-labeled training (scoop, shoe, and book) across all functions. Each unseen tool is evaluated under three initial settings with 30 rollouts per setting. For each demonstration, we randomize the target position and designated functional point, with different points possible for the same tool and function (e.g., using the hammer head or handle for hitting). A rollout is successful only if the specified function is completed using that point: *hitting* requires the hitting point to strike the red cube, *sweeping* requires the sweeping point to push the blue cylinder, and *hooking* requires the hooking point to pull the green ball inward.

Real-World Setting. We further construct a real-world benchmark on a Franka robot platform. The benchmark covers 7 tools and the same three functions, with two functional patterns defined for each function. For each training tool and functional pattern, we collect 20 expert demonstrations. For hitting, the policies are trained on hammer, pickaxe, and mustard and evaluated on three unseen tools (tobject, book, and shoe). For sweeping and hooking, they are trained on hammer and mustard and evaluated on the unseen book. Each functional pattern is evaluated with 10 real-world rollouts. Unlike simulation, the designated functional point specifies the required functional pattern rather than serving as a point-level success criterion. A rollout is successful if the robot completes the specified function following the corresponding pattern.

Baselines. We compare FUNCBRIDGE against three baselines covering end-to-end and keypoint-based policies. All baselines receive the same inputs as FUNCBRIDGE: visual observations, proprioceptive states, and the initial functional, target, and tool point positions. *Flow-Matching (FM)* (Lipman et al., 2023) and *Diffusion Policy (DP)* (Chi et al., 2025) are end-to-end visuomotor policies that map observations directly to actions, serving as references for the perception-to-action gap (Sec. 1). *ATM* (Wen et al., 2023) also conditions the execution policy on keypoint trajectories. Given the provided functional and target points, it uses its generic any-point trajectory model pretrained on LIBERO-90 to predict their trajectories. This serves as a keypoint-based baseline using generic trajectory modeling rather than explicitly predicting function-aware keypoint motion.

Evaluation Metrics. All experiments report the success rate (SR) as the main evaluation metric, which is defined as the number of successful trials divided by the total number of rollout trials.

4.2 MAIN RESULTS & ANALYSIS

Simulation Results. As shown in Tab. 1, FUNCBRIDGE outperforms all baselines across hitting, sweeping, and hooking on unseen tools. Compared with the best baseline results of 0.17, 0.30, and 0.14, it achieves average SRs of 0.36, 0.45, and 0.23, respectively. The weak performance of FM

Table 1: **Simulation Results on Unseen Tools.** Success rate (SR) on three unseen tools across hitting, sweeping, and hooking. Each tool is evaluated under 3 settings with 30 rollouts per setting.

	Hitting				Sweeping				Hooking			
Method	Scoop	Shoe	Book	Avg.	Scoop	Shoe	Book	Avg.	Scoop	Shoe	Book	Avg.
FM	0.07	0.03	0.16	0.09	0.49	0.23	0.19	0.30	0.22	0.06	0.14	0.14
DP	0.18	0.08	0.26	0.17	0.49	0.18	0.13	0.27	0.12	0.06	0.08	0.09
ATM	0.12	0.06	0.06	0.08	0.41	0.16	0.19	0.25	0.10	0.07	0.24	0.14
FUNCBRIDGE	**0.42**	**0.23**	**0.42**	**0.36**	**0.71**	**0.29**	**0.34**	**0.45**	**0.29**	**0.17**	**0.24**	**0.23**

Table 2: **Real-World Results on Unseen Tools.** Success rate (SR) under two functional patterns for hitting, sweeping, and hooking. Hitting is evaluated on three unseen tools, while sweeping and hooking are evaluated on the unseen book, with 10 rollouts per pattern.

	Hitting									**Sweeping**			**Hooking**		
	Tobject			Book			Shoe			Book			Book		
Method	P01	P02	Avg.	P01	P02	Avg.	P01	P02	Avg.	P01	P02	Avg.	P01	P02	Avg.
FM	0.00	0.60	0.30	0.50	0.00	0.25	0.00	0.20	0.10	0.30	0.00	0.15	0.20	0.00	0.10
FUNCBRIDGE	**0.60**	**0.80**	**0.70**	**0.60**	**1.00**	**0.80**	**0.20**	**0.60**	**0.40**	**0.80**	**0.30**	**0.55**	**0.90**	**0.30**	**0.60**

and DP suggests that direct perception-to-action mapping struggles to adapt tool-specific motions to unseen geometries without tool-specific action supervision. Despite being conditioned on the designated functional and target points, ATM performs less effectively, suggesting that its generic keypoint predictor may fail to produce effective function-relevant motion guidance for diverse tool geometries. Together, these results support **H1**: explicit function-aware intermediate representation provides useful guidance for functional generalization.

Real-World Results. We test FUNCBRIDGE on the real-world benchmark across all three functions. As shown in Tab. 2, FUNCBRIDGE consistently outperforms the FM baseline across all evaluated unseen-tool and functional-pattern settings. These results extend the evidence for **H1** to real-world manipulation, showing that function-aware keypoint plans provide more reliable guidance.

Failure Case Analysis. We further examine representative baseline failures on the hitting function in both simulation and the real world. As shown in Fig. 4, the baselines exhibit three typical failure modes: *pass over*, where the tool moves across the target without establishing the intended contact; *imprecise hitting*, caused by insufficient fine-grained spatial alignment; and *tool dropping*, associated with unstable or overly aggressive motions. In the corresponding cases, FUNCBRIDGE successfully completes the task by predicting function-aware keypoint trajectories that guide the functional region toward the target with the appropriate motion and alignment. Additional qualitative comparisons across hitting, sweeping, and hooking in both simulation and the real world are provided in App. B.

4.3 ABLATION STUDIES

Beyond the main comparisons, we conduct two ablations on the key design choices of FUNCBRIDGE: the functional intermediate representation and the diversity of action-labeled tools. Unless otherwise specified, all ablations are conducted on the hitting function in simulation.

4.3.1 EFFECT OF FUNCTIONAL INTERMEDIATE REPRESENTATIONS.

As discussed in Sec. 3.2, we investigate the effect of different functional intermediate representations by directly training and evaluating the execution policy π_{sys1} with pre-extracted representations as conditions. Results in Tab. 3 show that static affordance images and pooled human video prompts improve generalization, indicating that functional-region cues and motion demonstrations provide useful guidance. Providing temporally varying representations further improves performance, while object masks achieve stronger generalization by only retaining object-centric motion cues. Among all representations, 2D keypoint trajectories perform best, as they compactly encode fine-grained

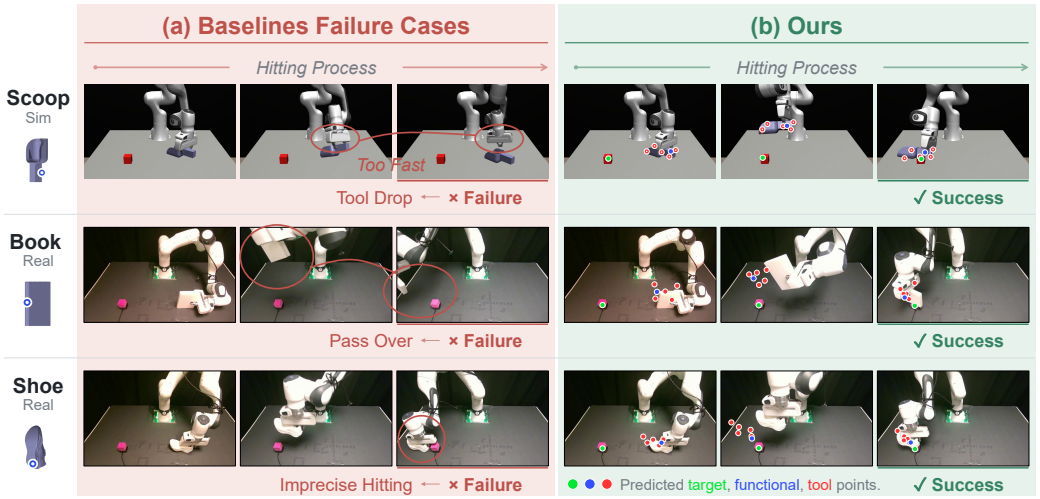


Figure 4: **Qualitative Comparison on Unseen Tools.** Representative baseline failure cases and successful rollouts of FUNCBRIDGE for hitting in simulation and the real world.

Table 3: **Ablation Results on Functional Intermediate Representations.** Success rate (SR) for hitting on unseen tools when different representations condition the same execution policy π_{sys1}.

Representation	Temporal input	Scoop	Shoe	Book	Avg.
None (FM)	–	0.07	0.03	0.16	0.09
Affordance image	Static	0.34	0.11	0.16	0.20
Human video prompt	Pooled	0.26	0.09	0.23	0.19
Per-frame video	Sequence	0.28	0.13	0.29	0.23
Per-frame object masks	Sequence	0.37	0.16	0.43	0.32
Keypoint trajectories	Sequence	**0.50**	**0.31**	**0.67**	**0.49**

functional and target locations together with their structured motion over time. These results support **H2**: keypoint trajectories best balance functional expressiveness and action groundability.

4.3.2 EFFECT OF ACTION-LABELED TOOL DIVERSITY.

Beyond the standard 4-to-3 setting, we vary the number of action-labeled training tools while consistently evaluating on the same three unseen tools: scoop, shoe, and book. As shown in Tab. 4, increasing the training diversity from 1 to 4 tools substantially improves the average SR from 0.16 to 0.36, while further increasing it to 7 tools yields a smaller gain to 0.39. These results support **H3**: functional generalization can be achieved with moderate action-labeled tool diversity in our two-stage framework.

Table 4: **Ablation on Action-Labeled Tool Diversity.** Success rate (SR) of FUNCBRIDGE when the execution policy π_{sys1} is trained with action-labeled data from $M = 1,4,7$ tools and consistently evaluated on the same three unseen tools: scoop, shoe, and book.

Method	Tools			
	Scoop	Shoe	Book	Avg.
FUNCBRIDGE (1-to-3)	0.14	0.10	0.23	0.16
FUNCBRIDGE (4-to-3)	0.42	0.23	0.42	0.36
FUNCBRIDGE (7-to-3)	0.39	0.27	0.52	0.39

4.4 SCALABILITY OF FUNCBRIDGE

Our default setting learns functional reasoning from action-free simulation videos, while the functional and target points are provided during evaluation. We further examine **H4** by extending FUNCBRIDGE with multi-source action-free data and VLM-based function-relevant point selection.

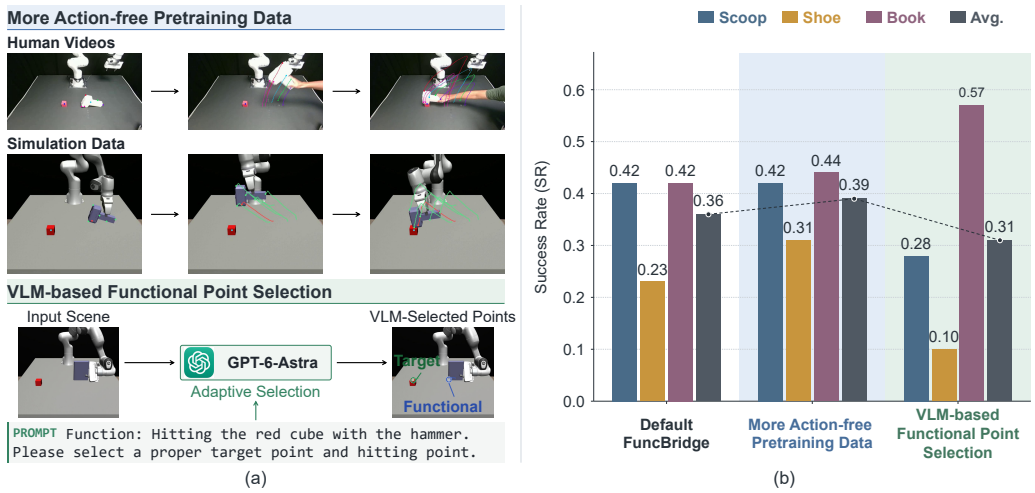


Figure 5: **Scalability of FUNCBRIDGE.** (a) Extensions using human videos and VLM-based functional point selection. (b) Success rates on unseen tools for the default model and each extension.

4.4.1 SCALING ACTION-FREE PRETRAINING WITH HUMAN VIDEOS.

As shown in Fig. 5 (a), we additionally collect and annotate keypoints of 525 human videos for the hitting function (see App. A.1 for details). Together with the original action-free data, these videos are used to pretrain π_{sys2}, while π_{sys1} is fine-tuned with the same four-tool action-labeled robotic data. As shown in Fig. 5 (b), incorporating human videos improves the average SR from 0.36 to 0.39. This result shows that FUNCBRIDGE can benefit from heterogeneous action-free data, suggesting the potential of broader data sources for functional reasoning.

4.4.2 FLEXIBLE VLM-BASED FUNCTIONAL POINT SELECTION.

Given the observed scene and task instruction, GPT-6-Astra is prompted to adaptively select the functional point on the tool and the target point, which are then used by FUNCBRIDGE for functional planning and grounded execution. A rollout is considered successful if the VLM-selected functional point is used to strike the target object, following the same hitting objective as the default setting. As shown in Fig. 5, VLM-selected points achieve an average SR of 0.31, compared with 0.36 using manually specified points. The performance gap mainly comes from VLM errors that select incorrect or unsuitable hitting points, such as the front side of a book. Nevertheless, the results show that VLM-based point selection can be integrated with FUNCBRIDGE to reduce manual specification and support more autonomous functional tool use.

5 CONCLUSION AND LIMITATION

We identify *functional generalization* as an important yet underexplored problem, where an agent must accomplish the same function with unseen tools. We introduce FUNCBRIDGE, a two-stage framework that decouples action-free functional reasoning from grounded execution. Across hitting, sweeping, and hooking, simulation and real-world experiments show improved generalization to unseen tools over existing baselines. Further analyses show that FUNCBRIDGE benefits from heterogeneous action-free data, including human videos, and can integrate VLM-based functional point selection for more autonomous tool use.

Limitations and Future Work. FUNCBRIDGE remains limited in generalizing to unseen tools that require substantially different functional motions. Future work could leverage larger-scale egocentric or web videos with more diverse tool-use behaviors to further improve functional generalization. In addition, our method focuses on functional motion reasoning and does not explicitly model dexterous interaction, such as grasp adaptation or in-hand manipulation, which is also essential for general tool use and could be incorporated in future work.

AI USE STATEMENT

In this work, we use generative AI tools, including ChatGPT and Codex, for language polishing and code assistance. The authors develop the core research ideas, methodology, experimental design, and scientific conclusions. Generative AI is not used to determine the main technical contributions or experimental claims. We also use generative AI to assist with figure design, layout, and formatting, while the final figures are manually assembled, reviewed, and revised by the authors. GPT-6-Astra is additionally used as the VLM component in the functional-point-selection experiment, where it selects the functional point on the tool and the target point from the observed scene and task instruction. The authors review and revise all AI-assisted text and manually inspect and test all AI-assisted code for correctness. We take full responsibility for the final content of this work, including the text, claims, code, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide detailed descriptions of the benchmark construction, data collection, training procedures, and evaluation protocols throughout the main paper and appendix. Sec. 4.1 specifies the simulation and real-world benchmarks, training and evaluation splits, demonstration collection, and evaluation metrics. Appendix A.1 describes the construction and extraction of the functional intermediate representations, including the human-video data used for action-free pretraining; Appendix A.2 details the MPPI-based demonstration generation procedure; and Appendix A.3 provides the training and implementation settings for FUNCBRIDGE and the baselines. During the review process, we additionally provide supplementary videos showing complete rollouts from both simulation and real-world experiments. The complete code and datasets will be publicly released upon acceptance.

REFERENCES

Tuo An, Jindou Jia, Gen Li, Jingliang Li, Chuhao Zhou, Pengfei Liu, Bofan Lyu, Jiaqi Bai, Xinying Guo, Geng Li, et al. Feedback world model enables precise guidance of diffusion policy. *arXiv preprint arXiv:2605.15705*, 2026.

Shikhar Bahl, Russell Mendonca, Lili Chen, Unnat Jain, and Deepak Pathak. Affordances from human videos as a versatile representation for robotics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13778–13790, 2023.

Gedas Bertasius, Heng Wang, and Lorenzo Torresani. Is space-time attention all you need for video understanding? In *Proceedings of the International Conference on Machine Learning*, 2021.

Shuxin Cao, Liquan Wang, Masoud Moghani, Benjamin Joffe, and Animesh Garg. KeyGen: Unsupervised keypoint-based object-centric representations for category-level policy generalization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2026.

Alexandre Chapin, Bruno Machado, Emmanuel Dellandrea, and Liming Chen. Object-centric representations improve policy generalization in robot manipulation. *arXiv preprint arXiv:2505.11563*, 2025.

Haonan Chen, Cheng Zhu, Yunzhu Li, and Katherine Driggs-Campbell. Tool-as-interface: Learning robot policies from human tool usage through imitation learning. *arXiv e-prints*, pp. arXiv–2504, 2025.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 2025.

Kuan Fang, Yuke Zhu, Animesh Garg, Andrey Kurenkov, Viraj Mehta, Li Fei-Fei, and Silvio Savarese. Learning task-oriented grasping for tool manipulation from simulated self-supervision. *The International Journal of Robotics Research*, 39(2-3):202–216, 2020. doi: 10.1177/0278364919872545.

Wei Gao and Russ Tedrake. kpam 2.0: Feedback control for category-level robotic manipulation. *IEEE Robotics and Automation Letters*, 2021.

Haoran Geng, Helin Xu, Chengyang Zhao, Chao Xu, Li Yi, Siyuan Huang, and He Wang. Gapartnet: Cross-category domain-generalizable object perception and manipulation via generalizable and actionable parts. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023.

Ankit Goyal, Jie Xu, Yijie Guo, Valts Blukis, Yu-Wei Chao, and Dieter Fox. Rvt: Robotic view transformer for 3d object manipulation. In *Conference on Robot Learning*, 2023.

Kaiming 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*, 2016.

Bohan Hou, Gen Li, Jindou Jia, Tuo An, Xinying Guo, Sicong Leng, Haoran Geng, Yanjie Ze, Tatsuya Harada, Philip Torr, et al. World model for robot learning: A comprehensive survey. *arXiv preprint arXiv:2605.00080*, 2026.

Wenlong Huang, Chen Wang, Yunzhu Li, Ruohan Zhang, and Li Fei-Fei. Rekep: Spatio-temporal reasoning of relational keypoint constraints for robotic manipulation. In *Conference on Robot Learning*, 2025.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. pi0.5: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Jindou Jia, Shixuan Han, Meng Wang, Gen Li, Zihan Yang, Sicheng Zhou, Kexin Guo, Jianfei Yang, Xiang Yu, Wei Wang, et al. Physics filtering favors the generalization of robot learning. *npj Robotics*, 4(1):48, 2026.

Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker: It is better to track together. In *European conference on computer vision*, 2024.

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.

Jingliang Li, Jindou Jia, Tuo An, Chuhao Zhou, Xiangyu Chen, Shilin Shan, Boyu Ma, Bofan Lyu, Gen Li, and Jianfei Yang. Compassad: Intent-driven 3d affordance grounding in functionally competing objects. *arXiv preprint arXiv:2604.02060*, 2026.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *International Conference on Learning Representations*, 2023.

Teli Ma, Jia Zheng, Zifan Wang, Chunli Jiang, Andy Cui, Junwei Liang, and Shuo Yang. Dit4dit: Jointly modeling video dynamics and actions for generalizable robot control. *arXiv preprint arXiv:2603.10448*, 2026.

Kaichun Mo, Leonidas J Guibas, Mustafa Mukadam, Abhinav Gupta, and Shubham Tulsiani. Where2act: From pixels to actions for articulated 3d objects. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021.

Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. In *Conference on Robot Learning*, 2023.

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)*, 2024.

Jonas Pai, Liam Achenbach, Victoriano Montesinos, Benedek Forrai, Oier Mees, and Elvis Nava. mimic-video: Video-action models for generalizable robot control beyond vlas. *arXiv preprint arXiv:2512.15692*, 2025.

Meiying Qin, Jake Brawer, and Brian Scassellati. Robot tool use: A survey. *Frontiers in Robotics and AI*, 2023.

Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, et al. Sam 2: Segment anything in images and videos. In *International Conference on Learning Representations*, 2025.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Conference on robot learning*, 2022.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, 2023.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. *arXiv preprint arXiv:2405.12213*, 2024.

Dylan Turpin, Liquan Wang, Stavros Tsogkas, Sven Dickinson, and Animesh Garg. Gift: Generalizable interaction-aware functional tool affordances without labels. *Robotics: Science and Systems*, 2021.

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. In *Conference on Robot Learning*, 2023.

Chuan Wen, Xingyu Lin, John So, Kai Chen, Qi Dou, Yang Gao, and Pieter Abbeel. Any-point trajectory modeling for policy learning. *arXiv preprint arXiv:2401.00025*, 2023.

Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. In *International Conference on Learning Representations*, 2024.

Shijie Wu, Yihang Zhu, Yunao Huang, Kaizhen Zhu, Jiayuan Gu, Jingyi Yu, Ye Shi, and Jingya Wang. Afforddp: Generalizable diffusion policy with transferable affordance. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 6971–6980, 2025.

Yifeng Zhu, Zhenyu Jiang, Peter Stone, and Yuke Zhu. Learning generalizable manipulation policies with object-centric 3d representations. In *Conference on Robot Learning*, 2023.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pp. 2165–2183. PMLR, 2023.

APPENDIX

A Benchmark Construction and Implementation Details

A.1 Extraction of Functional Intermediate Representations

A.2 MPPI-based Demonstration Generation

A.3 Training and Implementation Details

B Additional Experimental Results

B.1 More Qualitative Results in the Real World

B.2 More Qualitative Results in Simulation

A BENCHMARK CONSTRUCTION AND IMPLEMENTATION DETAILS

This section provides additional details on benchmark construction and implementation. We first describe the extraction of different functional intermediate representations in Appx. A.1, including those used for representation selection and human-video pretraining. We then introduce our MPPI-based demonstration generation procedure in Appx. A.2, using the hitting function as a representative example. Finally, we summarize the training and implementation details of FUNCBRIDGE and all baselines in Appx. A.3.

Table 5: **Statistics of Functional Intermediate Representations for Hitting.** We report the number of tools, settings, initial configurations, samples per configuration, and total samples used for each representation.

Representation	# Tools	# Settings	# Initial Configs.	# Samples / Config.	# Total
Affordance Images	10	1	1	1	10
Human Video Prompts	7	3	5	5	525
Functional Videos	10	3	30	1	900
Object Masks	10	3	30	1	900
Keypoint Trajectories (Simulation)	10	3	30	1	900
Keypoint Trajectories (Real World)	7	2	1	20	280

A.1 EXTRACTION OF FUNCTIONAL INTERMEDIATE REPRESENTATIONS

As discussed in Sec. 3.2, we consider five types of functional intermediate representations: affordance images, human video prompts, functional videos, object masks, and 2D keypoint trajectories. In this section, we use the hitting function as a representative example to illustrate their construction and usage. Specifically, we describe how each representation is extracted and incorporated as an additional condition for the grounded execution policy π_{sys1}. The statistics of these representations are summarized in Tab. 5.

Affordance Images. We manually annotate the grasp point and hitting region for each tool, producing 10 affordance images in total. When evaluating functional intermediate representations in Sec. 3.2, we use the affordance image of the corresponding tool as an additional condition for the grounded execution policy π_{sys1}. Since this representation is time-invariant, the same image feature is shared across all time steps of a trajectory.

Human Video Prompts. As shown in Fig. 6, we collect human video prompts for all 7 tools. Following the simulation benchmark, each tool has 3 settings. For each setting, we define 5 initial positions and collect 5 videos per position, resulting in 525 human tool-use videos in total. When evaluating functional intermediate representations, we use the video prompt corresponding to the same tool, setting, and sample from Position 01 as an additional condition for the grounded execution policy. These videos provide visual demonstrations of how humans accomplish the same hitting function with different tools. We further validate that human video data can serve as effective action-free pretraining data for functional reasoning in Sec. 4.4.1.

Functional Videos and Object Masks. For each simulation demonstration, we use the corresponding ground-truth functional execution video as a temporally varying condition. To construct object masks, we remove the background from each frame while retaining the tool and target object,

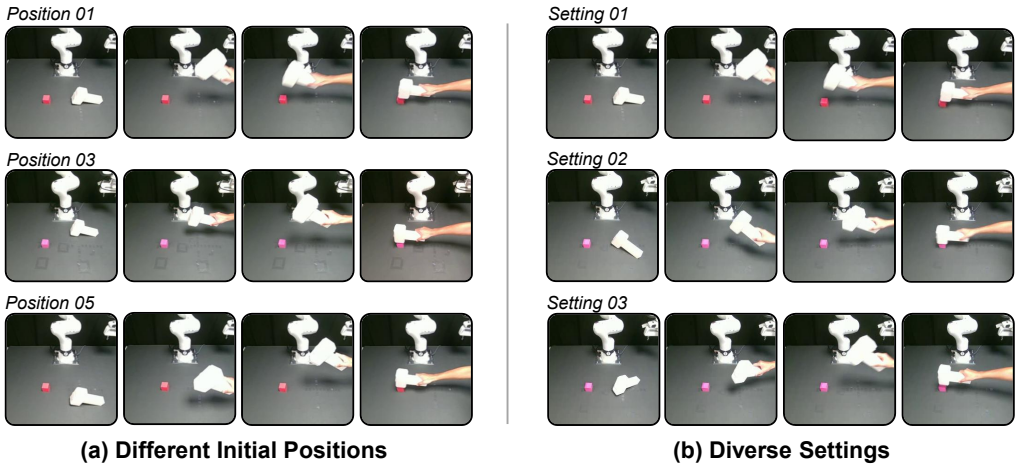


Figure 6: **Human Video Prompt Samples.** (a) Samples from different initial positions. (b) Samples from different settings.

thereby preserving object-centric geometry and motion while suppressing background appearance. Both representations are temporally aligned with the robot trajectory and used as sequential conditions for π_{sys1}. With 10 tools, 3 settings, and 30 randomized initial configurations per setting, we obtain 900 functional-video sequences and 900 object-mask sequences in total.

Keypoint Trajectories. We adopt 2D keypoint trajectories as the final functional intermediate representation and extract them for both simulation and real-world data. In simulation, the hitting and target points are directly obtained from the simulator. Given the initial hitting point on the tool, we use SAM2 (Ravi et al., 2025) to segment the corresponding tool mask and sample $N = 5$ additional tool keypoints with Farthest Point Sampling (FPS), ensuring spatial coverage of the tool geometry. We then track these sampled tool keypoints with CoTracker (Karaev et al., 2024) to obtain their 2D trajectories. Our simulation benchmark contains 10 tools with 3 settings each. For each setting, we randomly initialize 30 hitting and target positions, resulting in 900 simulation samples. For real-world data, we manually annotate the target and hitting points for each sample and obtain five additional tool keypoints using the same SAM2- and FPS-based procedure. Unlike simulation, where the hitting and target trajectories are available from the simulator, all 7 real-world keypoints are tracked by CoTracker. The real-world benchmark contains 7 tools, each with 2 hitting patterns and 1 initial position. We collect 20 samples for each initial position, resulting in 280 real-world samples.

A.2 MPPI-based Demonstration Generation

In the simulation benchmark, the action-labeled demonstrations are automatically generated rather than collected through teleoperation. We use a Model Predictive Path Integral (MPPI) planner with function-specific objectives. Taking the hitting function as an example, we adopt a four-stage rollout procedure, whose detailed reward terms are summarized in Tab. 6. The stages sequentially encourage the robot to lift the tool, move it toward the target, align the tool-specific hitting point with the target, and execute the final downward strike.

Besides the translational commands optimized by MPPI, we specify a target rotational pose at the end of each stage and generate the corresponding rotational commands through servo control. This helps the tool maintain an appropriate orientation during lifting, alignment, and hitting. Together, the stage-specific MPPI rewards and rotational servo control enable reliable automatic demonstration generation in simulation.

Table 6: **Main Reward Terms for MPPI-based Demonstration Generation.** Here, d_x, d_y, and d_{xy} denote the distances between the hitting point and the target along the x axis, the y axis, and the xy plane, respectively, while Δd_x, Δd_y, and Δd_{xy} denote their step-wise reductions. For the vertical direction, z_{lift} denotes the tool lifting height from its initial height, z_1 denotes the target lifting threshold in Stage 1, Δz_{lift} denotes the step-wise lifting progress, and e_z denotes the error between the current tool height and the target hovering height. Δz_{head} denotes the vertical displacement of the hitting point between consecutive steps, so $-\Delta z_{\text{head}}$ measures downward motion. The terms r_* reward the current state for satisfying the stage-specific objective, while r_{*_prog} rewards step-wise progress toward that objective.

Stage	Objective	Main Reward Terms
Stage 1	Lift the tool and align it with the target along the x and y axes.	$r_{\text{lift}} = 18.0 \min(z_{\text{lift}}, z_1)$, $r_{\text{lift_prog}} = 100.0 \min(\Delta z_{\text{lift}}, 0.02)$, $r_x = 22.0 \exp(-18.0 d_x)$, $r_{x_\text{prog}} = 120.0 \min(\Delta d_x, 0.02)$, $r_y = 8.0 \exp(-8.0 d_y)$, $r_{y_\text{prog}} = 35.0 \min(\Delta d_y, 0.02)$.
Stage 2	Move the tool toward the target while maintaining a suitable hovering height.	$r_{xy} = 14.0 \exp(-10.0 d_{xy})$, $r_x = 12.0 \exp(-18.0 d_x)$, $r_y = 8.0 \exp(-10.0 d_y)$, $r_z = 7.0 \exp(-18.0 e_z)$, $r_{xy_\text{prog}} = 95.0 \min(\Delta d_{xy}, 0.02)$, $r_{x_\text{prog}} = 80.0 \min(\Delta d_x, 0.02)$, $r_{y_\text{prog}} = 40.0 \min(\Delta d_y, 0.02)$.
Stage 3	Finely align the tool-specific hitting point above the target.	$r_{xy} = 14.0 \exp(-14.0 d_{xy})$, $r_x = 14.0 \exp(-24.0 d_x)$, $r_y = 10.0 \exp(-18.0 d_y)$, $r_{xy_\text{prog}} = 145.0 \min(\Delta d_{xy}, 0.018)$, $r_{x_\text{prog}} = 135.0 \min(\Delta d_x, 0.018)$, $r_{y_\text{prog}} = 110.0 \min(\Delta d_y, 0.018)$, $r_z = 7.0 \exp(-18.0 e_z)$.
Stage 4	Keep the hitting point close to the target and execute the downward strike.	$r_{xy} = 12.0 \exp(-16.0 d_{xy})$, $r_x = 14.0 \exp(-28.0 d_x)$, $r_y = 10.0 \exp(-18.0 d_y)$, $r_{xy_\text{prog}} = 120.0 \min(\Delta d_{xy}, 0.015)$, $r_{x_\text{prog}} = 100.0 \min(\Delta d_x, 0.015)$, $r_{y_\text{prog}} = 80.0 \min(\Delta d_y, 0.015)$, $r_{\text{hit}} = 150.0 \min(-\Delta z_{\text{head}}, 0.05)$,
Contact / Success	Encourage valid downward contact with the target.	$r_{\text{contact}} = 150.0$ if valid downward contact.

A.3 Training and Implementation Details

All models are trained on NVIDIA RTX 6000 Pro GPUs. For all methods, we use an action chunk size of 8, a history horizon of 8, and a batch size of 64. We build the simulation benchmark on Robomimic and conduct real-world experiments on a Franka robot platform.

For baseline methods, we train each policy for 120,000 steps using the corresponding action-labeled data. For FuncBridge, we follow the two-stage training procedure described in Sec. 3.3. We first pretrain the keypoint motion predictor π_{sys2} on independently collected action-free data for 240,000 steps. We then freeze the pretrained predictor and fine-tune the grounded execution policy π_{sys1} on the same action-labeled data used by the baselines for 120,000 steps.

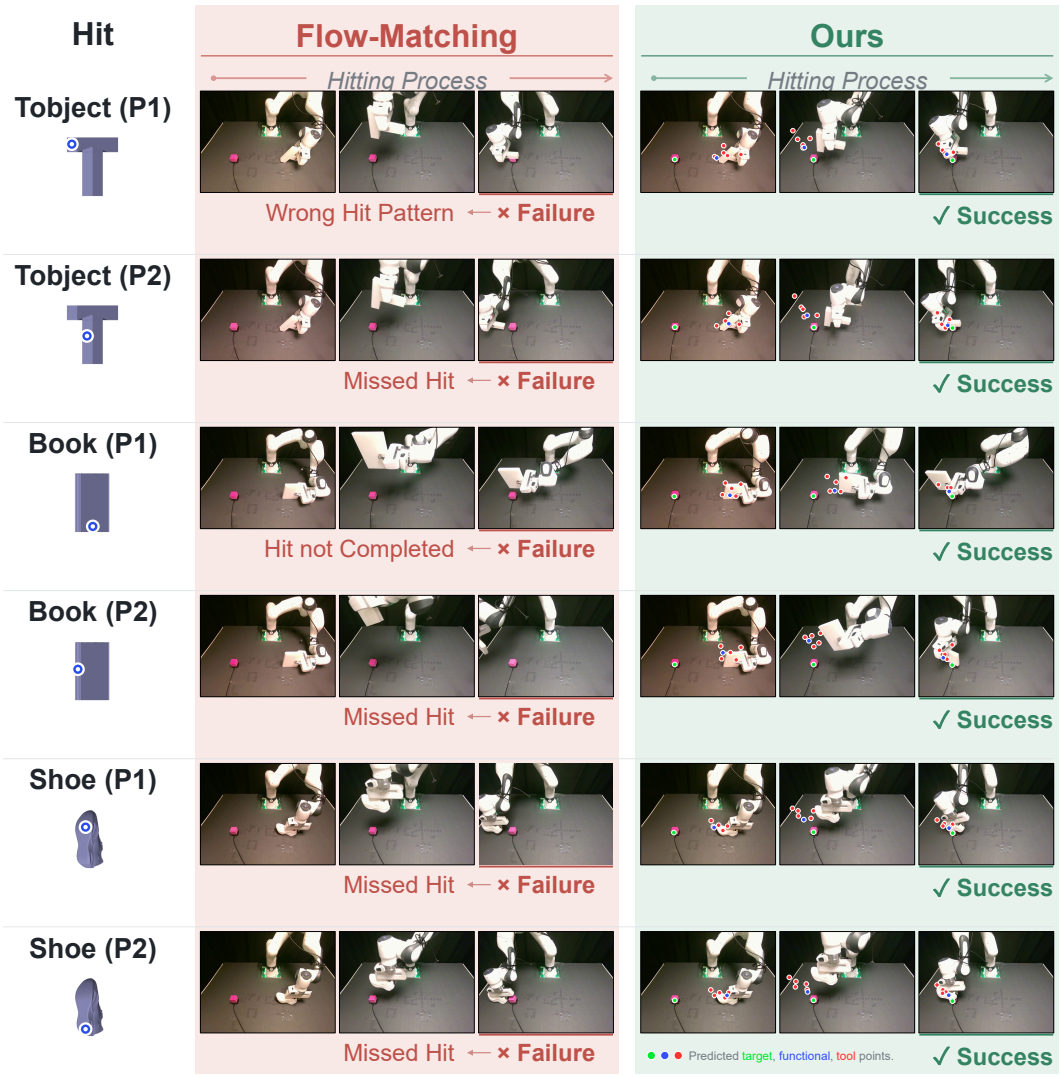


Figure 7: **Additional Real-World Qualitative Results for Hitting.** Failure cases of the FM baseline and successful demos of FUNCBRIDGE on unseen tools across different hitting patterns. The blue point indicates the reference hitting point for each pattern.

B ADDITIONAL EXPERIMENTAL RESULTS

This section provides additional qualitative comparisons across hitting, sweeping, and hooking in both the real world and simulation. These results complement the quantitative evaluations in the main paper by illustrating representative failure modes of the baselines and how FUNCBRIDGE addresses them through function-aware keypoint motion planning.

B.1 MORE QUALITATIVE RESULTS IN THE REAL WORLD

Figs. 7–9 provide additional real-world comparisons between the FM baseline and FUNCBRIDGE across hitting, sweeping, and hooking on unseen tools and different functional patterns. Across the three functions, FM exhibits representative failure modes including missed or incomplete interactions and incorrect functional motion patterns, showing that an end-to-end policy struggles to adapt both the functional region and the required motion to novel tool geometries. By contrast, FUNCBRIDGE uses predicted keypoint-based functional plans to guide functional-region alignment and motion execution, leading to successful interactions across the shown hitting, sweeping, and

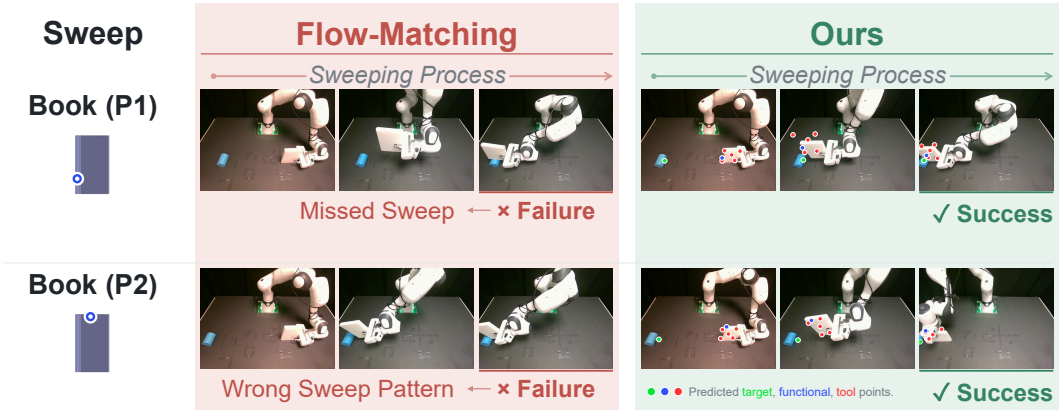


Figure 8: **Additional Real-World Qualitative Results for Sweeping.** Failure cases of the FM baseline and successful demos of FUNCBRIDGE on unseen tools across different sweeping patterns. The blue point indicates the reference sweeping point for each pattern.

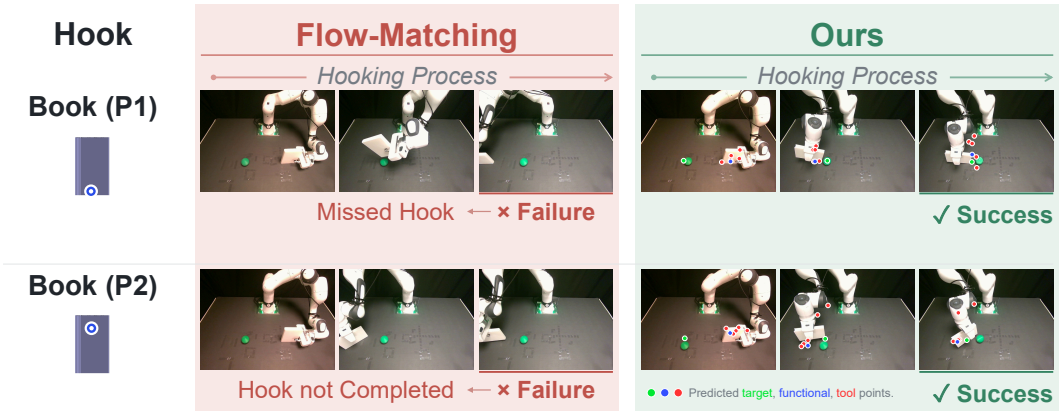


Figure 9: **Additional Real-World Qualitative Results for Hooking.** Failure cases of the FM baseline and successful demos of FUNCBRIDGE on unseen tools across different hooking patterns. The blue point indicates the reference hooking point for each pattern.

hooking cases. These results further demonstrate the benefit of an explicit function-aware intermediate representation for functional generalization.

B.2 More Qualitative Results in Simulation

Figs. 10–12 provide additional simulation comparisons among FM, ATM, and FUNCBRIDGE across hitting, sweeping, and hooking on unseen tools. FM exhibits representative failures such as passing over the target, imprecise interactions, and tool dropping. Although ATM also conditions on keypoint trajectories, its generic trajectories do not explicitly encode function-specific motion intent and can therefore exhibit similar failures. In contrast, FUNCBRIDGE predicts function-aware keypoint trajectories that explicitly guide how the functional region should move toward and interact with the target, enabling more reliable execution across the three functions.

Please refer to our supplementary videos for complete rollouts of both simulation and real-world experiments.

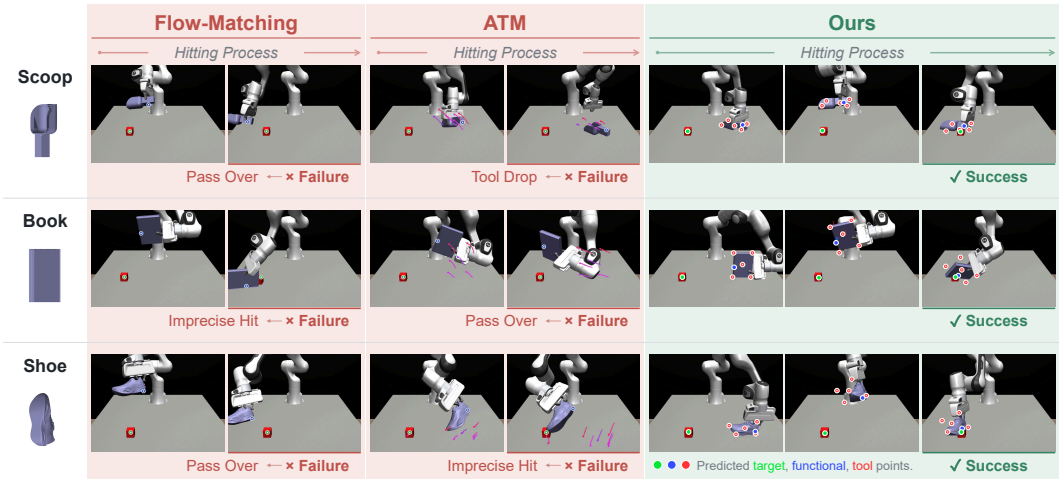


Figure 10: **Additional Simulation Qualitative Results for Hitting.** Failure cases of FM and ATM and successful rollouts of FUNCBRIDGE on unseen tools.

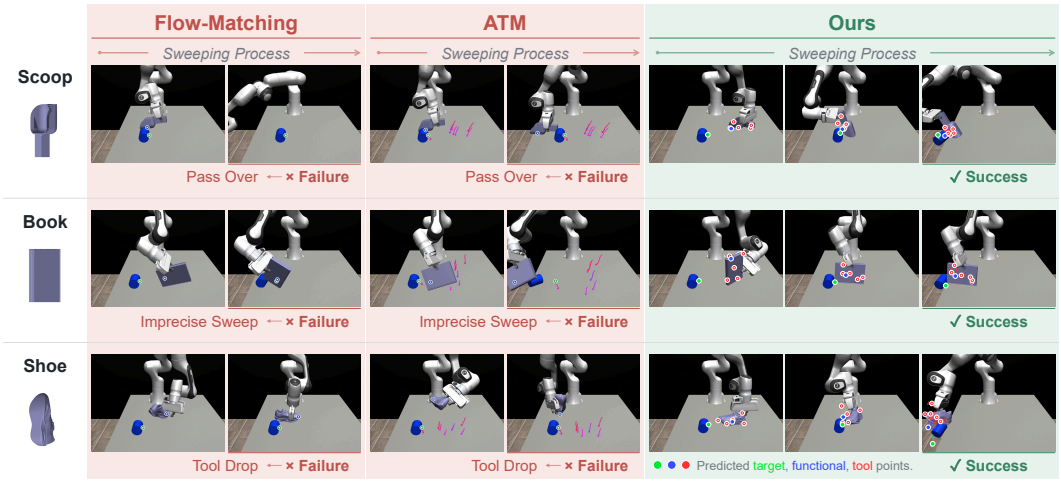


Figure 11: **Additional Simulation Qualitative Results for Sweeping.** Failure cases of FM and ATM and successful rollouts of FUNCBRIDGE on unseen tools.

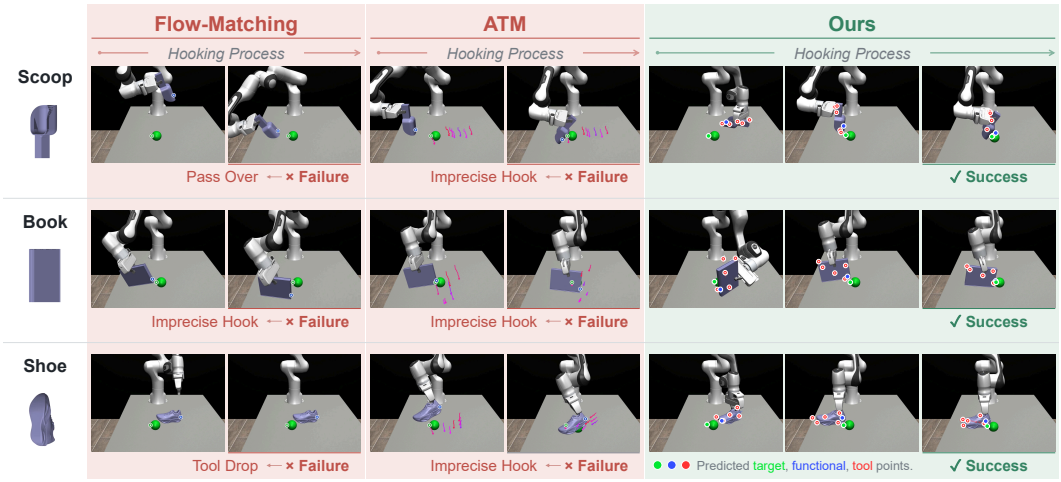


Figure 12: **Additional Simulation Qualitative Results for Hooking.** Failure cases of FM and ATM and successful rollouts of FUNCBRIDGE on unseen tools.