# BimanualShift: Arm-Conditioned Residual Skill Transfer from Unimanual Policies to Bimanual Manipulation

Yechen Fan, *Graduate Student Member, IEEE,* Jinhua Ye, *Senior Member, IEEE,* Xianyou Ji, Chenyang Song, Haibin Wu, Gengfeng Zheng, *Senior Member, IEEE,* and Jiafu Wan, *Fellow, IEEE*

***Abstract*—Pretrained unimanual policies encode rich manipulation priors, but reusing them for bimanual manipulation remains difficult due to shared-workspace perceptual interference, coupled dual-arm coordination, and scarce high-quality bimanual demonstrations. We propose BimanualShift, an arm-conditioned residual skill transfer framework that adapts frozen unimanual policies to bimanual tasks without end-to-end retraining. Rather than learning a full bimanual policy from scratch, BimanualShift reformulates bimanual control as residual coordination over frozen unimanual priors. Arm-conditioned visual projection first aligns the shared bimanual scene with the local observation distribution expected by each unimanual policy, reducing cross-arm perceptual interference at the input level. Bimanual coordination is then re-parameterized as low-dimensional residual modulation over a learnable skill basis, where the frozen policy provides the base manipulation prior and the lightweight residual branch accounts for synchronization, spatial avoidance, and contact compensation. For long-horizon execution, successful prior executions are not directly replayed; instead, they are retrieved as residual contexts to correct local execution deviations while keeping the pretrained policies frozen. Experiments in RLBench2 and on a real dual-arm platform show that BimanualShift improves data efficiency, generalization, and long-horizon robustness under few bimanual demonstrations.**

***Index Terms*—Bimanual manipulation, residual skill transfer, frozen policy priors, data-efficient learning.**

## I. Introduction

BIMANUAL manipulation is a key direction for general embodied intelligence, requiring proficient coordination between two hands. Compared with unimanual systems, such tasks exhibit stronger coupling and significantly higher degrees of freedom, perceptual load, and control complexity, making policy learning substantially more challenging.

While unimanual manipulation has benefited from large-scale pretrained policies, transferring such priors to bimanual manipulation remains non-trivial. End-to-end bimanual policies, including ACT [1], Diffusion Policy [2], [3], and recent Vision–Language–Action models [4], [5], [6], [7], [8], can model complex behaviors but usually require large amounts of high-quality bimanual demonstrations to cover increased action dimensionality, contact interactions, and temporal synchronization. LLM-based approaches [12], [13] provide stronger task-level reasoning, but their symbolic plans remain difficult to ground into continuous, contact-rich dual-arm control.

A more data-efficient direction is to reuse pretrained unimanual policies as manipulation priors. Recent transfer-based methods such as AnyBimanual [28] show the promise of unimanual-to-bimanual transfer through visual alignment and skill scheduling. However, bimanual manipulation involves complex coordination patterns, role assignments, and physical interactions [9], making direct adaptation to shared workspaces difficult. Holistic or weakly separated inputs can cause target-assignment ambiguity, while independently executed unimanual actions lack the coordination needed for spatial avoidance, force interaction, and temporal synchronization, especially in contact-rich or deformable-object tasks [11]. Existing bimanual imitation-learning and transfer-based methods still face challenges in data efficiency, robustness, and long-horizon generalization [10], and methods such as PerAct$^2$ [16] and VoxAct-B [17] lack explicit reuse of successful prior executions. These limitations motivate treating unimanual-to-bimanual transfer as residual coordination over frozen unimanual priors rather than direct policy reuse.

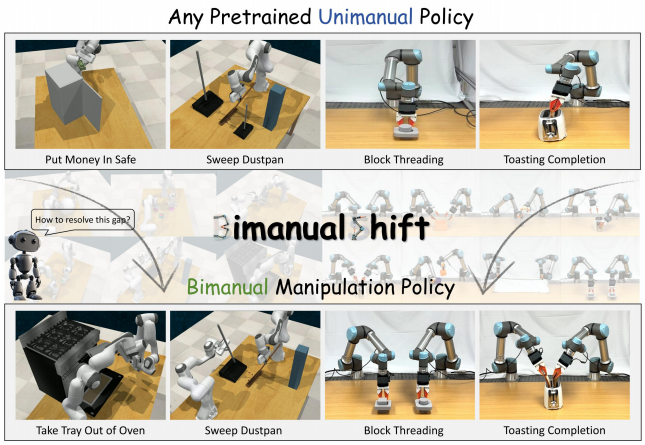


Fig. 1. Overview of BimanualShift. The framework adapts frozen pretrained unimanual policy priors to bimanual manipulation by combining arm-specific perception, residual coordination, and experience-conditioned refinement.

To address these challenges, we propose BimanualShift, an arm-conditioned residual skill transfer framework for adapting frozen unimanual policies to bimanual manipulation (see Fig. 1). Rather than treating bimanual manipulation as a new policy-learning problem, BimanualShift recasts it as residual coordination over frozen unimanual priors. Arm-conditioned visual projection aligns the shared workspace with the observation distribution expected by each unimanual policy, reducing cross-arm perceptual interference before inference. Residual skill modulation then re-parameterizes dual-arm coordination as lightweight residual actions over frozen policy outputs, compensating for synchronization, spatial avoidance, and contact variations. Retrieval-conditioned residual refinement further uses successful prior executions as residual contexts rather than replay trajectories, enabling local correction of accumu-

lated errors while keeping the pretrained policies unchanged.

In summary, the main contributions of this study are as follows:

- We formulate unimanual-to-bimanual transfer as an arm-conditioned residual skill transfer problem, where frozen unimanual policies are reused under arm-specific observations and only lightweight residual coordination modules are learned.
- We propose an arm-conditioned visual projection mechanism that converts shared bimanual observations into policy-compatible unimanual inputs, reducing perceptual interference before policy inference.
- We introduce residual skill modulation with retrieval-conditioned refinement, where dynamic skill residuals coordinate two frozen unimanual policies and successful prior executions are reused as residual contexts for long-horizon adaptation.

## II. Related Work

### A. Foundation Models for Robotic Manipulation

Large-scale pretrained models and imitation policies have substantially advanced robotic manipulation. End-to-end imitation learning methods, such as ACT [1] and Diffusion Policy [2], directly map visual observations to continuous actions and have shown strong performance in fine-grained manipulation. Vision–Language–Action models, including the $\pi$ series [4], [5], [6], Octo [7], and OpenVLA [8], further improve generalization by scaling across diverse robot embodiments and language-conditioned tasks. Video-generation and predictive-representation methods, such as Gen2Act [18], VPP [19], VidMan [20], and Dreamitate [21], exploit future visual prediction or implicit dynamics to support planning and policy learning. Although these methods provide powerful policy representations, they usually rely on large-scale demonstrations or end-to-end training. This makes them less practical for few-shot bimanual manipulation, where demonstrations are costly and failures often arise from coupled perception, contact interaction, and temporal synchronization. In contrast, BimanualShift reuses frozen unimanual policy priors and learns lightweight residual coordination, avoiding full bimanual policy training from scratch.

### B. Skill Learning and Transfer in Robotics

Reusing learned skills across tasks, environments, and embodiments is crucial for data-efficient robot learning. Prior works study domain adaptation and fine-tuning for robotic transfer [22], [23], cross-domain correspondences through visual imitation or hierarchical policies [24], [25], and latent-space alignment for cross-embodiment skill transfer [26], [27]. For bimanual manipulation, recent studies highlight the complexity of dual-arm coordination and the limitations of imitation learning under data scarcity [9], [10]. The most related method, AnyBimanual [28], transfers unimanual policies to bimanual tasks through visual alignment and dynamic skill scheduling, demonstrating the promise of unimanual-to-bimanual transfer. However, existing transfer methods still under-address explicit arm-specific perception, residual coordination over frozen policy priors, and long-horizon reuse of successful executions. BimanualShift addresses these gaps by combining arm-conditioned visual projection, residual skill modulation, and retrieval-conditioned residual refinement.

## III. Methodology

As illustrated in Fig. 2, BimanualShift formulates unimanual-to-bimanual transfer as arm-conditioned residual skill transfer. It preserves frozen unimanual policy priors [15] and learns only lightweight residual coordination modules. The framework consists of arm-conditioned visual projection, residual skill modulation, and retrieval-conditioned residual refinement.

### A. Arm-Conditioned Visual Projection

As shown in Fig. 2, the first stage of BimanualShift converts a shared bimanual workspace into policy-compatible unimanual observations. Directly feeding the global observation $o_t$ to frozen unimanual policies can introduce perceptual interference, especially in close-proximity or arm-crossing scenarios where both arms, task objects, and background clutter are entangled [9]. Unlike soft attention mechanisms that require sufficient task data to learn stable separation [28], we define an explicit arm-conditioned visual projection $\mathcal{P}_k$, where $k \in \{L, R\}$ denotes the left or right arm.

The projection maps the shared observation, language instruction, and proprioceptive state into an arm-specific masked observation:

$$\tilde{o}_{k,t} = \mathcal{P}_k(o_t, l, p_t) = V_t \odot \mathbf{M}_{3D,k,t}, \quad k \in \{L, R\} \tag{1}$$

where $V_t \in \mathbb{R}^{H \times W \times D}$ is the voxelized observation, $\mathbf{M}_{3D,k,t}$ is the arm-specific semantic mask, and $\odot$ denotes element-wise multiplication. Eq. (1) suppresses the opposite arm and task-irrelevant regions before policy inference, making the input closer to the unimanual observation distribution on which $\pi_{\text{uni}}$ was pretrained.

In implementation, $\mathcal{P}_k$ is realized by a semantic mask generation and back-projection pipeline based on Grounded-SAM [29], [30]. Given the instruction $l$, we extract the target object noun $\chi_{\text{goal}}$ and construct the text prompt $[\text{robotic arm}, \chi_{\text{goal}}]$. Grounded-SAM detects robotic-arm instances and the task target in image space. Since the robot bases are fixed, we use two calibrated image-plane anchors, $\mathbf{b}_L$ and $\mathbf{b}_R$, corresponding to the projected base centers of the left and right arms. For each arm $k$, the corresponding body mask is selected from the candidate arm masks by nearest-anchor assignment:

$$\mathbf{M}_{\text{body},k} = \underset{\mathbf{M} \in \mathcal{S}_{\text{arm}}}{\arg\min} \left\| \text{Centroid}(\mathbf{M}) - \mathbf{b}_k \right\|_2 \tag{2}$$

where $\mathcal{S}_{\text{arm}}$ denotes the set of candidate masks labeled as robotic arms. The final 2D mask for arm $k$ is obtained by combining the selected body mask and the target-object mask:

$$\mathbf{M}_{2D,k} = \mathbf{M}_{\text{body},k} \cup \mathbf{M}_{\text{goal}} \tag{3}$$

The 2D mask $\mathbf{M}_{2D,k}$ is then back-projected into the voxel grid using the depth map $\mathcal{D}_t$ and camera intrinsics $\mathbf{K}$, yielding

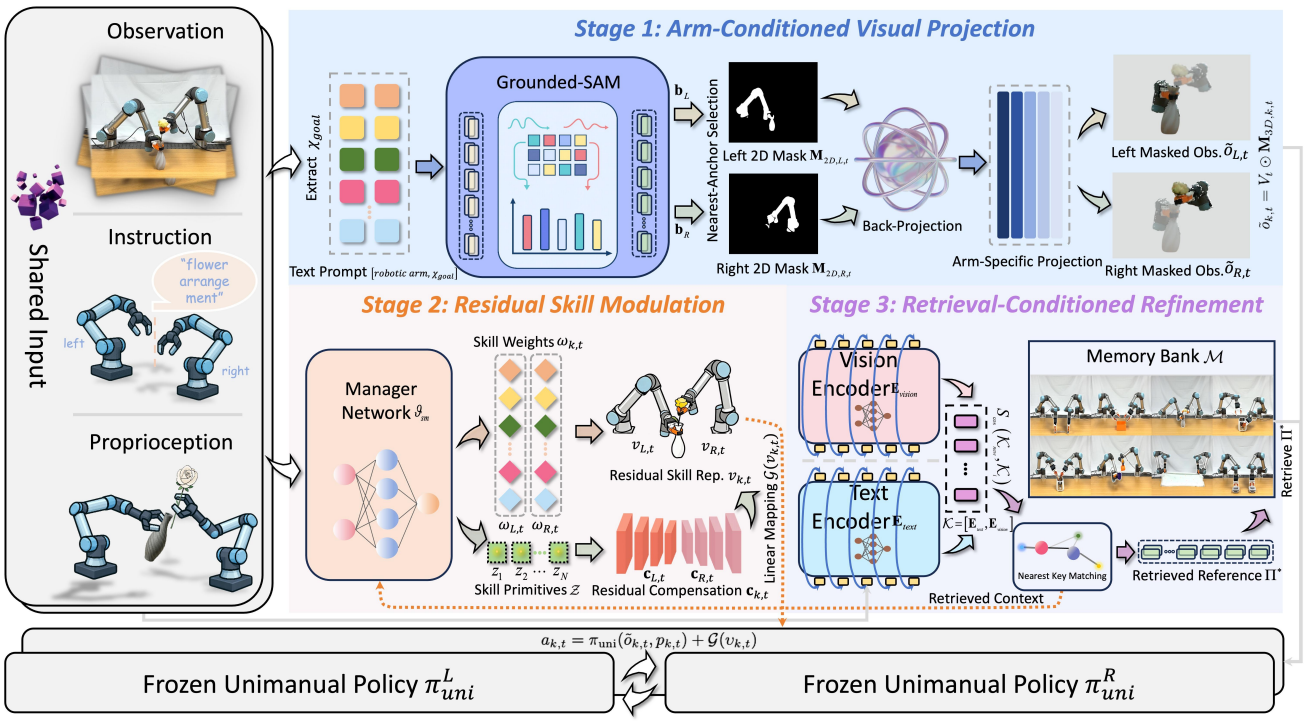


Fig. 2. Technical framework of BimanualShift. The framework treats bimanual manipulation as arm-conditioned residual skill transfer over frozen unimanual policy priors. Arm-conditioned visual projection converts the shared workspace into policy-compatible arm-specific observations, residual skill modulation re-parameterizes dual-arm coordination as lightweight residual actions added to frozen policy outputs, and retrieval-conditioned refinement reuses successful executions as residual contexts for long-horizon correction rather than direct trajectory replay.

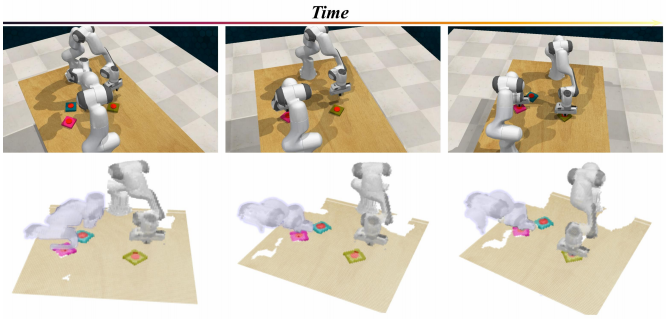


Fig. 3. Semantic voxel scenes for Push Two Buttons. The top row shows original RLBench viewpoints, and the bottom row shows arm-conditioned visual projection results that filter the interfering arm and background clutter while preserving the acting arm and task-relevant workspace.

the 3D mask $\mathbf{M}_{3D,k,t}$ used in Eq. (1). As visualized in Fig. 3, this projection filters the interfering arm and background clutter while retaining the acting arm, target object, and local support surface. Consequently, bimanual perception is converted into two decoupled unimanual observation streams, allowing the residual skill modulation stage in Section III-B to focus on coordination rather than resolving visual interference.

### B. Residual Skill Modulation

After obtaining arm-specific observations from Eq. (1), BimanualShift coordinates the two frozen unimanual policies through residual skill modulation. As illustrated in Fig. 4, instead of learning a bimanual policy from scratch [1], the Action Generator predicts a low-dimensional residual in a learnable skill basis and adds the projected residual to the frozen unimanual policy output. For each arm $k \in \{L, R\}$, the final action is defined as:

$$a_{k,t} = \pi_{\text{uni}}(\tilde{o}_{k,t}, p_{k,t}) + \mathcal{G}(v_{k,t}) \tag{4}$$

where $\pi_{\text{uni}}$ is the frozen pretrained unimanual policy, $\tilde{o}_{k,t}$ is the arm-specific observation from Eq. (1), $p_{k,t}$ is the proprioceptive state of arm $k$, and $\mathcal{G}(\cdot)$ projects the skill representation into the action residual space.

To construct the residual skill representation, we define a learnable skill primitive matrix:

$$\mathcal{Z} = \{\boldsymbol{z}_1, \boldsymbol{z}_2, \ldots, \boldsymbol{z}_N\}, \quad \boldsymbol{z}_i \in \mathbb{R}^{d_{\text{emb}}} \tag{5}$$

where each primitive $\boldsymbol{z}_i$ represents an atomic motion basis [27]. To provide semantic grounding, $\mathcal{Z}$ is initialized from embeddings of atomic action verbs extracted by a frozen CLIP text encoder. In all experiments, we set $N = 16$.

Within the residual skill modulation stage, we implement the Action Generator as a lightweight Manager Network $\vartheta_{\text{sm}}$. Given the arm-specific observation $\tilde{o}_{k,t}$, the full bimanual proprioceptive state $p_t$, and the language instruction $l$, the Manager predicts skill weights $\boldsymbol{\omega}_{k,t}$ and a latent compensation vector $\mathbf{h}_{k,t}$:

$$(\boldsymbol{\omega}_{k,t}, \mathbf{h}_{k,t}) = \vartheta_{\text{sm}}(\tilde{o}_{k,t}, p_t, l), \quad \sum_{i=1}^{N} \omega_{i,k,t} = 1 \tag{6}$$

where $\boldsymbol{\omega}_{k,t}$ is normalized by a softmax layer and $\mathbf{h}_{k,t} \in \mathbb{R}^{d_{\text{emb}}}$ denotes the latent compensation vector. $p_{k,t}$ denotes the proprioceptive state of arm $k$, while $p_t$ denotes the full bimanual proprioceptive state used by the residual modulator.

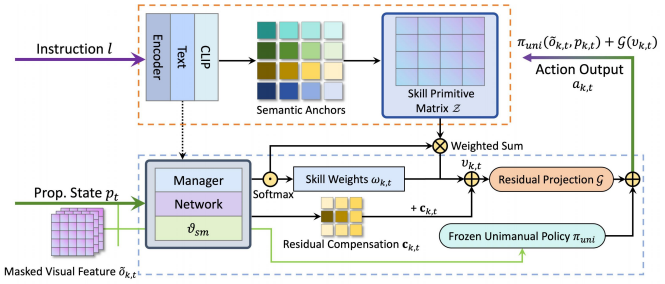


Fig. 4. Residual skill modulation via the Action Generator. The Manager Network $\vartheta_{\text{sm}}$ schedules skill weights $\boldsymbol{\omega}_{k,t}$ and latent compensation vectors $\mathbf{h}_{k,t}$ from arm-specific visual features, proprioception, and language. The projected residual $\mathcal{G}(\upsilon_{k,t})$ is added to the frozen unimanual policy output to produce the coordinated action $a_{k,t}$.

The residual skill representation is then computed as:

$$\upsilon_{k,t} = \sum_{i=1}^{N} \omega_{i,k,t} \boldsymbol{z}_i + \mathbf{h}_{k,t} \tag{7}$$

Here, the weighted skill primitives provide phase-dependent motion structure, while $\mathbf{h}_{k,t}$ compensates for latent deviations induced by local geometric mismatch, contact variation, and inter-arm coordination errors [11]. Through Eq. (4), BimanualShift turns bimanual coordination into residual correction over pretrained unimanual priors, rather than full action generation.

### C. Retrieval-Conditioned Residual Refinement

Although residual skill modulation in Section III-B enables lightweight coordination, long-horizon bimanual tasks can still suffer from compounding errors [10]. To improve execution stability, BimanualShift introduces retrieval-conditioned residual refinement, which is implemented with a memory bank $\mathcal{M}$ of successful executions. The memory bank serves as a residual-context repository: retrieved executions provide temporal references for correction [24] rather than directly replayed action sequences. This design allows the policy to reuse task-level experience while still adapting actions at execution time through the residual branch. For fair evaluation, $\mathcal{M}$ is constructed from training demonstrations and successful refinement rollouts, and remains fixed during evaluation.

Each memory entry stores a task key and a verified successful trajectory, denoted as $(\mathcal{K}_i, \Pi_i^*)$. Since the same instruction may correspond to different initial scenes, the task key jointly encodes the language instruction and the initial observation:

$$\mathcal{K}_i = [\mathbf{E}_{\text{text}}(l_i); \mathbf{E}_{\text{vision}}(o_0^i)] \tag{8}$$

where $\mathbf{E}_{\text{text}}$ and $\mathbf{E}_{\text{vision}}$ are pretrained text and vision encoders, respectively, and $[\cdot;\cdot]$ denotes feature concatenation. Given a new task $(l_{\text{new}}, o_0)$, BimanualShift computes its key $\mathcal{K}_{\text{new}}$ and retrieves the most similar memory entry:

$$i^* = \underset{i \in \{1,\ldots,|\mathcal{M}|\}}{\arg\max}\ S_{\cos}(\mathcal{K}_{\text{new}}, \mathcal{K}_i) \tag{9}$$

where $S_{\cos}(\cdot,\cdot)$ denotes cosine similarity.

The retrieved trajectory $\Pi_{i^*}^*$ is used as a temporal reference for the residual modulator, not as a fixed replayed action sequence. If the retrieval confidence is sufficiently high, the Action Generator adapts the retrieved execution by updating the residual representation in Eq. (7) through state-dependent latent compensation vectors $\mathbf{h}_{k,t}$. If the retrieval is ambiguous, the model relies on the current observation and instruction to generate residuals without using a retrieved reference.

When execution deviates from the retrieved reference, BimanualShift performs local residual refinement around the failure timestep $t_f$. In practice, corrective actions can be obtained from successful retry trajectories or human-provided corrective demonstrations. Given a corrective action $a_{k,t_f}^{\text{corr}}$ and the original frozen-prior action $a_{k,t_f}^{\text{base}}$, the target residual is defined as:

$$\mathbf{r}_{k,t_f}^* = a_{k,t_f}^{\text{corr}} - a_{k,t_f}^{\text{base}} \tag{10}$$

The residual branch is then refined by minimizing:

$$\mathcal{L}_{\text{corr}} = \left\| \mathcal{G}(\upsilon_{k,t_f}) - \mathbf{r}_{k,t_f}^* \right\|_2^2. \tag{11}$$

Only the residual branch is updated during this refinement, while the pretrained unimanual policy $\pi_{\text{uni}}$ remains frozen. Successful executions are inserted into $\mathcal{M}$, allowing BimanualShift to accumulate reusable residual coordination contexts for long-horizon tasks [23].

### D. Training Objectives

The training objective follows the arm-conditioned residual transfer formulation. The arm-conditioned visual projection module and the pretrained unimanual policy $\pi_{\text{uni}}$ are kept frozen, while only the Manager Network, i.e., the residual skill modulator $\vartheta_{\text{sm}}$, and the projection layer $\mathcal{G}$ are optimized. Given a small demonstration dataset $\mathcal{D}$, we train the residual branch by behavior cloning over the final bimanual action.

We decompose the target action into translation $a_{\text{trans}}$ and rotation $\theta_{\text{axis}}$. The behavior cloning loss is defined as:

$$\mathcal{L}_{\text{BC}}(\vartheta_{\text{sm}}) = \mathbb{E}_{(s,a)\sim\mathcal{D}}\left[\|a_{\text{trans}} - \hat{a}_{\text{trans}}\|_2^2 + \beta\|\theta_{\text{axis}} - \hat{\theta}_{\text{axis}}\|_2^2\right] \tag{12}$$

where $\hat{a}_{\text{trans}}$ and $\hat{\theta}_{\text{axis}}$ denote the expert translation and rotation targets, and $\beta$ balances the two terms. For retrieval-conditioned refinement, failure cases provide local action-space residual targets as defined in Eq. (10), and the residual branch is optimized using the correction loss in Eq. (11). The final objective is:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{BC}} + \lambda_{\text{corr}}\mathcal{L}_{\text{corr}} \tag{13}$$

This objective updates only the lightweight residual coordination components, while the pretrained unimanual policy remains frozen throughout training and refinement.

## IV. Experimental Evaluation and Results

### A. Simulation in RLBench2

We evaluate BimanualShift on six representative tasks from RLBench2 [16], a bimanual extension of RLBench [31], using two Franka Panda arms and six noise-free $256 \times 256$ RGB-D cameras.

***Baselines:*** We compare BimanualShift with representative bimanual manipulation baselines, including voxel-based

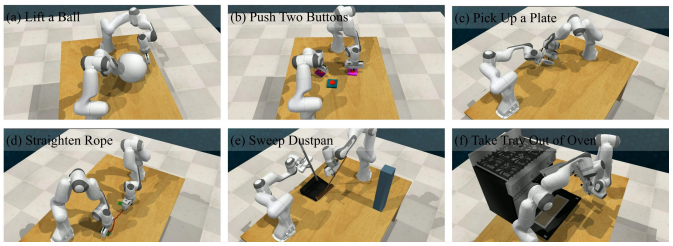


Fig. 5. Bimanual manipulation tasks in RLBench2.

methods PerAct$^2$ [16] and VoxAct-B [17], the trajectory-prediction method ACT [1], Leader-Follower variants RVT-LF [14] and PerAct-LF [15], and AnyBimanual [28], the closest unimanual-to-bimanual transfer baseline.

***Simulation Setup:*** To reflect real-world data-collection costs, we split the six tasks into complex 30-demo tasks (Take Tray Out of Oven, Straighten Rope, Sweep Dustpan) and simpler 100-demo tasks (Lift a Ball, Push Two Buttons, Pick Up a Plate) (see Fig. 5). All tasks include randomization in object color, size, quantity, placement, and category (see Fig. 6). We evaluate each method over 100 episodes per task on a single NVIDIA RTX 4090 GPU and report average success rates. All unimanual policies are initialized from checkpoints pretrained on 18 RLBench tasks [31].

***Implementation Details:*** Baselines are trained on two NVIDIA RTX A10 GPUs. BimanualShift freezes the pretrained unimanual backbones and fine-tunes only the lightweight adaptation modules. For BimanualShift-PerAct, the backbone is initialized from the public PerAct checkpoint pretrained on 18 RLBench tasks. We use LAMB with a learning rate of $5 \times 10^{-4}$, batch size 4, SE(3) augmentation, sparse keyframes, and motion planning for full-trajectory execution. No evaluation episode is used for online memory update or corrective supervision.

***Ablation Setup:*** To analyze the contribution of each component, we further evaluate three ablated variants of BimanualShift-PerAct: w/o Visual Projection, w/o Residual Modulation, and w/o Retrieval Refinement. All ablation variants use the same pretrained frozen PerAct backbone and are trained with the same demonstrations as the full model.

The results are summarized in Table I. Under the 30-demonstration complex-task setting, BimanualShift-PerAct achieves the highest average success rate of 31.7%, outperforming PerAct$^2$ (17.7%), VoxAct-B (14.7%), PerAct-LF (11.0%), and AnyBimanual-PerAct (22.0%). Under the 100-demonstration setting, it reaches the best average success rate of 50.7%, exceeding PerAct$^2$ (29.3%), VoxAct-B (22.7%), and AnyBimanual-PerAct (42.3%). Although PerAct$^2$ and AnyBimanual-PerAct perform better on Take Tray Out of Oven and Lift a Ball, respectively, BimanualShift-PerAct achieves stronger overall performance on tasks involving deformable-object interaction, close-proximity coordination, or target disambiguation, such as Straighten Rope, Sweep Dustpan, Push Two Buttons, and Pick Up a Plate. The ablation results show that removing Visual Projection, Residual Modulation, or Retrieval Refinement reduces the average success rates to 20.7%/33.0%, 17.0%/32.0%, and 23.0%/43.3%, respectively, confirming their complementary contributions. As shown in Fig. 7, BimanualShift reduces target-assignment ambiguity by better separating arm-specific workspaces.

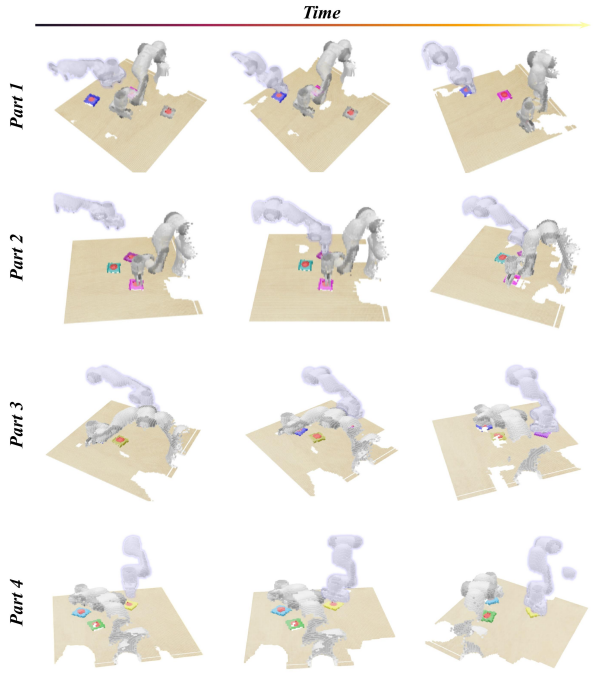


Fig. 6. Randomization examples of Push Two Buttons. Object colors and target positions vary across executions, and the shown images correspond to the generated left- and right-arm masks.

### B. Real-World Setup

We evaluate BimanualShift on a real dual-UR5e platform with PGI-140 grippers, custom TPU 95A compliant fingertips, and a front-mounted Intel RealSense L515 camera (see Fig. 8). The system runs on Ubuntu 20.04 with ROS1. Demonstrations are collected via GELLO teleoperation [32], which maps human bimanual motions to 6-DoF end-effector poses. After hand–eye calibration with easy_handeye and ArUco markers, RGB images are cropped to $256 \times 256$ and keyframes are extracted. At deployment, BimanualShift predicts action chunks $a_{t:t+H}$ from real-time observations and executes target poses using MoveIt.

### C. Core Performance and Robustness in Real-World

To evaluate BimanualShift across different backbones, we instantiate it with PerAct and Diffusion Policy (DP) [2], [33] pretrained only on separate unimanual data, and compare with an end-to-end DP trained from scratch on the same 15 bimanual demonstrations. We collect 15 demos per task for eight real-world tasks: Flower Arrangement, Pouring Water, Toasting Completion, Vegetable Sorting, Quilt Folding, Cable Routing, Toaster Activation, and Block Threading. BimanualShift-DP and BimanualShift-PerAct are fine-tuned on this data, while the end-to-end DP baseline is trained from scratch on the same dataset. All methods are evaluated on the real robotic platform shown in Fig. 9.

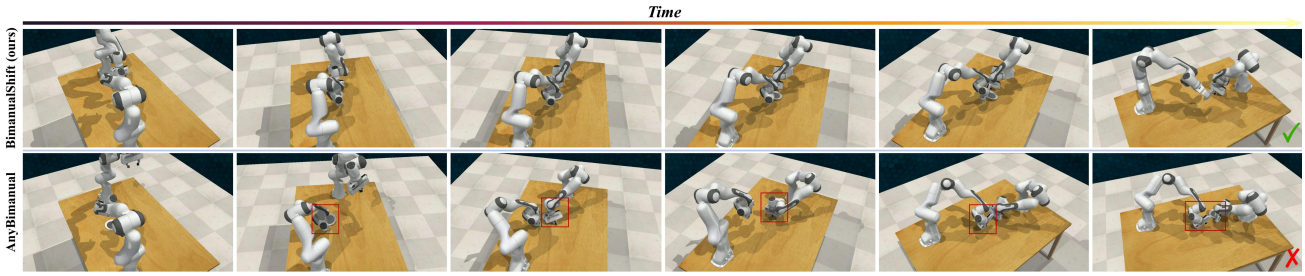


Fig. 7. Comparison between BimanualShift and AnyBimanual on the Pick Up a Plate task.

TABLE I
SIMULATION RESULTS AND COMPONENT ABLATIONS ON RLBENCH2

| Method | (i) Complex Tasks (30) | | | | (ii) Single-Action Tasks (100) | | | |
|---|---|---|---|---|---|---|---|---|
| | Take Tray Out of Oven | Straighten Rope | Sweep Dustpan | Average | Lift a Ball | Push Two Buttons | Pick Up a Plate | Average |
| RVT-LF [14] | 2.0 | 4.0 | 18.0 | 8.0 ± 1.6 | 17.0 | 39.0 | 3.0 | 19.7 ± 2.3 |
| AnyBimanual-RVT [28] | 4.0 | 6.0 | 24.0 | 11.3 ± 1.8 | 20.0 | 42.0 | 4.0 | 22.0 ± 2.4 |
| **BimanualShift-RVT (Ours)** | **6.0** | **9.0** | **30.0** | **15.0 ± 2.0** | **23.0** | **47.0** | 4.0 | **24.7 ± 2.5** |
| ACT [1] | 0.0 | 4.0 | 8.0 | 4.0 ± 1.2 | 36.0 | 4.0 | 0.0 | 13.3 ± 2.0 |
| VoxAct-B [17] | 4.0 | 12.0 | 28.0 | 14.7 ± 2.1 | 48.0 | 15.0 | 5.0 | 22.7 ± 2.4 |
| PerAct-LF [15] | 3.0 | 8.0 | 22.0 | 11.0 ± 1.9 | 40.0 | 10.0 | 2.0 | 17.3 ± 2.2 |
| PerAct$^2$ [16] | **13.0** | 10.0 | 30.0 | 17.7 ± 2.2 | 43.0 | 41.0 | 4.0 | 29.3 ± 2.7 |
| AnyBimanual-PerAct [28] | 9.0 | 16.0 | 41.0 | 22.0 ± 2.4 | **61.0** | 58.0 | 8.0 | 42.3 ± 3.0 |
| w/o Visual Projection | 7.0 | 16.0 | 39.0 | 20.7 ± 2.4 | 44.0 | 48.0 | 7.0 | 33.0 ± 2.9 |
| w/o Residual Modulation | 5.0 | 12.0 | 34.0 | 17.0 ± 2.2 | 38.0 | 52.0 | 6.0 | 32.0 ± 2.8 |
| w/o Retrieval Refinement | 8.0 | 18.0 | 43.0 | 23.0 ± 2.5 | 52.0 | 69.0 | 9.0 | 43.3 ± 3.1 |
| **BimanualShift-PerAct (Ours)** | 11.0 | **27.0** | **57.0** | **31.7 ± 2.8** | 60.0 | **80.0** | **12.0** | **50.7 ± 3.3** |

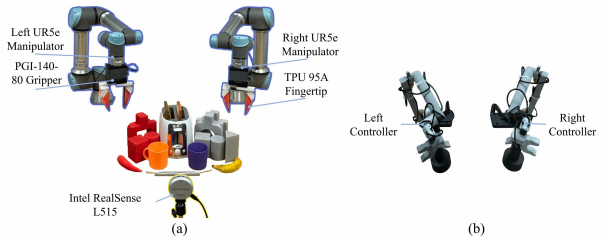


Fig. 8. Real-world setup. (a) Dual-UR5e platform with an Intel RealSense L515 camera. (b) GELLO teleoperation device.

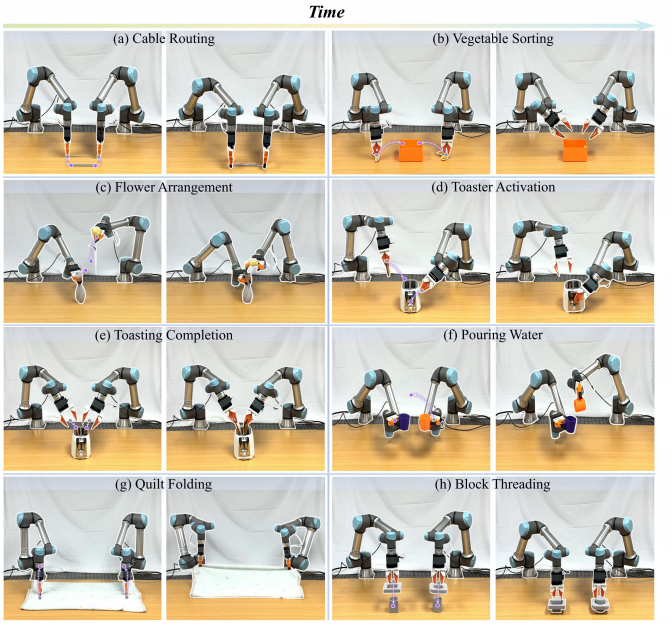


Fig. 9. Real-world bimanual manipulation tasks.

The results are summarized in Table II. The end-to-end DP achieves only 40% and 30% success on Vegetable Sorting and Toasting Completion, respectively, whereas BimanualShift-DP increases these to 100% and 95%, approaching the performance of BimanualShift-PerAct. On more complex collaborative and precision-critical tasks, the performance gap widens further. The end-to-end DP nearly fails on Quilt Folding and Block Threading, while BimanualShift-DP reaches 65% success on Block Threading. BimanualShift-PerAct performs best overall, achieving 95% on Block Threading and 75%/55% on the challenging Pouring Water and Cable Routing tasks, respectively. Notably, BimanualShift consistently unlocks the potential of different policy backbones, with the PerAct-based variant showing particular strength on high-precision and long-horizon tasks.

### D. Generalization in Real-World

To evaluate real-world generalization, we conduct a comprehensive study on the most challenging task, Block Threading, using the best-performing BimanualShift-PerAct. As shown in Fig. 10, we consider seven generalization settings: unseen object color, unseen object shape, lighting change, left–right task exchange, unseen background, extreme glare, and camera perturbation. Each setting is evaluated over 20 trials and decomposed into three stages: Approach, Alignment, and Insertion. We compare BimanualShift-PerAct with PerAct-LF and report stage-wise success rates, average performance, and relative degradation rate.

As shown in Table III, BimanualShift-PerAct consistently outperforms PerAct-LF across all settings. Under standard execution, BimanualShift-PerAct achieves 98.3% average stage-

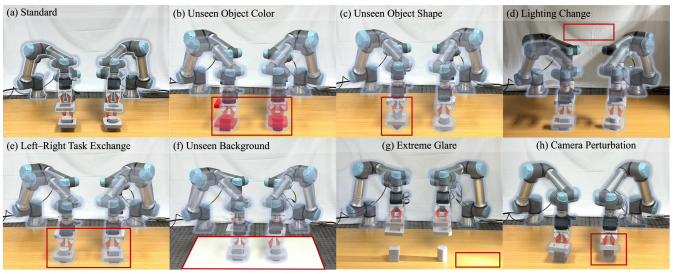


Fig. 10. Seven generalization scenarios for the Block Threading task.

TABLE II
REAL-WORLD TASK EVALUATION

| Tier | Task | Episodes | Keyframes | BimanualShift-PerAct | BimanualShift-DP | DP [2] |
|---|---|---|---|---|---|---|
| Coordination | Toasting Completion | 20 | 6 | 100.0 ± 0.0 | 95.0 ± 4.9 | 30.0 ± 10.2 |
| | Vegetable Sorting | 20 | 4 | **100.0 ± 0.0** | **100.0 ± 0.0** | 40.0 ± 11.0 |
| Collaboration | Quilt Folding | 20 | 5 | 90.0 ± 9.5 | 75.0 ± 8.0 | 10.0 ± 6.5 |
| | Flower Arrangement | 20 | 5 | 85.0 ± 7.5 | 70.0 ± 9.5 | 10.0 ± 6.5 |
| | Toaster Activation | 20 | 6 | 80.0 ± 8.5 | 45.0 ± 9.5 | 5.0 ± 5.0 |
| | Pouring Water | 20 | 4 | 75.0 ± 8.5 | 35.0 ± 8.0 | 0.0 ± 0.0 |
| Precision | Block Threading | 20 | 4 | 95.0 ± 5.0 | 65.0 ± 12.0 | 0.0 ± 0.0 |
| | Cable Routing | 20 | 5 | 55.0 ± 15.5 | 35.0 ± 12.5 | 0.0 ± 0.0 |
| - | Average | - | - | 85.6 ± 4.5 | 65.0 ± 6.0 | 11.9 ± 3.5 |

wise success and 95.0% insertion success, compared with 63.3% and 30.0% for PerAct-LF. Under appearance changes, including unseen color, unseen shape, lighting change, and unseen background, BimanualShift-PerAct maintains average success rates of 95.0%, 86.7%, 81.7%, and 76.7%, substantially higher than PerAct-LF. Under more challenging configuration and sensing changes, including left–right exchange, extreme glare, and camera perturbation, BimanualShift-PerAct still achieves 81.7%, 76.7%, and 68.3% average success, respectively, while PerAct-LF drops to 30.0%, 46.7%, and 20.0%. These results show that arm-conditioned visual projection and residual skill modulation improve generalization to unseen appearances, task configurations, illumination changes, and calibration variations.

### E. Performance in Data-Scarce Regimes

To evaluate performance under extremely limited demonstrations, we select three representative real-world tasks: Vegetable Sorting, Block Threading, and Pouring Water. BimanualShift-PerAct and PerAct-LF are trained with only 15 or 5 demonstrations per task and evaluated over 20 trials.

As shown in Fig. 11(a), BimanualShift-PerAct consistently outperforms PerAct-LF in both 15-shot and 5-shot settings. In the 15-shot setting, BimanualShift-PerAct achieves 100.0%, 95.0%, and 75.0% success on Vegetable Sorting, Block Threading, and Pouring Water, respectively, while PerAct-LF drops to 65.0%, 20.0%, and 35.0%. Under the more challenging 5-shot setting, BimanualShift-PerAct still maintains 85.0%, 65.0%, and 55.0% success, whereas PerAct-LF degrades to 25.0%, 5.0%, and 0.0%. These results indicate that frozen unimanual priors and lightweight adaptation enable more data-efficient bimanual skill transfer under severe data scarcity.

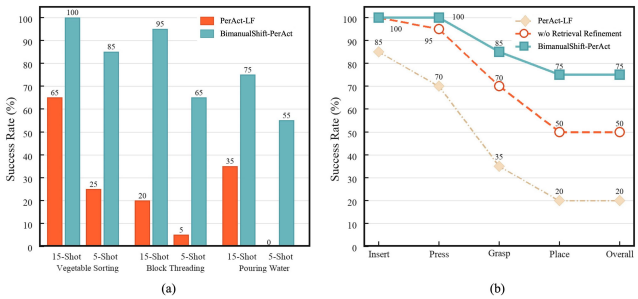


Fig. 11. Data scarcity and continuous learning analysis. (a) Few-shot performance on representative real-world tasks. (b) Segment-wise performance on the long-horizon task.

### F. Continuous Learning Capability

To evaluate retrieval-conditioned residual refinement in long-horizon execution, we construct a long-horizon task after the model has learned Toasting Completion and Toaster Activation. The task consists of four sequential actions: Insert, Press, Grasp, and Place (see Fig. 12). We fine-tune BimanualShift-PerAct with only 5 demonstrations for the novel skill and evaluate it over 20 trials, comparing against the variant w/o Retrieval Refinement and PerAct-LF.

As shown in Fig. 11(b), BimanualShift-PerAct achieves 100.0% success on the previously learned Insert and Press stages, indicating effective retrieval of prior residual contexts. It further maintains 85.0% and 75.0% success on the Grasp and Place stages, outperforming w/o Retrieval Refinement by 15 and 25 percentage points, respectively. Compared with PerAct-LF, which drops sharply after Press and achieves only 20.0% overall success, BimanualShift-PerAct reaches 75.0% overall success, demonstrating that retrieval-conditioned residual refinement improves long-horizon execution under limited demonstrations.

## V. CONCLUSION

This letter presented BimanualShift, an arm-conditioned residual skill transfer framework that reformulates unimanual-to-bimanual transfer as residual coordination over frozen unimanual policy priors. Instead of training a full bimanual policy from scratch, BimanualShift aligns shared observations to policy-compatible arm-specific inputs, re-parameterizes dual-arm coordination as lightweight residual modulation, and reuses successful executions as residual contexts for long-horizon correction. Experiments in RLBench2 and on a real dual-arm platform show that BimanualShift improves few-shot transfer, generalization, and robustness, achieving a 31.7% average success rate in the 30-demonstration complex-task setting and 85.6% average success across eight real-world tasks.

## REFERENCES

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

[2] C. Chi *et al.*, "Diffusion policy: Visuomotor policy learning via action diffusion," *Int. J. Robot. Res.*, vol. 44, pp. 1684–1704, 2025.

TABLE III
GENERALIZATION RESULTS ON THE BLOCK THREADING TASK

| Condition | Method | Approach | Alignment | Insertion | Average | RDR[1] | Condition | Method | Approach | Alignment | Insertion | Average | RDR[1] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Standard | PerAct-LF | 90.0 | 70.0 | 30.0 | 63.3 | - | Left–Right Exchange | PerAct-LF | 55.0 | 30.0 | 5.0 | 30.0 | 52.6 |
| | **BimanualShift-PerAct** | **100.0** | **100.0** | **95.0** | **98.3** | - | | **BimanualShift-PerAct** | **90.0** | **80.0** | **75.0** | **81.7** | **16.9** |
| Unseen Color | PerAct-LF | 80.0 | 60.0 | 20.0 | 53.3 | 15.8 | Unseen Background | PerAct-LF | 60.0 | 35.0 | 10.0 | 35.0 | 44.7 |
| | **BimanualShift-PerAct** | **100.0** | **95.0** | **90.0** | **95.0** | **3.4** | | **BimanualShift-PerAct** | **85.0** | **75.0** | **70.0** | **76.7** | **22.0** |
| Unseen Shape | PerAct-LF | 70.0 | 45.0 | 15.0 | 43.3 | 31.6 | Extreme Glare | PerAct-LF | 75.0 | 50.0 | 15.0 | 46.7 | 26.2 |
| | **BimanualShift-PerAct** | **95.0** | **85.0** | **80.0** | **86.7** | **11.8** | | **BimanualShift-PerAct** | **100.0** | **75.0** | **55.0** | **76.7** | **22.0** |
| Lighting Change | PerAct-LF | 65.0 | 40.0 | 10.0 | 38.3 | 39.5 | Camera Perturbation | PerAct-LF | 40.0 | 15.0 | 5.0 | 20.0 | 68.4 |
| | **BimanualShift-PerAct** | **90.0** | **80.0** | **75.0** | **81.7** | **16.9** | | **BimanualShift-PerAct** | **90.0** | **65.0** | **50.0** | **68.3** | **30.5** |

[1] RDR: relative degradation rate with respect to the standard-setting average of the same method.

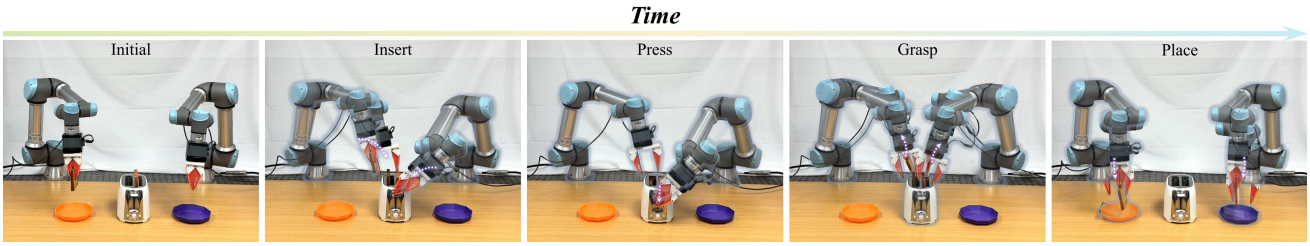


Fig. 12. Long-horizon task composed of four consecutive sub-actions: Insert, Press, Grasp, and Place.


[3] J. Barreiros *et al.*, "A careful examination of large behavior models for multitask dexterous manipulation," *Sci. Robot.*, vol. 11, Art. no. eaea6201, 2026.

[4] K. Black *et al.*, "$\pi_0$: A vision-language-action flow model for general robot control," in *Proc. Robot.: Sci. Syst.*, 2025.

[5] K. Black *et al.*, "$\pi_{0.5}$: A vision-language-action model with open-world generalization," *Proc. Mach. Learn. Res.*, vol. 305, pp. 17–40, 2025.

[6] A. Amin *et al.*, "$\pi^*_{0.6}$: A VLA that learns from experience," *arXiv preprint arXiv:2511.14759*, 2025.

[7] O. Mees *et al.*, "Octo: An open-source generalist robot policy," in *Proc. ICRA Workshop Vision-Language Models Navigation Manipulation*, 2024.

[8] M. J. Kim *et al.*, "OpenVLA: An open-source vision-language-action model," *Proc. Mach. Learn. Res.*, vol. 270, pp. 2679–2713, 2025.

[9] F. Krebs and T. Asfour, "A bimanual manipulation taxonomy," *IEEE Robot. Autom. Lett.*, vol. 7, no. 4, pp. 11031–11038, 2022.

[10] M. Drolet *et al.*, "A comparison of imitation learning algorithms for bimanual manipulation," *IEEE Robot. Autom. Lett.*, vol. 9, no. 10, pp. 8579–8586, 2024.

[11] A. Sintov, S. Macenski, A. Borum, and T. Bretl, "Motion planning for dual-arm manipulation of elastic rods," *IEEE Robot. Autom. Lett.*, vol. 5, no. 4, pp. 6065–6072, 2020.

[12] R. Mon-Williams *et al.*, "Embodied large language models enable robots to complete complex tasks in unpredictable environments," *Nat. Mach. Intell.*, pp. 1–10, 2025.

[13] D. Martinez-Baselga *et al.*, "Hey robot! Personalizing robot navigation through model predictive control with a large language model," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2025, pp. 11002–11009.

[14] A. Goyal, J. Xu, Y. Guo, V. Blukis, Y.-W. Chao, and D. Fox, "RVT: Robotic view transformer for 3D object manipulation," in *Proc. Conf. Robot Learn.*, 2023, pp. 694–710.

[15] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-actor: A multi-task transformer for robotic manipulation," in *Proc. Conf. Robot Learn.*, 2023, pp. 785–799.

[16] M. Grotz, M. Shridhar, Y.-W. Chao, T. Asfour, and D. Fox, "PerAct2: Benchmarking and learning for robotic bimanual manipulation tasks," in *Proc. CoRL Workshop Whole-Body Control Bimanual Manipulation*, 2024.

[17] I.-C. A. Liu, S. He, D. Seita, and G. S. Sukhatme, "VoxAct-B: Voxel-based acting and stabilizing policy for bimanual manipulation," *Proc. Mach. Learn. Res.*, vol. 270, pp. 4354–4370, 2025.

[18] H. Bharadhwaj *et al.*, "Gen2Act: Human video generation in novel scenarios enables generalizable robot manipulation," *arXiv preprint arXiv:2409.16283*, 2024.

[19] Y. Hu *et al.*, "Video prediction policy: A generalist robot policy with predictive visual representations," *Proc. Mach. Learn. Res.*, vol. 267, pp. 24328–24346, 2025.

[20] Y. Wen *et al.*, "VidMan: Exploiting implicit dynamics from video diffusion model for effective robot manipulation," *Adv. Neural Inf. Process. Syst.*, vol. 37, pp. 41051–41075, 2024.

[21] J. Liang *et al.*, "Dreamitate: Real-world visuomotor policy learning via video generation," *Proc. Mach. Learn. Res.*, vol. 270, pp. 3943–3960, 2025.

[22] J. Truong, S. Chernova, and D. Batra, "Bi-directional domain adaptation for sim2real transfer of embodied navigation agents," *IEEE Robot. Autom. Lett.*, vol. 6, pp. 2634–2641, 2021.

[23] R. Julian, B. Swanson, G. S. Sukhatme, S. Levine, C. Finn, and K. Hausman, "Never stop learning: The effectiveness of fine-tuning in robotic reinforcement learning," *Proc. Mach. Learn. Res.*, vol. 155, pp. 2120–2136, 2021.

[24] D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet, and A. Zisserman, "Temporal cycle-consistency learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 1801–1810.

[25] P. Sharma, D. Pathak, and A. Gupta, "Third-person visual imitation learning via decoupled hierarchical controller," *Adv. Neural Inf. Process. Syst.*, vol. 32, 2019.

[26] T. Wang, D. Bhatt, X. Wang, and N. Atanasov, "Cross-embodiment robot manipulation skill transfer using latent space alignment," *arXiv preprint arXiv:2406.01968*, 2024.

[27] M. Xu, Z. Xu, C. Chi, M. Veloso, and S. Song, "XSkill: Cross embodiment skill discovery," in *Proc. Conf. Robot Learn.*, 2023, pp. 3536–3555.

[28] G. Lu, T. Yu, H. Deng, S. S. Chen, Y. Tang, and Z. Wang, "AnyBimanual: Transferring unimanual policy for general bimanual manipulation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2025, pp. 13662–13672.

[29] S. Liu *et al.*, "Grounding DINO: Marrying DINO with grounded pre-training for open-set object detection," in *Proc. Eur. Conf. Comput. Vis.*, 2024, pp. 38–55.

[30] A. Kirillov *et al.*, "Segment anything," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 4015–4026.

[31] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, "RLBench: The robot learning benchmark & learning environment," *IEEE Robot. Autom. Lett.*, vol. 5, pp. 3019–3026, 2020.

[32] P. Wu, Y. Shentu, Z. Yi, X. Lin, and P. Abbeel, "GELLO: A general, low-cost, and intuitive teleoperation framework for robot manipulators," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2024, pp. 12156–12163.

[33] T.-W. Ke, N. Gkanatsios, and K. Fragkiadaki, "3D diffuser actor: Policy diffusion with 3D scene representations," *Proc. Mach. Learn. Res.*, vol. 270, pp. 1949–1974, 2025.