AnyBimanual: Transferring Unimanual Policy for General Bimanual Manipulation

Supplementary Material

1. Additional Experimental Details

1.1. Simulation

We utilize RLBench2 [2] as our main simulated task suite for mulit-task learning. Table 2 is an overview of the 12 selected tasks we use in the experiments. Table 1 is an overview of the 18 selected tasks from RLBench [3] used to pretrain the unimanual checkpoint. The task variations include randomly sampled colors, sizes, counts, placements, and categories of objects. Other properties vary depending on the specific task. Furthermore, objects are randomly arranged on the tabletop within a certain range, adding to the diversity of the tasks. The visual observation includes 6 RGB-D frames of 256×256 resolutions (*i.e.*, front, shoulder left, shoulder right, wrist left, wrist right and overhead), which are shown in Figure 1. In the ablation study, we group the RLBench2 tasks of Table 2 into 3 categories according to their key challenges. The task groups include:

- The `Long` group includes long-term tasks that requires more than 6.5 keyframes. The included tasks are: `pick plate`, `pick laptop`, `put in fridge`, `handover item`, `sweep to dustpan`, `take out of tray` and `handover easy`.
- The `Generalized` group includes tasks with multiple variations, which are differentiated by instructions. The included tasks are: `press buttons` and `handover item`.
- The `Sync` group requires precise coordination, which often causes failures due to asynchronous manipulation. The included tasks are: `lift ball`, `lift tray`, `straighten rope` and `push box`.

1.2. Real-Robot

Hardware Setup. The real robot setup uses two Universal Robots UR5e manipulators, each equipped with a Robotiq 2F-85 gripper. See Figure 2 for reference. For the perception, we use a Realsense RGB-D camera mounted on a tripod, positioned to mimic the viewpoint of human eyes during bimanual tasks. The Realsense provides RGB-D images at a resolution of 640x480 with a frame rate of 30 Hz. The extrinsics between the camera and right arm base-frame are calibrated using the easy_handeye package[1]. Additionally, an ARUCO marker[2] attached to the UR5e's end-effector is employed to aid in the calibration process.

Tasksuite Descriptions. We provide a detailed description of the real-world tasks we used to evaluate our AnyBimanual in Figure 3, where we follow [4] to classify our 9 real-world experiments into 5 categories, covering 5 bimanual collaboration patterns.

- `Lift Cabinet` requires the agent to simultaneously hold two sides of the cabinet to lift it up, which poses challenges to synchronous coordination.
- `Fold Clothes` requires the agent to simultaneously grasp and fold the flexible clothes, which involves soft manipulation and synchronous coordination.

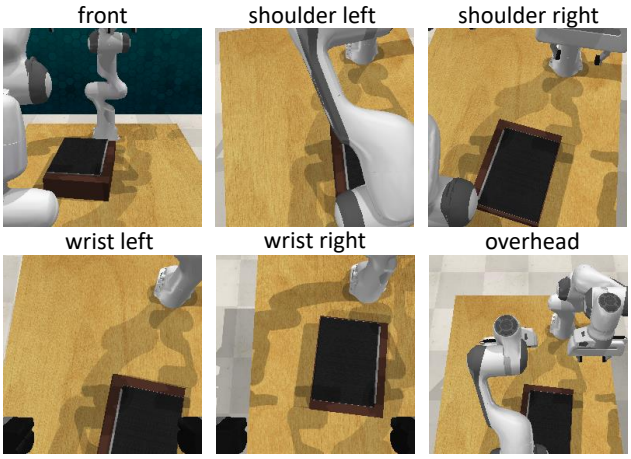

Figure 1. **Visual Observation in RLBench2.** We adopt 6 RGB-D cameras to cover the whole workspace.

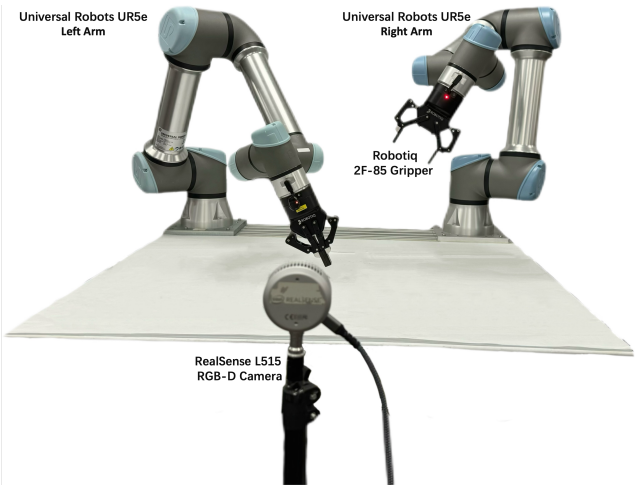

Figure 2. **Real-Robot Setup** with one RealSense L515 RGB-D Camera and Two UR5e Manipulators.

[1] https://github.com/IFL-CAMP/easy_handeye
[2] https://github.com/pal-robotics/aruco_ros

Table 1. **Unimanual Tasks.** This table shows the 18 unimanual tasks in RLBench that are used to pretrain the unimanual policy.

Task	# of Variations	# of Average Keyframes	Human Instruction Template
open drawer	3	3.0	*"open the ___ drawer"*
slide block	4	4.7	*"slide the block to ___ target"*
sweep to dustpan	2	4.6	*"sweep dirt to the ___ dustpan"*
meat off grill	2	5.0	*"take the ___ off the grill"*
turn tap	2	2.0	*"turn ___ tap"*
put in drawer	3	12.0	*"put the item in the ___ drawer"*
close jar	20	6.0	*"close the ___ jar"*
drag stick	20	6.0	*"use the stick to drag the cube onto the ___ target"*
stack blocks	60	14.6	*"stack ___ ___ blocks"*
screw bulb	20	7.0	*"screw in the ___ light bulb"*
put in safe	3	5.0	*"put the money away in the safe on the ___ shelf"*
place wine	3	5.0	*"stack the wine bottle to the ___ of the rack"*
put in cupboard	9	5.0	*"put the ___ in the cupboard"*
sort shape	5	5.0	*"put the ___ in the shape sorter"*
push buttons	50	3.8	*"push the ___ button, [then the ___ button]"*
insert peg	20	5.0	*"put the ring on the ___ spoke"*
stack cups	20	10.0	*"stack the other cups on top of the ___ cup"*
place cups	3	11.5	*"place ___ cups on the cup holder"*

Table 2. **Bimanual Tasks.** This table shows the 12 bimanual tasks in RLBench2 that are used to fine-tune the bimanual policy.

Task	# of Variations	# of Average Keyframes	Human Instruction Template
pick laptop	1	7.2	*"pick up the notebook."*
pick plate	1	6.6	*"pick up the plate."*
straighten rope	1	5.9	*"straighten the rope."*
lift ball	1	4.0	*"lift the ball."*
lift tray	1	5.1	*"lift the tray."*
push box	1	2.1	*"push the box to the red area."*
put in fridge	1	7.8	*"put the bottle into the fridge."*
press buttons	5	4.0	*"push the ___ and ___ button."*
handover item	5	7.6	*"hand over the ___ item."*
sweep to dustpan	1	7.3	*"sweep the dust to the pan."*
take out tray	1	8.7	*"take tray out of oven."*
handover easy	1	7.5	*"hand over the item."*

- `pick&place Sync` requires the agent to simultaneously pick up two colored cubes specified by the human instructions, and then place them in corresponding boxes, which involves semantic understanding.
- `Handover Bowl` requires the agent to lift a bowl and hand it over to the other hand, posing challenges in coordinating the handover between hands.
- `Wash Dishes` requires the agent to pick up a bowl and then apply detergent to it, testing the asynchronous coordination of both hands.
- `Play Ping Pong` requires the agent to pick up both a ball and a racket, toss the ball with one hand and hit it with the racket in the other in a high toss serve, testing asynchronous coordination.
- `Rotate Toothbrush` requires the agent to pick up a toothbrush and a cup simultaneously, then rotate the toothbrush 180 degrees to drop it into the cup, challenging the precision of rotation and coordination of both hands.
- `Type Writing` requires the agent to sequentially type the letters "robot" on a keyboard using both hands, testing long-horizon coordination.
- `Pick & Place Async` requires the agent to simultaneously pick up two cubes of different colors and then place them into the same bowl, testing the asynchronous coordination of both hands.

Data Collection and Preprocessing. We collect demonstrations with two Xbox joysticks. Each gamepad is a 6-DoF controller. The gamepads adjust the velocity of the arm's end-effector to translate and rotate in all directions, with reference to the arm's base frame. For robot control, we utilize the Universal Robots ROS Driver[3]. After collecting the complete trajectory, we extract keyframes from the trajectory manually to simplify the robot learning. Two examples of the extracted keyframes are shown in Figure 4. The total time consumption of collecting one episode including controlling the robot arms and keyframe extraction is ~ 1.5 minutes.

Data Augmentation. In line with [2, 5], we turn on SE(3) augmentation to boost the robustness of all learned policies. Based on the open source code of [2], we modify the perturbation strategy to ensure that the perturbed ground-truth actions of both arms are reasonable in Figure 5. We plan to release our code for more details.

Policy Deployment. During the testing phase, we first randomize the positions of related objects in the current task. After resetting the manipulators to the home positions, we

[3] https://github.com/UniversalRobots/Universal_Robots_ROS_Driver

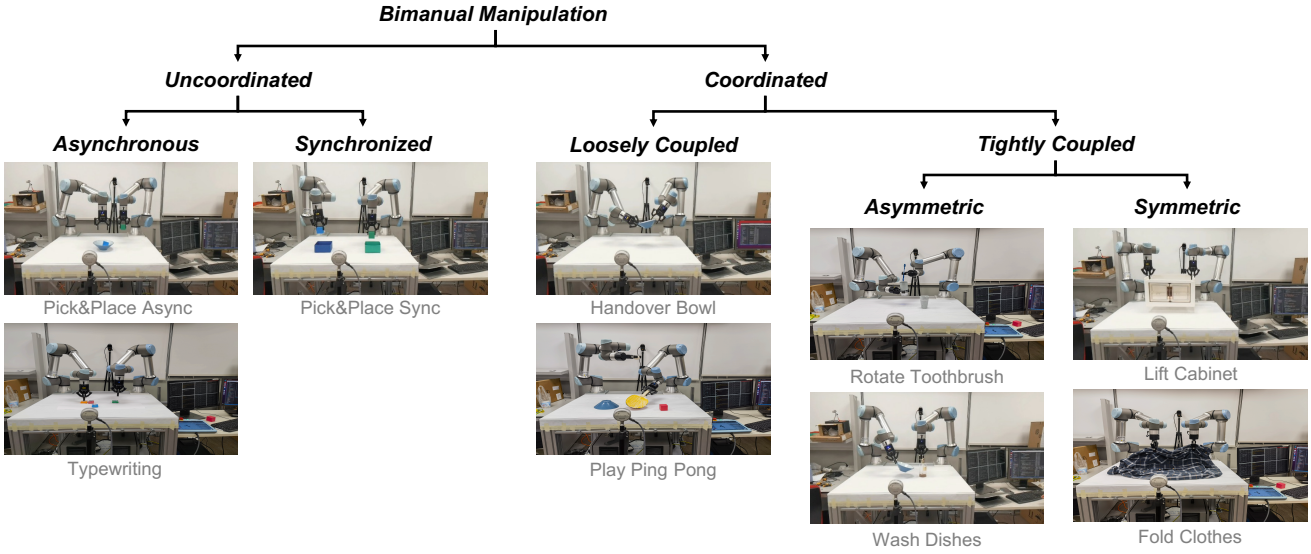

Figure 3. **Real-Robot Task Taxonomy**. Following [4], we classify the reported 9 real-world tasks into 5 categories according to their collaboration patterns.

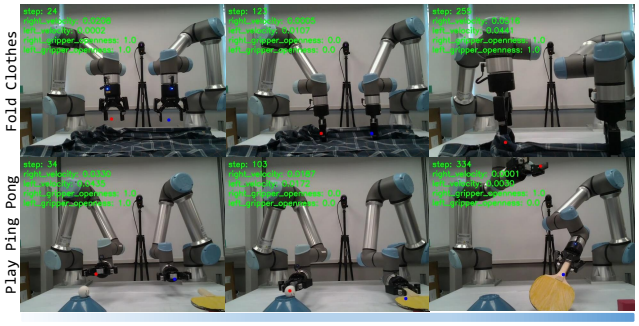

Figure 4. **Real-world Keyframes.** We manually select keyframes from the collected trajectories to simplify training.

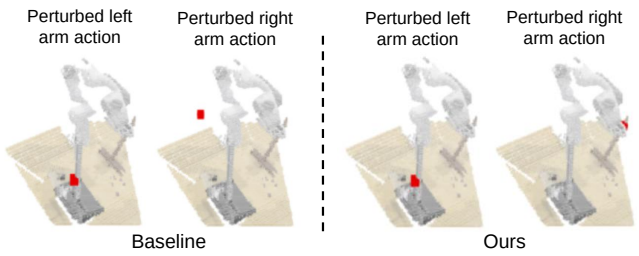

Figure 5. **SE(3) Augmentation.** We slightly modify the SE(3) augmentation strategy to ensure that the perturbed ground-truth actions of both arms are reasonable.

query the policy with the language instruction, initial observation and proprioception to obtain the next best poses of two end-effectors. We then use Moveit[4] to reach the poses and query the policy again with new observations, until task completion or reaching the maximal episode length.

[4] https://github.com/moveit/moveit

Table 3. **A default set of hyper-parameters.**

Config	Value
Training iteration	100k
Image size	$256 \times 256 \times 3$
Voxel size	$100 \times 100 \times 100$
Batch size	4
Optimizer	LAMB [8]
Learning rate	0.0005
Weight decay	0.000001
λ_{skill}	0.0001
λ_{voxel}	0.001
Number of skill primitives K	18
Dimension of Skill Representations D	512

2. Additional Implementation Details

2.1. Hyperparameters

The hyperparameters used in AnyBimanual are shown in Table 3. To ensure that the auxiliary objectives $\mathcal{L}_{\text{skill}}$ and $\mathcal{L}_{\text{voxel}}$ remain in the same magnitude as $\mathcal{L}_{\text{BC}}$, we set $\lambda_{\text{skill}} = 0.0001$ and $\lambda_{\text{voxel}} = 0.001$. Other hyperparameters are in line with previous works [2, 5] for fair comparisons.

2.2. Architectures

Skill Manager. We fuse voxel embeddings and language embeddings by projecting both to a hidden size of 256 and concatenating them. After layer normalization, the combined sequence is processed by a frozen GPT-2 Transformer configured with a hidden size of 256 and 8 attention heads. The transformer's output is passed through a linear layer

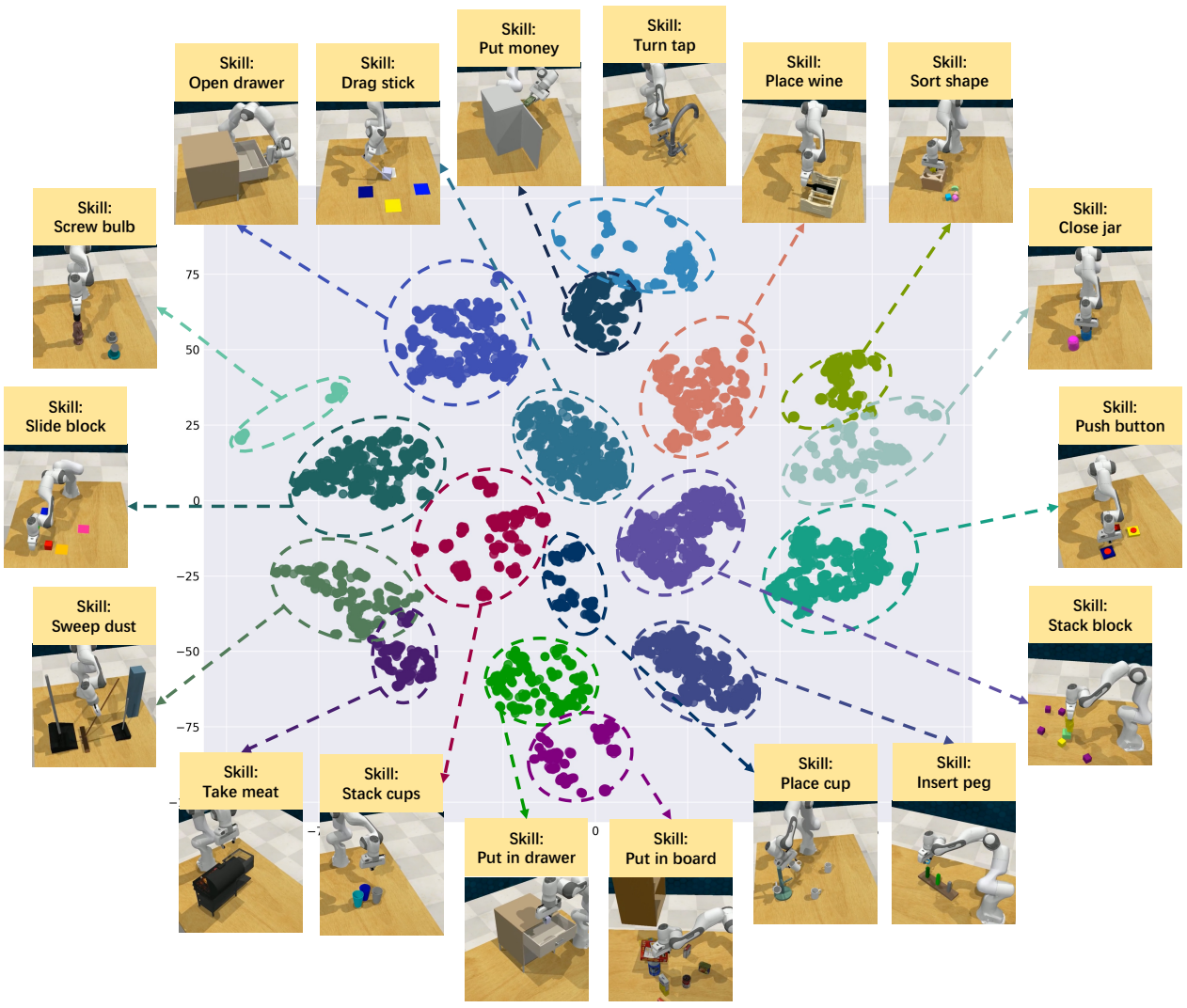

Figure 6. **Skill Clustering.** We cluster the skill representations to further interpret the discovered skills of our AnyBimanual.

to produce logits over 18 classes, and softmaxed to obtain probabilities. These probabilities are then used to reconstruct the skill representation via a weighted sum over the skill primitives.

Visual Aligner. We employ a lightweight module to process input embeddings through a Conv1D-ReLU block, then split into two branches to generate left and right masks via additional Conv1D-ReLU layers. After element-wise multiplication of these masks with the original input, we incorporate residual connections by adding the masked outputs back to the input embeddings.

2.3. Skill Primitives

The skill primitives use CLIP embeddings of unimanual language templates (*e.g.*, 'open the __ drawer') from Table 1 of the supplementary file, as we expect them to represent foundational motions that generalize across object categories in unseen scenarios.

3. Additional Experimental Results

3.1. Skill Clustering

The goal of the skill manager is to obtain the language embedding via linear skill representations, so that the pre-trained unimanual policy model (*i.e.*, PerAct) can be prompted to generate feasible manipulation actions. In Figure 6, we cluster the skill representations while evaluating the last checkpoint of PerAct+AnyBimanual in RLBench2. We can observe that the learned skill representations are reasonably clustered into 18 categories, which corresponds to the number of pretrained unimanual tasks. We also visualize the related unimanual skill of each cluster by finding the nearest unimanual task embedding. The clustering results further interpret that the proposed skill manager en-

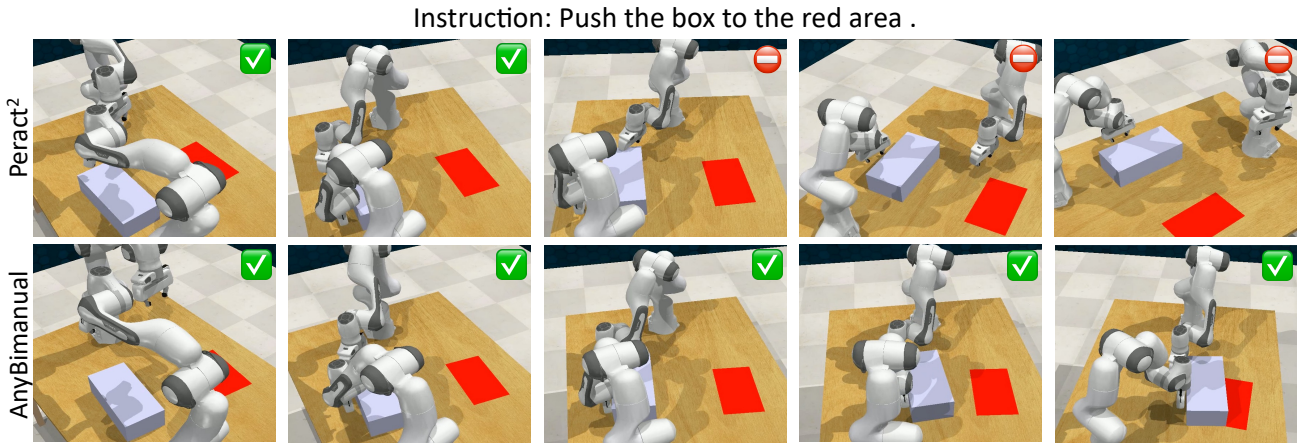

Figure 7. **Comparions of the rollouts of PerAct^2 and our AnyBimanual.** AnyBimanual demonstrates complex collaboration patterns by incorporating the skill manager and the visual aligner to schedule the manipulators correctly.

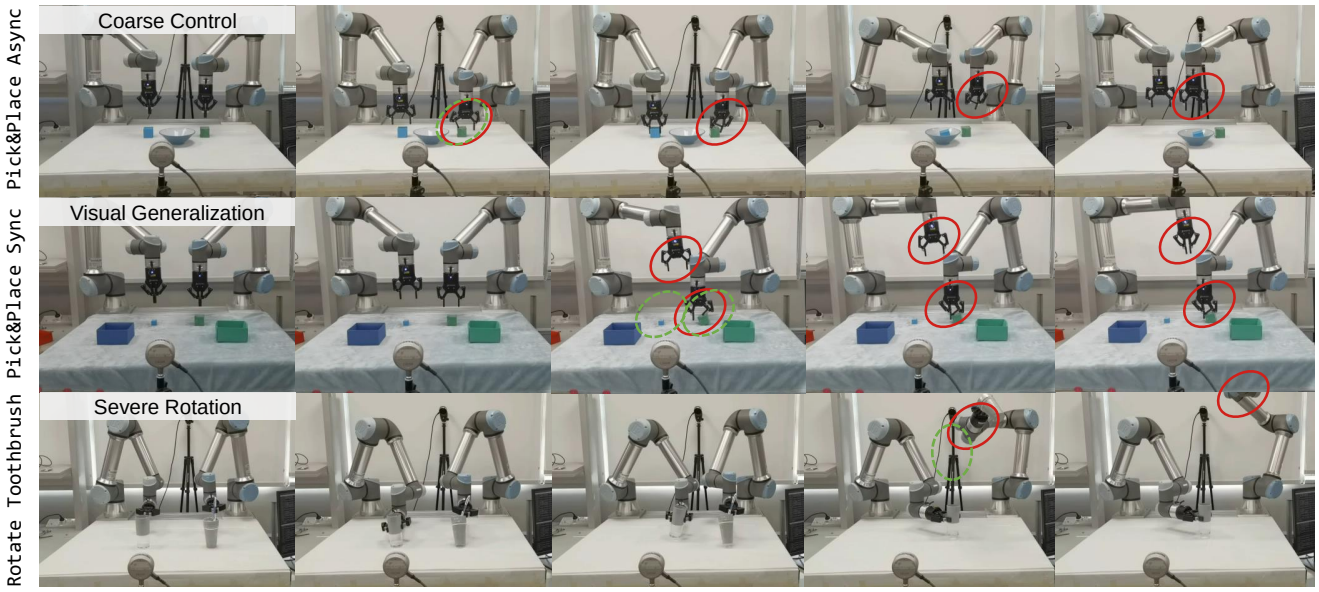

Figure 8. **Error Modes in Real-world Rollouts.** We study the common failure cases of our AnyBimanual in real-world settings, where the incorrect end-effector actions are marked with red circles. We also provide the correct actions with green circles for reference.

ables compact skill representation learning with rich semantics for dynamic scheduling.

3.2. Rollout Comparisons

We present a qualitative example of the generated action sequences in Figure 7, comparing PerAct^2 and our AnyBimanual method. In this case, the instruction is "Push the box to the red area". The PerAct^2 agent fails to touch the box and only mimics the expert's forward-pushing motion, resulting in an incomplete action. In contrast, our AnyBimanual agent ensures both arms make contact with the box and successfully push it into the red area, demonstrating the superior ability of our method in having each arm correctly identify contact with the object and successfully complete the task.

3.3. Visualization

Skill Manager. In Figure 9, we visualize the skill manager in two challenging tasks requiring heavy coordination. AnyBimanual completes these tasks by scheduling skill representations across arms simultaneously and dynamically.

Viusal Aligner. Figure 10 provides additional visual aligner visualizations in two tasks. by predicting spatial soft masks for voxel observation, the visual aligner encourages both

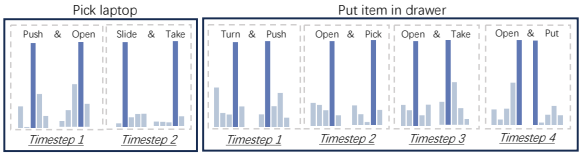

Figure 9. **Visualization of Skill Manager.**

arms to focus on the interaction and ignore irrelevant parts.

3.4. Error Modes

To further study the limitations and risks of the proposed method, we visualize the common failure cases in Figure 8. Common error modes include coarse control, which is often viewed in `Pick&Place Sync` and `Pick&Place Async`. In the upper case, the robot fails to grasp the green block due to a small end-effector deviation, which may be caused by inconsistent data collection. In the middle case, the robot struggles to generalize to different backgrounds and object sizes, which leads to an out-of-distribution movement of the left robot arm. In the bottom case, the low-level motion planner cannot handle the severe rotation predicted by the agent, which can result in a dangerous cup fall. We hope that the analysis can enlighten future works in bimanual policy transferring and prevent potential risks in real-world deployments.

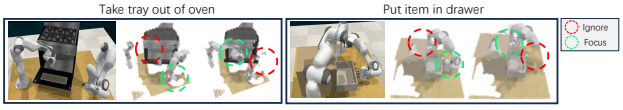

Figure 10. **Visualization of Visual Aligner.**

3.5. Bimanual Decomposability

We complement a heuristic measurement to assess the feasibility of decomposing a given bimanual task with a learned skill manager at the semantic level. Specifically, higher entropy in the predicted combination weights indicates greater difficulty in decomposition with certain unimanual primitives. In Figure 11, this metric strongly correlates with final success rates in four simulation tasks, delineating the boundary.

3.6. Real-world Videos

We provide 9 additional comprehensive real-world episodes generated by our AnyBimanual in the attached video file (*demo.mp4*). Note that all 9 tasks are completed by a single model with different natural language instructions to specify the current task.

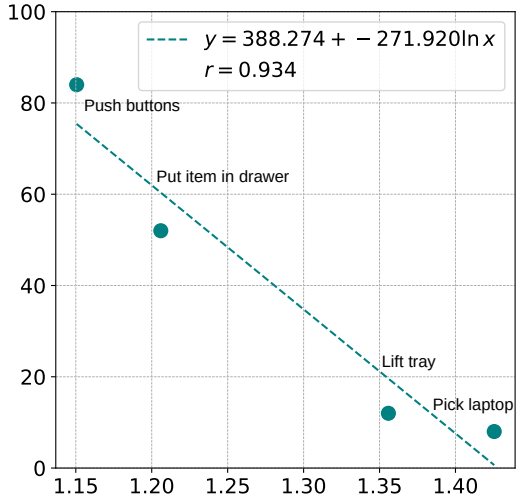

Figure 11. **Bimanual Decomposability.**

Table 4. **Real-world Results with 5 Demonstrations Per Task.** We train AnyBimanual on 9 real-world tasks with 5 demonstrations for each task, and evaluate each variation over 5 episodes.

Task	Success Rate (%)
`Lift Cabinet`	80.0
`Fold Clothes`	80.0
`Pick&Place Sync`	80.0
`Handover Bowl`	20.0
`Wash Dishes`	20.0
`Play Ping Pong`	40.0
`Rotate Toothbrush`	20.0
`Typewriting`	60.0
`Pick&Place Async`	80.0

3.7. Real-world Experiments with Limited Data.

To further validate the effectiveness and practicality of AnyBimanual with limited expert data, we train AnyBimanual on 9 real-world tasks, each with 5 demonstrations. As per the main paper, we evaluate each variation over 5 episodes, and report the success rates of each task. The results are illustrated in Table 4, where AnyBimanual still maintains an acceptable average success rate of 53.33%.

4. Discussions on Limitations

Cross-embodiment Skill Transferring. AnyBimanual needs careful visual alignment between unimanual and bimanual systems to ensure robust policy transfer, which means it is not designed for cross-embodiment skill transferring. Even though the 7-DoF end-effector action space can be used with various robots, differences in the visual appearance of unimanual and bimanual robots may limit generalizability. Because the visual aligner only attempts to align the observation by segmenting out unimanual workspace from the bimanual system, without consid-

ering their appearance differences. This can be addressed by visual inpainting [7] or transferring cross-embodiment unimanual policies [6].

Cross-task Generalization. Although AnyBimanual enables general multi-task bimanual manipulation with few demonstrations, it struggles to perform zero-shot generalization to unseen tasks. This is mainly due to the end-to-end skill manager and visual aligner still necessitating expert demonstrations to learn proper coordination patterns and unlock the capacity of unimanual policy. Explicit methods such as leveraging large language models [1] to orchestrate unimanual policies can realize weak zero-shot generalization, but struggle with contact-rich tasks without learning.

References

[1] Zeyu Gao, Yao Mu, Jinye Qu, Mengkang Hu, Lingyue Guo, Ping Luo, and Yanfeng Lu. Dag-plan: Generating directed acyclic dependency graphs for dual-arm cooperative planning. *arXiv preprint arXiv:2406.09953*, 2024. 7

[2] Markus Grotz, Mohit Shridhar, Yu-Wei Chao, Tamim Asfour, and Dieter Fox. Peract2: Benchmarking and learning for robotic bimanual manipulation tasks. In *Conference on Robot Learning (CoRL)*, 2024. 1, 2, 3

[3] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters (RAL)*, 5(2):3019–3026, 2020. 1

[4] Franziska Krebs and Tamim Asfour. A bimanual manipulation taxonomy. *IEEE Robotics and Automation Letters (RAL)*, 7 (4):11031–11038, 2022. 1, 3

[5] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning (CoRL)*, 2023. 2, 3

[6] Lirui Wang, Xinlei Chen, Jialiang Zhao, and Kaiming He. Scaling proprioceptive-visual learning with heterogeneous pre-trained transformers. *arXiv preprint arXiv:2409.20537*, 2024. 7

[7] Su Wang, Chitwan Saharia, Ceslee Montgomery, Jordi Pont-Tuset, Shai Noy, Stefano Pellegrini, Yasumasa Onoe, Sarah Laszlo, David J Fleet, Radu Soricut, et al. Imagen editor and editbench: Advancing and evaluating text-guided image inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18359–18369, 2023. 7

[8] Yang You, Jing Li, Sashank J. Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training BERT in 76 minutes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020. 3