Supplementary Material

A More Details for EchoArena Training

A.1 Datasets and Data Collection

We train EchoArena and the baseline world models separately on three datasets: (1) *Simpler-BridgeV2 WidowX* [6, 10], which contains 4 tasks with single-view observations; (2) *RoboTwin* [2], which contains 50 tasks on a dual-arm AgileX platform with three synchronized camera views; and (3) a *Franka-Real* dataset collected in this work, which contains 20 tasks with two camera views, i.e., a left-side camera and a wrist camera. To enable the world model to capture realistic policy behaviors, for each task we collect rollout trajectories with a balanced ratio of successful and failed episodes, i.e., 50% success and 50% failure. We train a separate EchoArena and a separate baseline world model on each dataset.

For the Simpler and RoboTwin settings, we additionally collect approximately 1,000 task-agnostic trajectories for pre-training. These trajectories cover diverse robot motions beyond the target task set and are used to improve the generalization of action-conditioned trajectory control. Figure 1 summarizes the composition of our training data.

A.2 World Model Training Details

Our EchoArena is built on top of IRASim [12]. Specifically, IRASim is a latent diffusion video model that first encodes input frames into the latent space of a frozen SDXL VAE [9], then tokenizes the latent video and models it with a Diffusion Transformer (DiT) [8] equipped with factorized spatial-temporal attention. To strengthen the alignment between action trajectories and future frames, IRASim further injects frame-level action conditions into the transformer blocks. We adopt this design as the backbone of EchoArena and use the IRASim-XL/2 configuration in all experiments.

In the Simpler and RoboTwin experiments, we first pre-train EchoArena on approximately 1,000 task-agnostic trajectories collected by ourselves before fine-tuning on task-specific data. The purpose of this stage is to expose the model to a broader range of motions and action patterns, so that it can learn a richer action space and generalize trajectory control better across tasks. During task-specific training, we explicitly use embodiment-dependent joint-space actions as the control signal, instead of task-space end-effector poses. This design removes the burden of implicitly learning inverse kinematics inside the world model and improves fine-grained control in our setting. Moreover, several VLA policies used in our evaluation take current joint angles as proprioceptive inputs during inference; therefore, using joint angles as the action interface for interacting with the world model is natural and consistent with the policy's own proprioceptive loop.

We use 16-frame video clips for training, where one frame is used as the conditioning frame and the model predicts the remaining 15 future frames conditioned

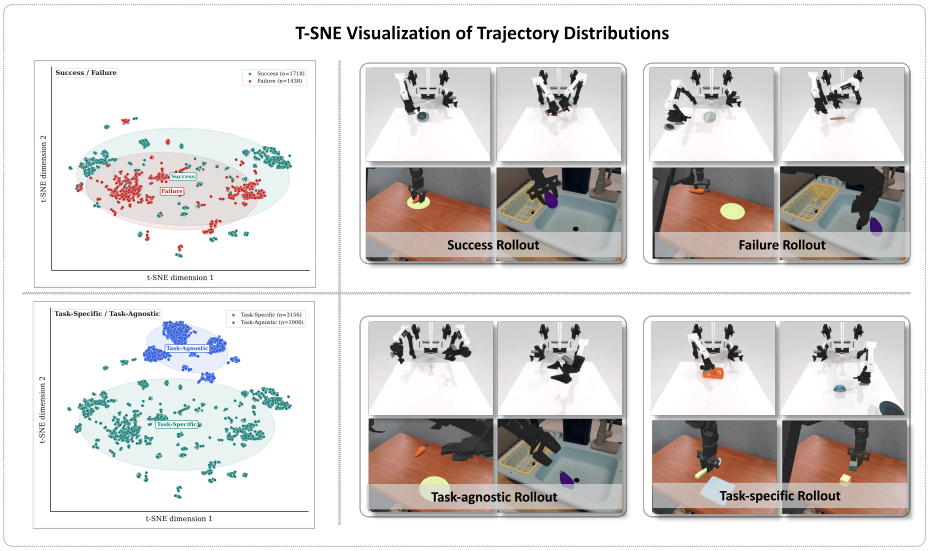

Fig. 1: Training data distribution for EchoArena. We train world models on three task-specific datasets covering Simpler-BridgeV2 WidowX, RoboTwin, and Franka-Real settings. For each task, we collect a balanced set of successful and failed rollouts to capture realistic policy behaviors under both positive and negative outcomes. In addition, for the Simpler and RoboTwin settings, we collect approximately 1,000 task-agnostic trajectories for pre-training, which expands the diversity of motion patterns and improves the generalization of action-conditioned trajectory control.

on joint actions. Detailed hyperparameters are summarized in Table 1. For the baseline world models, IRASim [12] and Ctrl-World [4], we strictly follow their official training recipes and train them directly on task-specific data without our task-agnostic pre-training stage. In addition, both baselines use task-space control, where the action condition is the end-effector pose $g_t \in SE(3)$.

A.3 Policy Model Training Details

We evaluate our framework with five representative vision-language-action (VLA) / policy models, including DP [3], ACT [11], RDT [7], π_0 [1], and $\pi_{0.5}$ [5]. For fair policy-in-the-loop evaluation, each policy is trained to be strictly aligned with the corresponding world model setting in both observation space and action space. Concretely, if a world model is trained in the RoboTwin setting with three camera views and dual-arm AgileX joint-space control, then the corresponding policy is also trained with the same three-view observations and outputs dual-arm AgileX joint actions. We follow the same alignment rule for the Simpler and Franka-Real settings, ensuring that the policy-world-model interface is fully consistent in embodiment, camera configuration, and control space.

Regarding training data, for the Simpler experiments, all policies are trained on the real-world Bridge dataset [10]. For the RoboTwin and Franka-Real exper-

Table 1: Training hyperparameters of EchoArena. We use the same backbone across datasets and only change the observation configuration and embodiment-specific settings. For Simpler and RoboTwin, EchoArena is first pre-trained on task-agnostic trajectories and then fine-tuned on task-specific data.

Hyperparameter	Simpler	RoboTwin	Franka-Real
Backbone	IRASim-XL/2	IRASim-XL/2	IRASim-XL/2
Number of frames	16	16	16
Image size per view	256×320	176×304	240×320
Number of views	1	3	2
Multi-view	False	True	True
Dual-arm	False	True	False
Use joint-space action	True	True	True
Learn sigma	False	False	False
Condition type	Frame-level conditioning	Frame-level conditioning	Frame-level conditioning
Conditioning frames	1 masked frame	1 masked frame	1 masked frame
Task-agnostic pre-training	Yes (∼1000 traj.)	Yes (∼1000 traj.)	No
Learning rate	1×10^{-4} (pretrain), 2.5×10^{-5} (finetune)	1×10^{-4} (pretrain), 2.5×10^{-5} (finetune)	2.5×10^{-5}
Training steps	200k pretrain + 150k finetune	200k pretrain + 150k finetune	150k finetune
Gradient clipping	0.1	0.1	0.1
Start clip iteration	0	0	0
Local batch size	2	2	2
Local validation batch size	1	1	1
Gradient accumulation steps	1	1	1
Global seed	3407	3407	3407
Attention mode	math	math	math

Table 2: Policy models used in evaluation and their action horizons. Unless otherwise specified, the action horizon is set to 15. For RoboTwin, several policies preserve the native prediction horizon of their released checkpoints.

Policy	Action Horizon
DP	15
ACT	15 (45 on RoboTwin)
RDT	15 (60 on RoboTwin)
π_0	15 (50 on RoboTwin)
$\pi_{0.5}$	15

iments, all policies are trained on our in-house collected datasets corresponding to the target embodiments and camera setups.

We also align the action horizon of each policy checkpoint with the rollout setting used in the world-model evaluation. For most policy models, we set the action horizon to 15. The only exceptions are in the RoboTwin experiments, where we use the original checkpoint-specific horizons for several policies: the action horizon of π_0 is set to 50, ACT is set to 45, and RDT is set to 60. These settings follow the native prediction horizon of their corresponding checkpoints and are kept unchanged during evaluation. The detailed action horizon configuration is listed in Table 2.

B More Details for Policy Evaluation

B.1 Interaction Between the World Model and Policy

Our policy evaluation framework tests the closed-loop interaction between a world model (WM) and a vision-language-action (VLA) policy. Several VLA policies in our evaluation require the robot's current joint angles as proprioceptive inputs when predicting the next action. For a policy evaluator based purely on video generation, however, such proprioceptive states are not directly observable from the generated frames. To avoid this mismatch, all VLAs used in our evaluation are trained to output joint-space actions. This design allows each policy to reuse its own predicted joint trajectory as proprioception for the next round of action inference, thereby forming a consistent autoregressive evaluation loop.

For our EchoArena, the world model is itself conditioned on joint-space control. Given the current visual observation o_t and language instruction l, the policy predicts a horizon of joint actions

$$a^{\text{joint}}_{t:t+H-1} = \pi(o_t, q^{\text{joint}}_t, l),$$

where q^{joint}_t denotes the current proprioceptive joint state and H is the action horizon of the policy. EchoArena then autoregressively consumes these H joint actions and predicts future visual observations. In each rollout round, EchoArena generates 15 future frames conditioned on the predicted joint trajectory:

$$o_{t+1:t+15} = \text{WM}_{\text{EchoArena}}(o_t, a^{\text{joint}}_{t:t+H-1}).$$

After generation, we take the last $f \cdot v$ frames from the generated clip as the next policy input, where f denotes the number of input frames required by the VLA and v denotes the number of camera views. These frames are fed back to the policy for the next action prediction, yielding a closed evaluation loop between the policy and the world model.

For Ctrl-World, the world model is conditioned on task-space end-effector poses rather than joint angles. Therefore, before interacting with Ctrl-World, we convert the policy-predicted joint trajectory into end-effector poses using forward kinematics (FK):

$$g_{t:t+H-1} = \text{FK}(a^{\text{joint}}_{t:t+H-1}),$$

where $g_t \in SE(3)$ denotes the end-effector pose. The converted pose sequence is then used as the control condition for video generation:

$$o_{t+1:t+15} = \text{WM}_{\text{Ctrl-World}}(o_t, g_{t:t+H-1}).$$

For IRASim, the control signal is not the absolute end-effector pose, but the end-effector delta action, i.e., incremental commands. Therefore, we first obtain the task-space pose sequence from the predicted joint trajectory using FK, and then compute the pose increments:

$$g_{t:t+H-1} = \text{FK}(a^{\text{joint}}_{t:t+H-1}),$$

$$\Delta g_{t+i} = g_{t+i}^{-1} g_{t+i+1}, \quad i = 0, \ldots, H-2.$$

The resulting incremental commands are used to condition IRASim:

$$o_{t+1:t+15} = \mathrm{WM}_{\mathrm{IRASim}}(o_t, \Delta g_{t:t+H-2}).$$

Overall, although the three world models use different action interfaces, they are all evaluated under the same policy-in-the-loop protocol. At each rollout step, a VLA predicts a horizon of joint actions from the current visual observations, the actions are converted into the control format required by the corresponding world model, and the generated future frames are returned to the policy as the observation for the next step. This protocol enables a unified and fair comparison of different world models for long-horizon policy evaluation.

B.2 Task Details and Criteria

In our experiments, we use human annotators to evaluate whether each rollout is successful. Although this process could in principle be automated by a reward model for large-scale benchmarking, the focus of this paper is on the accuracy of world-model-based policy evaluation itself. Therefore, we adopt human evaluation as our primary protocol to avoid introducing extra uncertainty from imperfect reward modeling. For all settings, annotators are provided with explicit task-specific criteria and judge whether each rollout follows the instruction and achieves the intended outcome.

Simpler. For the Simpler setting [6], we evaluate four tasks, and for each task we perform 12 rollouts. Following the official Simpler evaluation protocol [6], we record not only whether the task is fully completed, but also whether the robot successfully grasps the correct target object. The four tasks are:

- **Put the spoon on the towel.** A spoon and a towel are placed on two different vertices of a square layout on the tabletop. The spoon may start in either a horizontal or a vertical orientation, requiring the robot to adapt its gripper orientation before grasping. A rollout is considered partially successful if the robot grasps the spoon correctly, and fully successful if it further places the spoon onto the towel.
- **Put carrot on plate.** This task follows the same setup as "put the spoon on the towel", except that the spoon is replaced by a carrot and the towel is replaced by a plate. A rollout is considered partially successful if the carrot is correctly grasped, and fully successful if it is placed onto the plate.
- **Stack the green block on the yellow block.** A green block and a yellow block are placed at different positions on the tabletop. The robot must grasp the green block and stack it on top of the yellow block. As in the official setup, we vary the initial layout across different square sizes. A rollout is considered partially successful if the correct block is grasped, and fully successful if the green block is stably stacked on the yellow block.
- **Put eggplant into yellow basket.** An eggplant is placed in a sink basin and a yellow basket is placed at another position. The eggplant is initialized

at random positions and orientations while remaining directly graspable. A rollout is considered partially successful if the eggplant is correctly grasped, and fully successful if it is placed into the yellow basket.

RoboTwin. For the RoboTwin setting [2], we evaluate four tasks, and for each task we perform 12 rollouts. Different from Simpler, RoboTwin uses only binary success labels and does not assign partial credit. The four tasks are:

- **Grab Roller.** A roller is placed on the tabletop. The dual-arm AgileX robot is required to use both grippers to simultaneously grasp the roller and lift it from the table. A rollout is considered successful if the robot establishes a stable bimanual grasp and lifts the roller successfully.
- **Place Container Plate.** A container and a plate are placed on the tabletop. The robot is required to grasp the container and place it onto the plate. In each rollout, the shape, color, and position of both the container and the plate are varied to increase scene diversity. A rollout is considered successful if the robot correctly grasps the container and places it onto the target plate.
- **Place Burger Fries.** A plate is placed on the tabletop, with a burger and a pack of fries located on the two sides of the plate. The robot is required to grasp the two food items and place them onto the plate. A rollout is considered successful if both the burger and the fries are successfully picked up and placed onto the target plate.
- **Stack Bowls Two.** Two bowls are initialized at random positions on the tabletop. The robot is required to first grasp one bowl and place it at the center of the table, and then grasp the other bowl and stack it on top of the first one. A rollout is considered successful if the robot completes both stages and the final stacked configuration is stable.

Franka-Real. For the Franka-Real setting, we evaluate two tasks, and we use the same criteria for both real-robot execution and world-model-generated videos. Instead of binary success labels, we adopt a four-level scoring scheme with additive credits. Specifically, each rollout receives 0.25 for moving to the correct source position, 0.25 for successfully grasping the target object, 0.25 for moving to the correct destination position, and 0.25 for fully completing the task. The two tasks are:

- **Pick up the Milk.** A milk carton and a cup are placed on the tabletop. The robot is required to grasp the milk carton and place it next to the cup. A rollout receives partial credit according to the four stages above. It is considered fully successful only if the milk carton is placed at the target location next to the cup and remains stable without falling afterward.
- **Upright the Bucket.** A fallen tennis bucket and two tennis balls are placed on the tabletop. The robot is required to grasp the bucket and reorient it into an upright pose. A rollout receives partial credit according to the same four stages. It is considered fully successful only if the bucket is successfully uprighted and remains stable afterward.

C Supplementary Experiment Visualizations

C.1 Simulation Visualizations

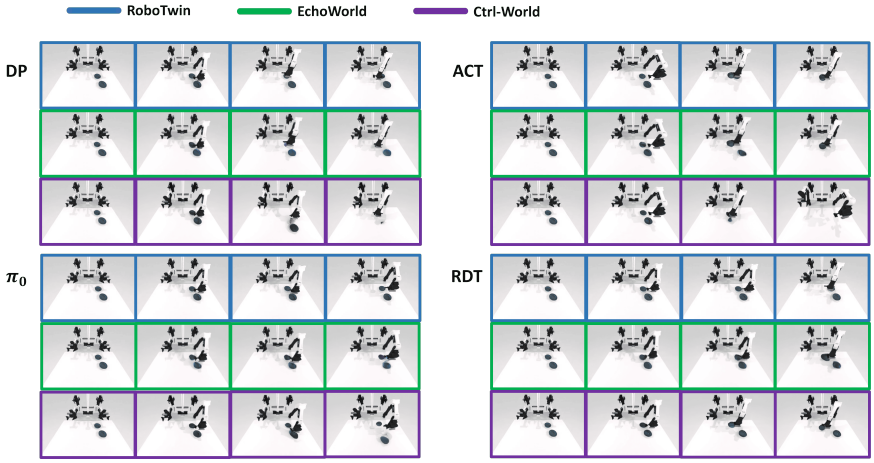

Fig. 2: Closed-loop policy rollouts in RoboTwin. Qualitative comparison of policy-in-the-loop rollouts generated by different world models on RoboTwin tasks. Given the same initial observations and policy outputs, EchoArena, Ctrl-World, and other baselines are rolled out autoregressively in closed loop with multiple VLA policies. EchoArena produces temporally coherent future observations that better preserve robot motion consistency, robot-object interactions, and long-horizon task progression.

C.2 Real-World Visualizations

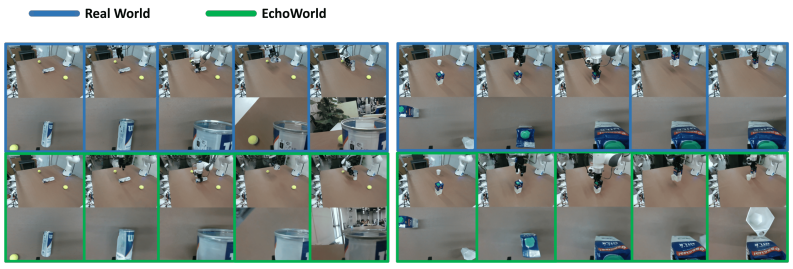

Fig. 3: Comparison between real-world execution and EchoArena rollout. Qualitative comparison between $\pi_{0.5}$ real robot rollouts and rollouts generated by EchoArena under the same setup. EchoArena reproduces key stages of task execution, including grasp, transport, and final object placement or reorientation, demonstrating strong visual and behavioral consistency with real-world policy execution.

References

1. Black, K., Brown, N., Driess, D., Esmail, A., Equi, M., Finn, C., Fusai, N., Groom, L., Hausman, K., Ichter, B., et al.: π_0: A vision-language-action flow model for general robot control. arXiv preprint arXiv:2410.24164 (2024)
2. Chen, T., Chen, Z., Chen, B., Cai, Z., Liu, Y., Li, Z., Liang, Q., Lin, X., Ge, Y., Gu, Z., et al.: Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. arXiv preprint arXiv:2506.18088 (2025)
3. Chi, C., Xu, Z., Feng, S., Cousineau, E., Du, Y., Burchfiel, B., Tedrake, R., Song, S.: Diffusion policy: Visuomotor policy learning via action diffusion. The International Journal of Robotics Research **44**(10-11), 1684–1704 (2025)
4. Guo, Y., Shi, L.X., Chen, J., Finn, C.: Ctrl-world: A controllable generative world model for robot manipulation. arXiv preprint arXiv:2510.10125 (2025)
5. Intelligence, P., Black, K., Brown, N., Darpinian, J., Dhabalia, K., Driess, D., Esmail, A., Equi, M., Finn, C., Fusai, N., Galliker, M.Y., Ghosh, D., Groom, L., Hausman, K., Ichter, B., Jakubczak, S., Jones, T., Ke, L., LeBlanc, D., Levine, S., Li-Bell, A., Mothukuri, M., Nair, S., Pertsch, K., Ren, A.Z., Shi, L.X., Smith, L., Springenberg, J.T., Stachowicz, K., Tanner, J., Vuong, Q., Walke, H., Walling, A., Wang, H., Yu, L., Zhilinsky, U.: $\pi_{0.5}$: a vision-language-action model with open-world generalization (2025), `https://arxiv.org/abs/2504.16054`
6. Li, X., Hsu, K., Gu, J., Pertsch, K., Mees, O., Walke, H.R., Fu, C., Lunawat, I., Sieh, I., Kirmani, S., Levine, S., Wu, J., Finn, C., Su, H., Vuong, Q., Xiao, T.: Evaluating real-world robot manipulation policies in simulation (2024), `https://arxiv.org/abs/2405.05941`
7. Liu, S., Wu, L., Li, B., Tan, H., Chen, H., Wang, Z., Xu, K., Su, H., Zhu, J.: Rdt-1b: a diffusion foundation model for bimanual manipulation. arXiv preprint arXiv:2410.07864 (2024)
8. Peebles, W., Xie, S.: Scalable diffusion models with transformers (2023), `https://arxiv.org/abs/2212.09748`
9. Podell, D., English, Z., Lacey, K., Blattmann, A., Dockhorn, T., Müller, J., Penna, J., Rombach, R.: Sdxl: Improving latent diffusion models for high-resolution image synthesis. arXiv preprint arXiv:2307.01952 (2023)
10. Walke, H.R., Black, K., Zhao, T.Z., Vuong, Q., Zheng, C., Hansen-Estruch, P., He, A.W., Myers, V., Kim, M.J., Du, M., et al.: Bridgedata v2: A dataset for robot learning at scale. In: Conference on Robot Learning. pp. 1723–1736. PMLR (2023)
11. Zhao, T.Z., Kumar, V., Levine, S., Finn, C.: Learning fine-grained bimanual manipulation with low-cost hardware. arXiv preprint arXiv:2304.13705 (2023)
12. Zhu, F., Wu, H., Guo, S., Liu, Y., Cheang, C., Kong, T.: Irasim: Learning interactive real-robot action simulators. arXiv preprint arXiv:2406.14540 (2024)