FM-Steer: Enhance Generalist Policies with Value-Guided Cascaded Denoising

Supplementary Material

7. FM-Steer Visualization Analysis

In this section, we first demonstrate the detailed inference process of FM-Steer through the dexterous Push-T task in Sec. 7.1. We then visualize two key designs of FM-Steer, value-guided test-time sampling and cascaded action denoising, in Sec. 7.2 and Sec. 7.3 to provide a comprehensive understanding of the method.

7.1. FM-Steer Workflow Visualization

To illustrate the detailed inference workflow of FM-Steer, we evaluate it on the Push-T task, which requires complex and contact-rich control to push the T block precisely. The Push-T task requires the policy to control a blue dot on a two-dimensional plane to push a gray T-shaped block into the green area. Since the action space is two-dimensional, we can visualize the actions predicted by FM-Steer as trajectories on a plane.

As shown in Fig. 7, we illustrate the detailed inference process of FM-Steer in the Push-T task. Specifically, during inference we sample the candidate actions $\mathbf{A}_t^{\tau_i}$ from the original VLA at time steps $t = 0, 30, 60, 90$ with a horizon of $H = 30$, and produce 10 candidate actions at each time step. The selected action with the highest value is conveyed to the Lite-Flow Denoiser and continuously denoised into the executed action, shown with a solid line. We can see that the final denoised action from the Lite-Flow Denoiser is smoother and more precise for accomplishing the task.

7.2. Value-Guided Sampling Visualization

Fig. 8 visualizes estimated state-action values of the candidate actions at different time steps in the LIBERO-GOAL benchmark. Since the action space in LIBERO is 7-dimensional, we used Principal Component Analysis (PCA) to project the high dimensional actions onto a two-dimensional plane. For each projected candidate action, we use different colors to represent their corresponding state-action values estimated by the flow verifier. These projected points together with their corresponding state-action values generate the value map shown in the figure. In the value map, yellow regions represent actions with high state-action values, while purple regions represent actions with low state-action values. Additionally, we also show the ground-truth actions from collected demonstrations in the value map for comparison.

By observing the positions of the ground-truth actions in the value map, we find that they are consistently located in high-value regions, which demonstrates that the flow verifier is capable of making reasonable estimates of the state-action values. We also find that the ground-truth actions are not located at the highest-value positions in the value map, which proves the flow verifier has not been overfitted to ground truth actions, but is able to estimate appropriate state-action values in the whole action space, selecting the optimal candidate action. Furthermore, by comparing value maps across different time steps, we observe adjacent time steps (step=0 and step=1) have similar value maps, while value maps at distant time steps (step=50 and step=100) exhibit significant differences. This demonstrates that the flow verifier can reasonably adjust its estimation of state-action values by capturing real-world dynamics to make smooth choices for robot control.

7.3. Cascaded Action Denoising Visualization

Fig. 9 visualizes the cascaded action denoising process in LIBERO-OBJECT. For the 7 dimensions in the action space (X, Y, Z, Roll, Pitch, Yaw, Gripper), we pair them into combinations for illustration, *i.e.*, X-Y, X-Z, Y-Z, and R-P. The plotted points are down-sampled from the actual denoised action sequences for accomplishing one task, where the blue points represent candidate actions $\mathbf{A}_t^{\tau_i}$ sampled from the original VLA, the red points represent optimal candidates $\mathbf{A}_t^{\tau^*}$ selected by the flow verifier, and the orange points represent final actions $\tilde{\mathbf{A}}_{t+kh}^1$ denoised by the Lite-Flow Denoiser. The red points and orange points are generally distributed within the region covered by blue points, while the distribution of orange points shows slight differences from the red points. This demonstrates that cascaded action denoising is actually refining selected actions from the original VLA with higher-frequency new observation inputs in the Lite-Flow Denoiser, achieving accurate, fluid, and delicate robot control.

8. Implementation Details

In this section, we provide further implementation details of FM-Steer, including the training details of the flow verifier and the hyperparameters used by the model in training.

8.1. Value Objective Functions

Formally, the goal of the flow verifier is to learn $Q_\theta(\mathbf{q}_t, \mathbf{A}_t)$ to approximate the state-action value function in a Markov decision process:

$$Q^\pi(\mathbf{q}_t, \mathbf{A}_t) = \frac{1}{1-\gamma} \sum_t \mathbb{E}_{\mathbf{A}_t \sim \pi(\mathbf{q}_t)} \left[\gamma^t r(\mathbf{q}_t, \mathbf{A}_t) \mid \mathbf{q}_{t_0} = \mathbf{q}_t,\ \mathbf{A}_{t_0} = \mathbf{A}_t \right]. \tag{6}$$

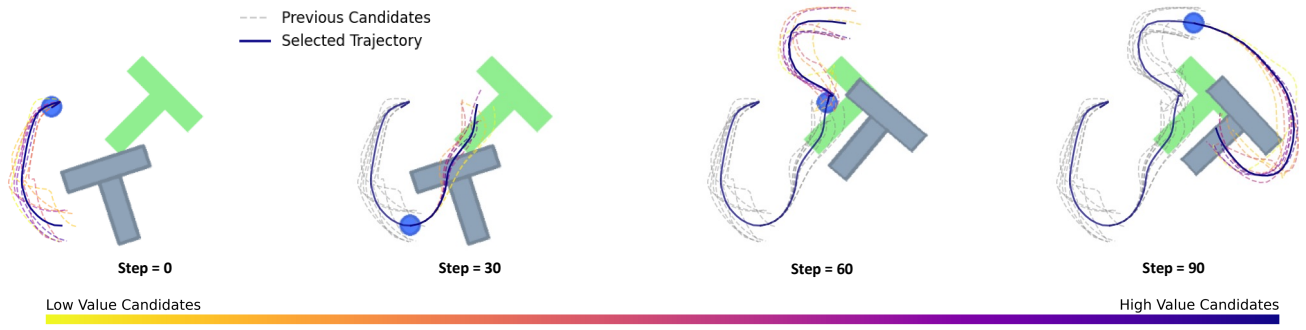


Figure 7. **Visualization of FM-Steer in Push-T.** We visualize the candidate actions $\mathbf{A}_t^{\tau_i}$ sampled from the original VLA with dashed lines and the final executed action $\tilde{\mathbf{A}}_{t+kh}^1$ from the Lite-Flow Denoiser with a solid line. The color intensity of each line indicates the magnitude of the state-action value $Q(\mathbf{q}_t, \mathbf{A}_t^{\tau_i})$ of the corresponding candidate.

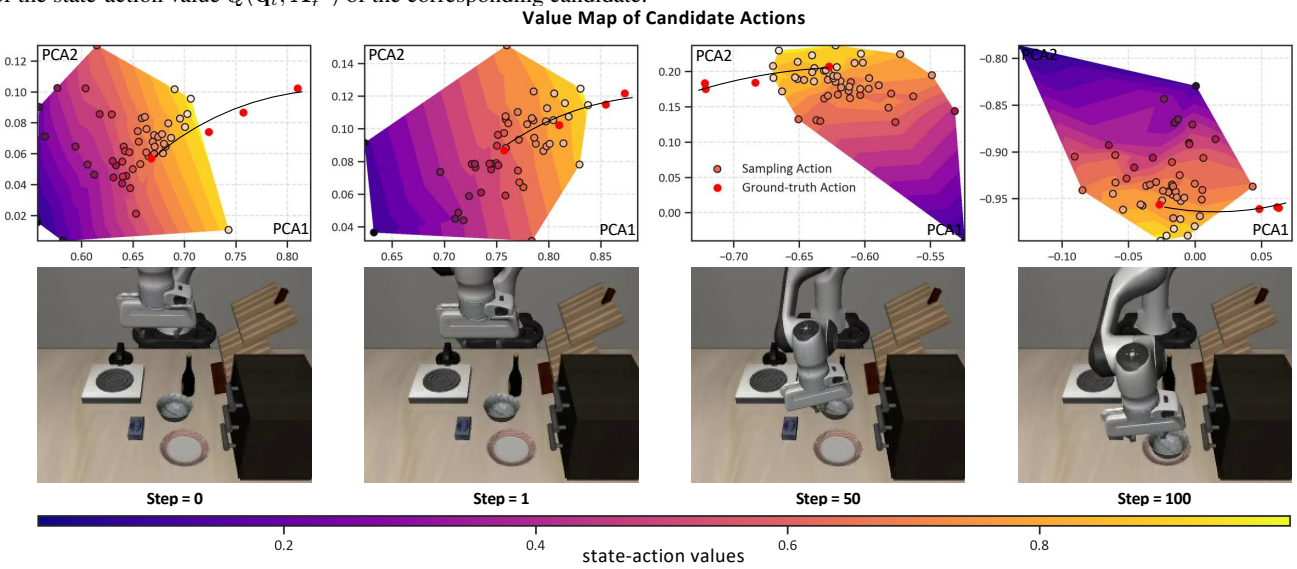


Figure 8. **Value Map of Candidate Actions.** The candidate actions $\mathbf{A}_t^{\tau_i}$ sampled from the original VLA and the ground-truth actions $\mathbf{A}_t^{\text{GT}}$ are projected into the same two-dimensional space using Principal Component Analysis (PCA). The color intensity indicates the magnitude of the state-action value $Q(\mathbf{q}_t, \mathbf{A}_t^{\tau_i})$ of each candidate.

The Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, \rho, \gamma)$, where $\mathcal{S}, \mathcal{A}$ denote the state and action spaces, while $P(\mathbf{q}'|\mathbf{q}, \mathbf{A})$ and $r(\mathbf{q}_t, \mathbf{A}_t)$ are the dynamics and reward functions. $\rho(\mathbf{q})$ denotes the initial state distribution, and $\gamma \in (0, 1)$ denotes the discount factor. The training objective of the flow verifier is to minimize the Bellman error with a regularization term $\mathcal{R}(\theta)$, which is defined as:

$$\min_{\theta} \ \alpha\,\mathcal{R}(\theta) + \frac{1}{2}\,\mathbb{E}_{\mathbf{q}_t, \mathbf{A}_t, \mathbf{q}_t' \sim \mathcal{D}}\Big[\big(Q_\theta(\mathbf{q}_t, \mathbf{A}_t) - \mathcal{B}^\pi \bar{Q}(\mathbf{q}_t, \mathbf{A}_t)\big)^2\Big]. \tag{7}$$

The second term in Eq. (7) is the standard TD error [5, 6, 10], where $Q_\theta(\mathbf{q}_t, \mathbf{A}_t)$ is the output of the flow verifier, and $\mathcal{B}^\pi \bar{Q}(\mathbf{q}_t, \mathbf{A}_t)$ is the backup operator applied to the delayed target Q-network $\bar{Q}$: $\mathcal{B}^\pi \bar{Q}(\mathbf{q}_t, \mathbf{A}_t) := r(\mathbf{q}_t, \mathbf{A}_t) + \gamma \mathbb{E}_{\mathbf{A}_t' \sim \pi(\mathbf{A}_t'|\mathbf{q}_t')}[\bar{Q}(\mathbf{q}_t', \mathbf{A}_t')]$, which can be calculated from the offline dataset $\mathcal{D} = \{(\mathbf{q}_t, \mathbf{A}_t, r, \mathbf{q}_t')\}$. The $\mathcal{R}(\theta)$ in Eq. (7) is a calibrated conservative regularizer that aims to prevent overestimation in the Q-values for OOD actions by penalizing the Q-values, and compensating for this pessimism on actions seen in the training dataset, and α is a hyperparameter to control the conservative penalty. Specifically, the regularization term $\mathcal{R}(\theta)$ is defined as:

$$\mathcal{R}(\theta) := \mathbb{E}_{\mathbf{q}_t \sim \mathcal{D},\, \mathbf{A}_t' \sim \pi(\cdot|\mathbf{q}_t)}\Big[\max\big(Q_\theta(\mathbf{q}_t, \mathbf{A}_t'),\, Q^\mu(\mathbf{q}_t, \mathbf{A}_t')\big)\Big] - \mathbb{E}_{\mathbf{q}_t, \mathbf{A}_t \sim \mathcal{D}}\big[Q_\theta(\mathbf{q}_t, \mathbf{A}_t)\big]. \tag{8}$$

Here, $Q^\mu(\mathbf{q}_t, \mathbf{A}_t)$ is the value function of the calibrated policy μ, which is the policy used to collect the offline dataset $\mathcal{D}$.

8.2. Training and Inference Hyperparameters

LIBERO In LIBERO, FM-Steer takes images from a third-person camera, a wrist camera, and the robot state as input. We set the chunk size of the original VLA to 16 and the

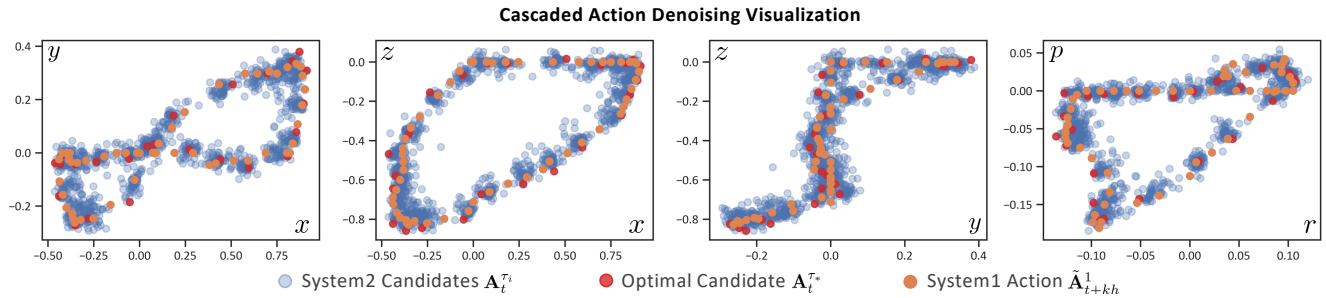


Figure 9. **Cascaded Action Denoising on Different Action Dimensions.** We visualize denoised actions grouped to coordinate pairs (X-Y, X-Z, Y-Z, and R-P) from the 7-dimensional action space (X, Y, Z, Roll, Pitch, Yaw, Gripper). For each group, we display the candidate actions $\mathbf{A}_t^{\tau_i}$ sampled from the original VLA, the optimal candidate $\mathbf{A}_t^{\tau^*}$, and final action $\tilde{\mathbf{A}}_{t+kh}^1$ denoised by the Lite-Flow Denoiser.

chunk size of the Lite-Flow Denoiser's output to 8. The model was trained using 8 GPUs with a batch size of 16.

SimplerEnv In SimplerEnv, FM-Steer takes images from a third-person camera and the robot state as input. In Bridge tasks, we set the chunk size of the original VLA to 8 and the chunk size of the Lite-Flow Denoiser's output to 4. In Google Robot tasks, we set the chunk size of the original VLA to 4 and the chunk size of the Lite-Flow Denoiser's output to 2. The model was trained using 8 GPUs with a batch size of 32.

Franka In real-world experiments on the Franka Emika Panda, FM-Steer takes images from a third-person camera and the robot state as input. We set the chunk size of the original VLA to 16 and the chunk size of the Lite-Flow Denoiser's output to 8. The model was trained using 4 GPUs with a batch size of 32.

WidowX In real-world experiments on WidowX 250s, FM-Steer uses the same training settings as in the Bridge environment of SimplerEnv, taking images from a third-person camera and robot state as input. The chunk size of the original VLA is 8 and chunk size of the Lite-Flow Denoiser is 4. The model is trained using 8 GPUs with a batch size of 32.

AgiBot G-1 In real-world experiments on AgiBot G-1, FM-Steer takes images from the head camera and wrist cameras on both arms along with robot state as input. We set the chunk size of the original VLA to 30 and the chunk size of the Lite-Flow Denoiser's output to 15. The model was trained using 8 GPUs with a batch size of 8.

9. Experiment Details

In this section, we provide experimental details including evaluation setups and additional test results. In Sec. 9.1, we provide detailed descriptions of the test setups for standard simulation benchmarks and the implementation details of all comparison methods, along with the detailed test results. In Sec. 9.2, we introduce the detailed setups of the test tasks on real-robot platforms and the evaluation standards, and provide the detailed test results.

9.1. Simulation Benchmark Details

Simulation Benchmark Setup. In LIBERO, all methods use third-person camera, wrist camera, and robot state as input, where the results of Diffusion Policy and OpenVLA-OFT are from the technical report of OpenVLA-OFT [8], the results of π_0 [2] and π_0-FAST [12] are provided by Physical Intelligence's open-source repository, and the results of GR00T N1 are obtained by training and testing on the LIBERO dataset using its open source code. In SimplerEnv, test results of RT-1 [3], RT-1-X [4], RT-2-X [4], Octo [11], OpenVLA [7], HPT [14], TraceVLA [15], RoboVLM [9], SpatialVLA [13] come from their official technical reports. The results of GR00T [1], π_0-FAST [12], and π_0 [2] are obtained by us fine-tuning and testing them on the corresponding datasets using their official code.

Detailed Results of Simulation Benchmark. Since the Google Robot tasks in SimplerEnv include various test settings such as environment layout, object position, and texture variations, we provide more detailed results in Tab. 4.

9.2. Real-World Evaluation Details

Real-World Evaluation Setup. In this section, we provide detailed descriptions of task setups on three real-world robot platforms: WidowX 250s, Franka Emika Panda, and AgiBot G-1.

As shown in Fig. 10, the detailed task specifications on WidowX 250s are:

- **Put eggplant in the basket**: A complex task requiring the robot to identify and pick an eggplant from a sink containing multiple vegetables, then place it in a yellow basket. This task evaluates object discrimination and spatial awareness.
- **Put carrot on the plate**: The robot needs to perform a pick-and-place task by grasping a carrot from the sink and placing it on a plate, assessing both grasping precision and placement accuracy.
- **Close microwave**: The robot must close a toy microwave door positioned at various angles (30°, 45°, 60°, and 90°),

Table 4. **Detailed Google Robot Results on SimplerEnv.** We report the results for the Google Robot benchmark under the variant-aggregation and visual-matching settings.

Google Robot Evaluation Setup	Policy	Pick Coke Can				Move Near	Open / Close Drawer			#Overall
		Horizontal Laying	Vertical Laying	Standing	Average	Average	Open	Close	Average	Average
Variant Aggregation	RT-1 (begin)	2.2%	1.3%	3.1%	2.2%	4.0%	0.5%	13.2%	6.9%	4.4%
	RT-1 (15%)	92.0%	70.4%	81.3%	81.2%	44.6%	21.2%	32.3%	26.8%	50.9%
	RT-1 (converged)	96.9%	76.0%	96.4%	89.8%	50.0%	27.0%	37.6%	32.3%	57.4%
	RT-1-X	56.9%	20.4%	69.8%	49.0%	32.3%	6.9%	51.9%	29.4%	36.9%
	RT-2-X	82.2%	75.4%	89.3%	82.3%	79.2%	33.3%	37.2%	35.3%	65.6%
	Octo-Base	0.5%	0.0%	1.3%	0.6%	3.1%	0.0%	2.1%	1.1%	1.6%
	OpenVLA	71.1%	27.1%	65.3%	54.5%	47.7%	15.8%	19.5%	17.7%	40.0%
	TraceVLA	—	—	—	60.0%	56.4%	—	—	31.0%	49.1%
	RoboVLM	93.8%	49.8%	83.1%	75.6%	60.0%	2.6%	18.5%	10.6%	48.7%
	SpatialVLA	93.3%	78.2%	92.4%	88.0%	72.7%	28.6%	55.0%	41.8%	67.5%
	π_0	82.0%	58.0%	85.6%	75.2%	63.7%	18.0%	33.2%	25.6%	54.8%
	π_0-FAST	84.0%	63.0%	85.8%	77.6%	68.2%	24.0%	38.6%	31.3%	59.0%
	FM-Steer	99.0%	96.0%	99.0%	98.0%	79.7%	38.0%	51.2%	44.6%	74.1%
Visual Matching	RT-1 (Begin)	5.0%	0.0%	3.0%	2.7%	5.0%	0.0%	27.8%	13.9%	7.2%
	RT-1 (15%)	86.0%	79.0%	48.0%	71.0%	35.4%	46.3%	66.7%	56.5%	54.3%
	RT-1 (Converged)	96.0%	90.0%	71.0%	85.7%	44.2%	60.1%	86.1%	73.1%	67.7%
	RT-1-X	82.0%	33.0%	55.0%	56.7%	31.7%	29.6%	89.1%	59.4%	49.3%
	RT-2-X	74.0%	74.0%	88.0%	78.7%	77.9%	15.7%	34.3%	25.0%	60.5%
	Octo-Base	21.0%	21.0%	9.0%	17.0%	4.2%	0.9%	44.4%	22.7%	14.6%
	OpenVLA	27.0%	3.0%	19.0%	16.3%	46.2%	19.4%	51.8%	35.6%	32.7%
	TraceVLA	—	—	—	28.0%	53.7%	—	—	57.0%	46.2%
	RoboVLM	94.0%	47.0%	91.0%	77.3%	61.7%	33.3%	53.1%	43.2%	60.7%
	SpatialVLA	85.0%	76.0%	97.0%	86.0%	77.9%	50.0%	64.8%	57.4%	73.8%
	HPT	—	—	—	56.0%	60.0%	—	—	24.0%	46.7%
	π_0	76.0%	57.0%	85.1%	72.7%	65.3%	30.0%	46.6%	38.3%	58.8%
	π_0-FAST	79.0%	61.0%	85.9%	75.3%	67.5%	34.0%	51.8%	42.9%	61.9%
	FM-Steer	99.0%	93.0%	99.0%	97.0%	80.4%	52.0%	65.6%	58.8%	78.7%

testing the model's capability to manipulate articulated objects in different configurations.

- **Lift red pepper**: A basic pick task requiring the robot to grasp and lift a red pepper from the sink, designed to evaluate the model's object localization accuracy.
- **Put green cup on the pink cloth**: This task suite comprises two scenarios testing vertical spatial understanding. In the first scenario, the robot grasps a green cup positioned either on a stove or elevated on a yellow block. In the second scenario, the cup is placed either at the bottom of a sink or elevated on a bowl. This variation in object heights challenges the model's ability to adapt its manipulation strategy according to spatial configurations.
- **Put purple cup on the white plate**: The robot must identify and transfer a purple cup to a white plate within a sink containing multiple plates, testing color recognition and precise manipulation.

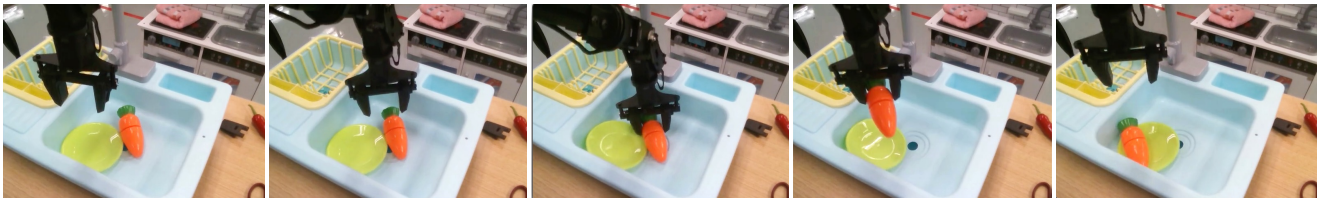
As shown in Fig. 11, the detailed task specifications on the Franka Emika Panda are:

- **Kitchen Banana**: A pick-and-place task where the robot must transfer a banana from the table to a basket. With only 50 human demonstrations, this task evaluates model performance with limited data.
- **Kitchen Pot**: Another pick-and-place task requiring the robot to grasp a white bowl from the right side of the table and place it on a cutting board, trained with 100 human demonstrations.
- **Tea**: The robot must push a teapot handle from a perpendicular to a parallel position relative to the desktop, using its gripper tip. This task tests the manipulation of revolute joints and includes 50 human demonstrations.
- **Plush Toy**: The robot must identify and grasp the nearest plush toy among two options and place it on a green car. To rigorously assess spatial understanding, we systematically vary the relative positions of plush toys in testing.
- **Three Cube**: An instruction-following task where the robot must identify and place a specifically colored cube (red, green, or blue) onto a green car. Training included 50 demonstrations for each color, totaling 150.

As shown in Fig. 12, the detailed task specifications on AgiBot G-1 are:

- **Restock the hanging basket area**: This task requires the

Put eggplant in the basket: Put the eggplant in the basket.



Put carrot on the plate: Put the carrot on the plate.

Close microwave: Close the microwave.

Lift red pepper: Lift the red pepper.

Put green cup on the pink cloth : Put the green cup on the pink cloth.

Put purple cup on the white plate : Put the purple cup on the white plate.

Figure 10. **Evaluation Setup of WidowX 250s.** We evaluated models on six tasks on WidowX 250s to verify the model's learning ability on a large multi-task manipulation dataset.

robot to grab snacks from a cart and place them at a designated location on the shelf. This task includes different types of snacks, different placements on the shelf, and interfering objects in the cart to verify the generalization of the model.

- **Pour water**: This is a long-horizon task that requires the robot to first grasp the handle of the teapot, lift the teapot and accurately pour the water from the teapot into the cup, and then put the teapot back on the mat after the cup is full. This task involves changing the material of the cup, as well as the position of the teapot and the cup. To complete this challenging task, the robot needs to accurately

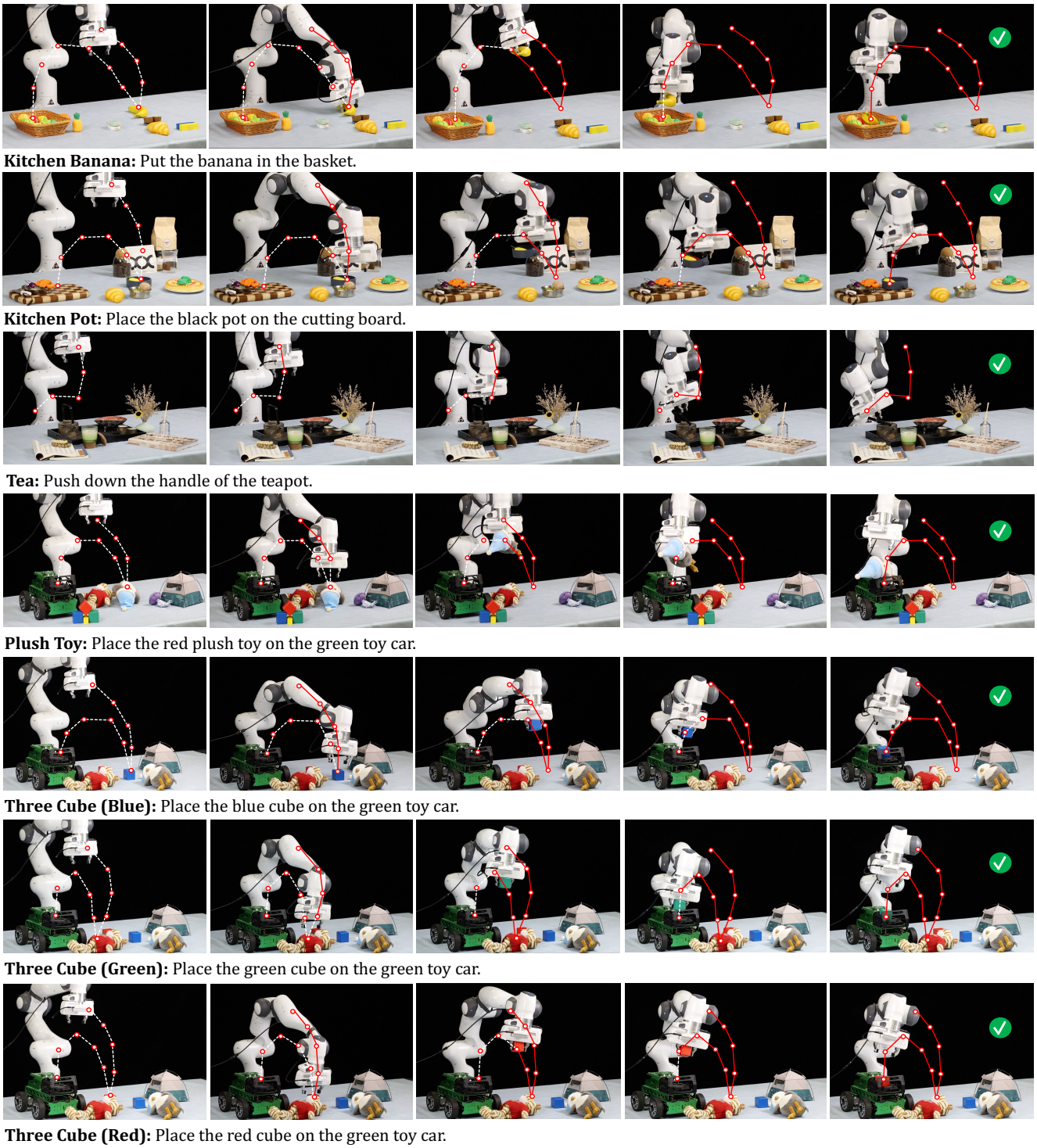


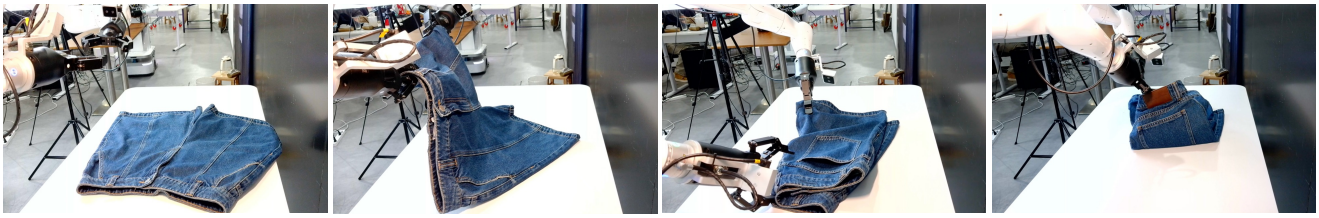
Figure 11. **Evaluation Setup of the Franka Emika Panda.** We evaluate policies on the Franka Emika Panda robot with five tasks, including instruction following, articulated manipulation, and pick-and-place tasks.

identify the location of the cup and pour water into it, and it needs to correctly sense the water level in the cup to avoid spilling water on the table.

- **Fold Shorts**: This is a long-horizon task that requires the collaboration of both arms and involves the manipulation of flexible objects. In this task, the robot first needs to accurately identify the position of the shorts and use the grippers on both arms to fold the shorts for the first time. After completing the first fold, the robot needs to confirm the current state of the shorts again and perform a second

Restock the hanging basket area: The robot is in front of the snack shelf, with the shopping cart positioned between the snack shelf and the robot. The snacks that need to be restocked are in the box inside the shopping cart.

Pour water: Lift the kettle on the table with the right arm. Pour two-thirds of the water into the cup on the table with the right arm. Place the held kettle on the coaster on the table with the right arm.



Fold Shorts: Grasp the bottom legs and waist of the shorts with both hands and fold them over the top legs and waist. Grasp the waistband of the shorts and fold it to the leg with both arms.

Pass the water: Pick up the green mineral water on the table with the right arm. Pass the picked-up green mineral water to the guest with the right arm.

Figure 12. **Evaluation Setup of AgiBot G-1.** We evaluate policies on four challenging tasks on AgiBot G-1 to test the ability to control a humanoid robot on dexterous and long-horizon tasks.

fold. During the testing of this task, we used shorts of different colors and materials to verify the model's long-horizon manipulation capabilities for flexible objects of different shapes.

- **Pass the water**: This task is designed to test the model's ability to follow instructions and collaborate with humans. In this task, the robot needs to grab the correct bottle according to the language instructions and hand it to the human. We used different types of bottles in the test, and we also arbitrarily adjusted the position of the bottles on the white table shown in the figure to verify the generalization of the robot to the position of objects.

Detailed Results of Real-World Evaluation. To ensure the reliability of real robot platform test results, we used standardized evaluation metrics. For simple tasks such as "Lift red pepper" on WidowX, we only tracked the overall task completion success rate. For more general tasks, such as "Kitchen Banana" on Franka, we tracked both partial and overall success rates, meaning the robot's success rate in grasping the banana and placing it at the designated location. For more complex long-horizon tasks, such as "Pour water" on AgiBot G-1, we tracked the success rate of each subtask and the overall task success rate, including the robot's success in grasping the kettle, pouring water from the kettle into the cup, and placing the kettle on the pad. In addition, to verify the model's generalization capability, we conducted tests under different lighting conditions, with various types of objects, different environmental

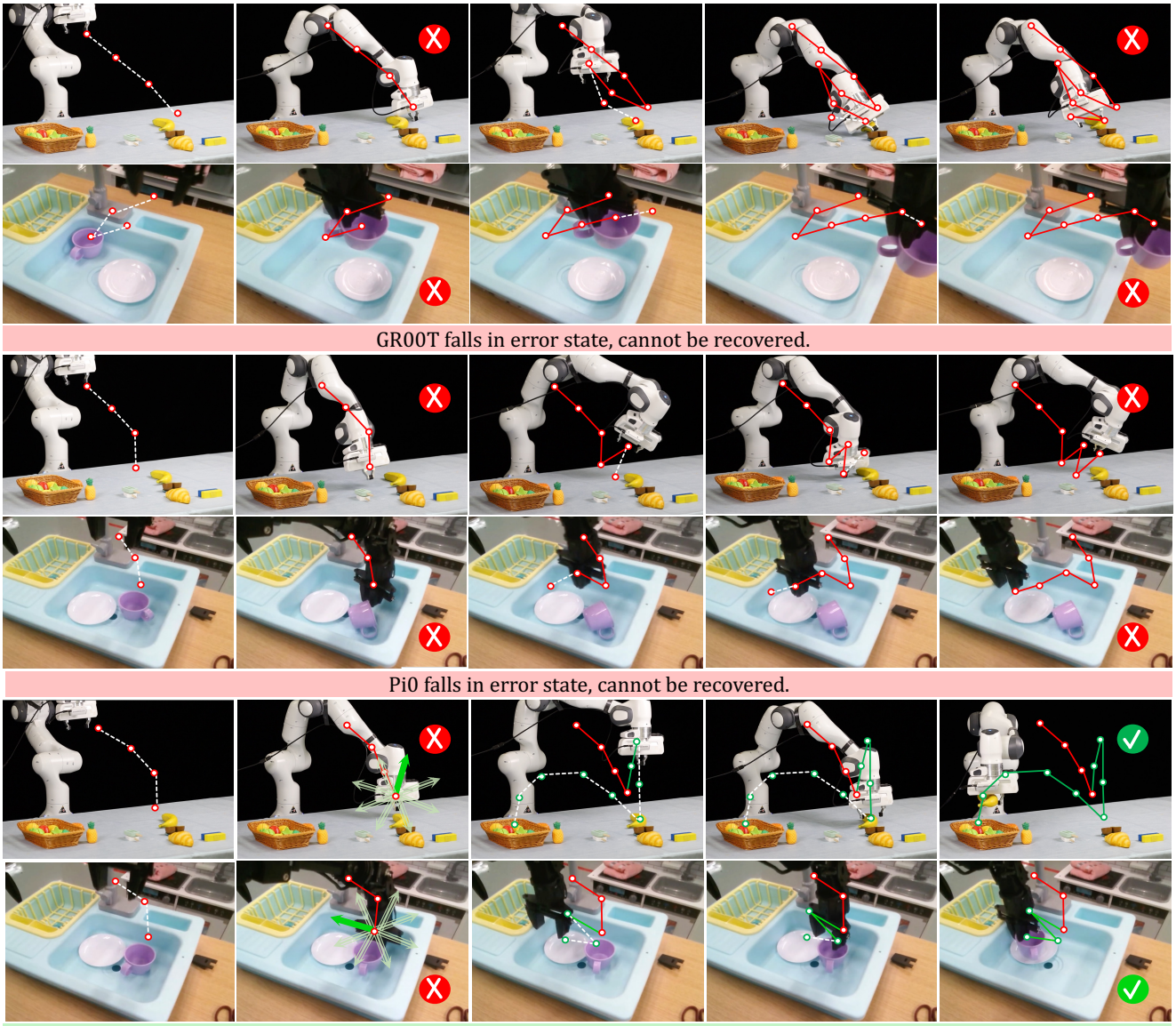


Figure 13. **Failure Recovery of FM-Steer.** When a failure occurs, such as missing the grasping position, other policies fall into a failure state, whereas FM-Steer selects the correct action through value-guided test-time sampling to recover from the failure state and successfully complete the task.

layouts, and diverse language instructions. For example, in the "put green cup on the pink cloth" task on the WidowX robot, the initial position of the green cup was significantly adjusted. In the "Fold Shorts" task on AgiBot G-1, shorts of different colors and materials were used for testing.

As shown in Fig. 13, we also observe that FM-Steer can recover from failures during evaluation. Compared with other methods, FM-Steer escapes failure states more often and completes the task successfully. Benefiting from value-guided test-time sampling, when the model falls into a failure state, such as missing the correct grasp position, FM-Steer can select the correct action from a variety of candidate actions to recover and gradually guide the robot back to a successful trajectory. For common imitation policies such as π_0 and GR00T, when they enter an error state, since the observation of the error state does not appear in their training dataset, these models are easily trapped in the error state and cannot recover, resulting in the failure of the final task. For FM-Steer, although the error-state observation also does not appear in the training dataset, the model can still include a recovery action by repeatedly sampling candidate actions that are not yet fully denoised and then selecting the best

one according to the state-action value estimated by the flow verifier, thereby achieving strong failure-recovery ability.

References

[1] Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025. 15

[2] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024. 15

[3] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022. 15

[4] Open X-Embodiment Collaboration, 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. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2024. 15

[5] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning (ICML)*, pages 1587–1596, 2018. 14

[6] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018. 14

[7] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024. 15

[8] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025. 15

[9] Xinghang Li, Peiyan Li, Minghuan Liu, Dong Wang, Jirong Liu, Bingyi Kang, Xiao Ma, Tao Kong, Hanbo Zhang, and Huaping Liu. Towards generalist robot policies: What matters in building vision-language-action models. *arXiv preprint arXiv:2412.14058*, 2024. 15

[10] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015. 14

[11] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024. 15

[12] Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025. 15

[13] Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025. 15

[14] Lirui Wang, Xinlei Chen, Jialiang Zhao, and Kaiming He. Scaling proprioceptive-visual learning with heterogeneous pre-trained transformers. In *Proceedings of the Conference on Neural Information Processing System (NeurIPS)*, 2024. 15

[15] Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. *arXiv preprint arXiv:2412.10345*, 2024. 15