

A Technical Appendices and Supplementary Material

In this appendix, we first present further details on the model architecture of FreqPolicy in Sec. A.1. Next, Sec. A.2 provides additional visualizations and frequency-domain analyses of various action chunks. We then supplement the simulation experiments with additional information in Sec. A.3, along with details on integrating FreqPolicy into VLA models in Sec. A.4. Finally, Sec. A.5 describes the setup and results of the real-world experiments.

A.1 More Model Configuration

As a Visuomotor Policy. Following previous works [11, 76, 28, 42, 78], we adopt a standard 1D CNN-based U-Net architecture as the backbone of FreqPolicy to ensure a fair comparison with existing models. FreqPolicy is designed to accept both 2D and 3D observations as input. For 2D images, we employ ResNet-18 as the visual encoder. For 3D input, we follow [76, 78] and use a lightweight MLP to encode the input point cloud.

As a Head of the VLA Model. FreqPolicy can also serve as a policy head for existing VLA models, as long as the underlying VLA is capable of predicting action vector fields. In this work, we integrate FreqPolicy into OpenVLA [33], following the setup in [32], where the predicted action vector field is obtained by applying a nonlinear mapping to the noise-conditioned latent features.

A.2 Extended Frequency Analysis for Robotic Manipulation

In Fig. 4, we present visualizations of action chunks from different real-world and simulation scenarios, including the observed images, the corresponding multi-dimensional temporal action signals, and the transformed DCT frequency coefficients. In the simulation scenario shown in Fig. 4a, the Franka robot transitions from “approaching the blue can” to “grasping the can”, during which the expected action chunk contains high-frequency variations associated with gripper motion. Similarly, in Fig. 4b, as the robot moves from “approaching the stove” to “placing down the kettle”, the gripper opening action introduces high-frequency signals, while other control signals remain relatively smooth. In the real-world scenario shown in Fig. 4c, during the process of picking up the doll, the Franka robot exhibits relatively smooth motion transitions, with the action signal primarily dominated by low-frequency components. Hence, based on the above observations of different action chunks and their frequency-domain characteristics, we justify the motivation of the adaptive frequency coefficient loss introduced in Sec 3. This loss enables the model to focus more effectively on the frequency components that exhibit meaningful variation across diverse action chunks.

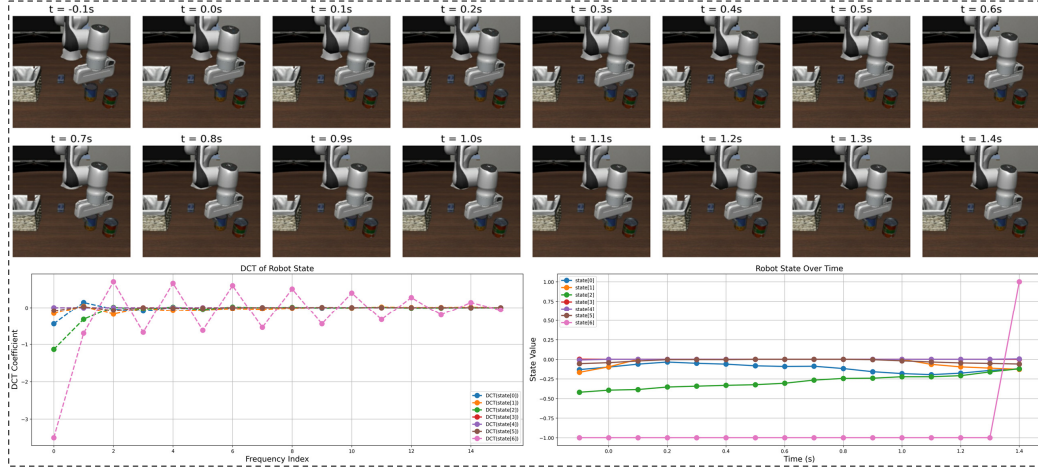
A.3 More Details on Visuomotor Policy Simulation

A.3.1 More Details on Simulation with 2D Inputs

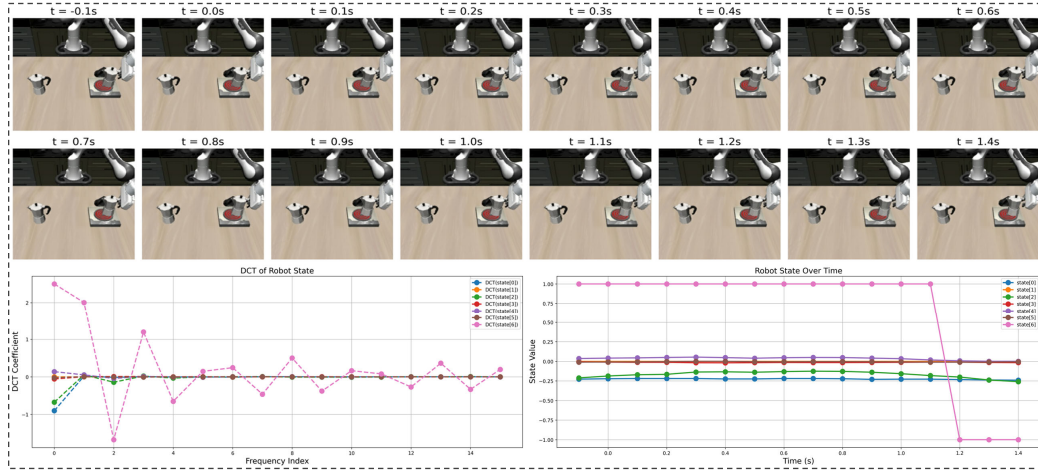
Implementation Details. We implement the RectifiedFlow [40, 38], ConsistencyPolicy [53], and Consistency-FM [68] to comprehensively evaluate our proposed FreqPolicy. For RectifiedFlow, we adopt a 1D CNN-based U-Net to predict action vector fields from observations. Following [6], we sample time steps t from a Beta distribution during training to interpolate intermediate states. For ConsistencyPolicy, we follow the procedure in [53], where an EDM [29] teacher model is first trained in the initial stage, and then a student model is distilled for one-step action generation using the CTM objective [31]. For Consistency-FM, we also use a 1D CNN-based U-Net to predict action vector fields from observations. We then sample different time steps r and s from a uniform schedule and apply velocity field consistency constraints between them. To train all models, we use observations from the past 2 time steps as input and predict a 16-step action chunk, from which the first 8 steps are selected for execution. All models are trained using a batch size of 128 with the AdamW optimizer and a learning rate of $1.0e-4$. Training is conducted for 1000 epochs on a single NVIDIA A100 GPU. For ConsistencyPolicy, the student model is trained for 450 epochs.

A.3.2 More Details on Simulation with 3D Inputs

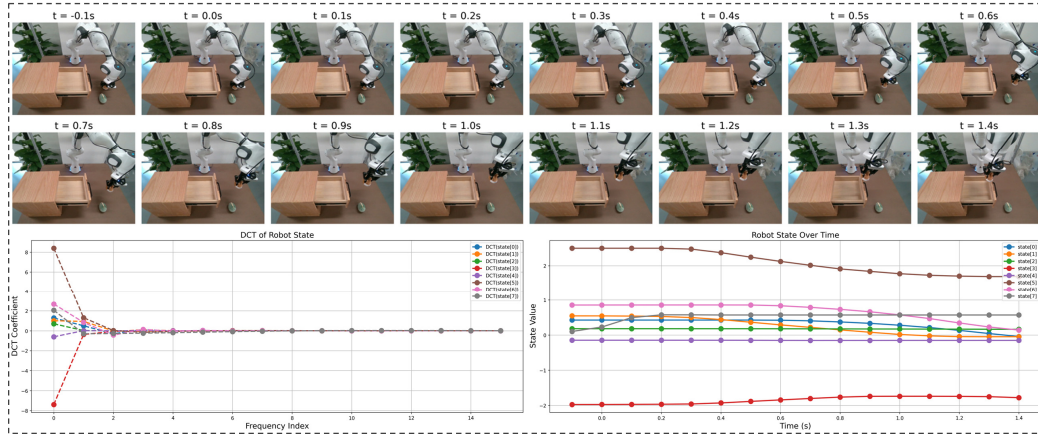
Implementation Details. Following the prior works [76, 78], we use a lightweight MLP to encode the point cloud and use a 1D CNN-based U-Net to predict the action vector field. To train the FreqPolicy model, we use a batch size of 128 and the AdamW optimizer with a learning rate of



(a) Visualization of an action chunk from the Franka simulation dataset. When the robot transitions from “reach the blue can” to “grasping it”, the change in gripper introduces high-frequency signal variations. Other states exhibit no significant changes and tend to be smoother.



(b) Visualization of an action chunk from the Franka simulation dataset. During the transition from “approaching the stove” to “placing down the kettle”, the gripper exhibits high-frequency variations, while other states remain nearly unchanged.



(c) Visualization of an action chunk from the Franka robot demonstrations. Once the robot successfully grasps the doll and begins placing it into the drawer, the overall motion exhibits relatively slow changes under a 30 Hz action sampling rate, dominated by low-frequency signals.

Figure 4: Visualization of the spectral and temporal signals across different action chunks.

Table 5: We report the evaluation details of the 53 challenging tasks from Adroit [55] and Meta-World [74] under 3 random seeds, and report the mean success rate (%) and standard deviation for each task. Tasks marked with an asterisk * indicate re-implemented versions. Compared to ManiCM [42], a one-step action generation model using consistency distillation based on diffusion-based policies, our method achieves superior performance. Similarly, we achieve an overall improvement of 9.4% over SDM [28] that adopts variational score distillation [72, 71] to achieve a one-step diffusion policy. Finally, in comparison to the flow-based policy FlowPolicy [78], which uses only spatial-domain consistency loss, FreqPolicy still leads by 2.8% gains, demonstrating the effectiveness of our frequency consistency constraint.

Alg / Task	Adroit			Meta-World (Easy)		
	Adroit Hammer	Adroit Door	Adroit Pen	Button Press	Coffee Button	Plate Slide Back Side
Diffusion Policy	45 ± 5	37 ± 2	13 ± 2	99 ± 1	99 ± 1	100 ± 0
3D Diffusion Policy	100 ± 0	75 ± 3	48 ± 3	100 ± 0	100 ± 0	100 ± 0
ManiCM	100 ± 0	68 ± 1	49 ± 4	100 ± 0	100 ± 0	100 ± 0
SDM Policy	100 ± 0	73 ± 2	49 ± 4	100 ± 0	100 ± 0	100 ± 0
FlowPolicy*	97 ± 3	62 ± 6	49 ± 5	100 ± 0	100 ± 0	100 ± 0
Ours	98 ± 2	68 ± 5	52 ± 4	100 ± 0	100 ± 0	100 ± 0

Alg / Task	Meta-World (Easy)					
	Button Press Topdown	Button Press Topdown Wall	Button Press Wall	Peg Unplug Side	Door Close	Door Lock
Diffusion Policy	98 ± 1	96 ± 3	97 ± 3	74 ± 3	100 ± 0	86 ± 8
3D Diffusion Policy	99 ± 1	96 ± 3	100 ± 0	93 ± 3	100 ± 0	96 ± 3
ManiCM	100 ± 0	96 ± 2	98 ± 3	71 ± 15	100 ± 0	98 ± 2
SDM Policy	98 ± 2	99 ± 1	100 ± 0	74 ± 19	100 ± 0	96 ± 2
FlowPolicy*	100 ± 0	100 ± 0	100 ± 0	88 ± 5	100 ± 0	100 ± 0
Ours	100 ± 0	100 ± 0	100 ± 0	87 ± 4	100 ± 0	100 ± 0

Alg Task	Meta-World (Easy)						
	Door Open	Door Unlock	Drawer Close	Drawer Open	Faucet Close	Faucet Open	Handle Press
Diffusion Policy	98 ± 3	98 ± 3	100 ± 0	93 ± 3	100 ± 0	100 ± 0	81 ± 4
3D Diffusion Policy	100 ± 0	100 ± 0	100 ± 0	100 ± 0	100 ± 0	100 ± 0	52 ± 8
ManiCM	100 ± 0	82 ± 16	100 ± 0	100 ± 0	100 ± 0	100 ± 0	10 ± 10
SDM Policy	100 ± 0	100 ± 0	100 ± 0	100 ± 0	99 ± 1	100 ± 0	28 ± 11
FlowPolicy*	100 ± 0	100 ± 0	100 ± 0	100 ± 0	99 ± 0	100 ± 0	25 ± 8
Ours	100 ± 0	100 ± 0	100 ± 0	100 ± 0	100 ± 0	100 ± 0	22 ± 5

Alg Task	Meta-World (Easy)							
	Handle Press Side	Handle Pull Side	Lever Pull	Plate Slide	Plate Slide Back	Dial Turn	Reach	Reach Wall
Diffusion Policy	100 ± 0	23 ± 17	49 ± 5	83 ± 4	99 ± 0	63 ± 10	18 ± 2	59 ± 7
3D Diffusion Policy	0 ± 0	82 ± 5	84 ± 8	100 ± 0	100 ± 0	91 ± 0	26 ± 3	74 ± 3
ManiCM	0 ± 0	48 ± 11	82 ± 7	100 ± 0	96 ± 5	84 ± 2	33 ± 3	62 ± 5
SDM Policy	0 ± 0	68 ± 6	84 ± 9	100 ± 0	100 ± 0	88 ± 3	34 ± 3	80 ± 1
FlowPolicy*	100 ± 0	50 ± 6	74 ± 5	95 ± 3	100 ± 0	81 ± 4	29 ± 10	69 ± 6
Ours	100 ± 0	59 ± 6	80 ± 5	96 ± 2	100 ± 0	91 ± 5	32 ± 10	72 ± 5

Alg Task	Meta-World (Easy)			Meta-World (Medium)				
	Plate Slide Side	Window Close	Window Open	Basketball	Bin Picking	Box Close	Coffee Pull	Coffee Push
Diffusion Policy	100 ± 0	100 ± 0	100 ± 0	85 ± 6	15 ± 4	30 ± 5	34 ± 7	67 ± 4
3D Diffusion Policy	100 ± 0	100 ± 0	99 ± 1	100 ± 0	56 ± 14	59 ± 5	79 ± 2	96 ± 2
ManiCM	100 ± 0	100 ± 0	80 ± 26	4 ± 4	49 ± 17	73 ± 2	68 ± 18	96 ± 3
SDM Policy	100 ± 0	100 ± 0	78 ± 18	28 ± 26	55 ± 13	61 ± 3	72 ± 9	97 ± 2
FlowPolicy*	100 ± 0	100 ± 0	100 ± 0	85 ± 7	45 ± 7	56 ± 4	89 ± 3	94 ± 2
Ours	100 ± 0	100 ± 0	100 ± 0	77 ± 4	40 ± 7	61 ± 5	84 ± 5	95 ± 3

Alg Task	Meta-World (Medium)						Meta-World (Hard)		
	Hammer	Peg Insert Side	Push Wall	Soccer	Sweep	Sweep Into	Assembly	Hand Insert	Pick Out of Hole
Diffusion Policy	15 ± 6	34 ± 7	20 ± 3	14 ± 4	18 ± 8	10 ± 4	15 ± 1	0 ± 0	0 ± 0
3D Diffusion Policy	100 ± 0	79 ± 4	78 ± 5	23 ± 4	92 ± 4	38 ± 9	100 ± 0	28 ± 8	44 ± 3
ManiCM	98 ± 2	75 ± 8	31 ± 7	27 ± 3	54 ± 16	37 ± 13	87 ± 3	28 ± 15	30 ± 16
SDM Policy	98 ± 2	83 ± 5	83 ± 4	25 ± 2	90 ± 6	32 ± 15	100 ± 0	24 ± 14	34 ± 24
FlowPolicy*	97 ± 3	69 ± 5	56 ± 6	24 ± 6	91 ± 3	26 ± 5	91 ± 2	23 ± 5	33 ± 3
Ours	95 ± 2	70 ± 4	64 ± 11	27 ± 4	88 ± 4	23 ± 4	100 ± 0	18 ± 3	40 ± 4

Alg Task	Meta-World (Hard)			Meta-World (Very Hard)					Average
	Pick Place	Push	Push Back	Shelf Place	Disassemble	Stick Pull	Stick Push	Pick Place Wall	
Diffusion Policy	0 ± 0	30 ± 3	0 ± 0	11 ± 3	43 ± 7	11 ± 2	63 ± 3	5 ± 1	55.5 ± 3.58
3D Diffusion Policy	0 ± 0	56 ± 5	0 ± 0	47 ± 2	91 ± 4	67 ± 0	100 ± 0	74 ± 4	76.1 ± 2.32
ManiCM	0 ± 0	55 ± 2	0 ± 0	48 ± 3	87 ± 3	63 ± 2	100 ± 0	37 ± 16	69.0 ± 4.60
SDM Policy	0 ± 0	57 ± 0	100 ± 0	51 ± 4	86 ± 10	68 ± 10	0 ± 0	53 ± 12	74.8 ± 4.51
FlowPolicy*	57 ± 4	59 ± 5	-	47 ± 6	74 ± 5	66 ± 7	100 ± 0	92 ± 4	77.2 ± 2.84
Ours	63 ± 5	60 ± 5	-	51 ± 3	82 ± 6	74 ± 6	100 ± 0	90 ± 4	78.5 ± 2.61

1.0e-4, training for 3000 epochs. Evaluation is conducted every 200 epochs, and the best-performing checkpoint is saved. All experiments are conducted on a single NVIDIA A100 GPU.

Result Details. We report the detailed success rates of FreqPolicy on 53 tasks from the Adroit and MetaWorld benchmark in Table 5. The success rate of each task is averaged over experiments conducted with 3 random seeds.

A.4 More Details about VLA Settings

Implementation. For Diffusion Policy, we leverage the implementation of ‘OpenVLA (fine-tuned) + PD&AC, Cont-Diffusion’ from OpenVLA-OFT (OpenVLA-DP in this paper). For the Flow Matching Policy, we integrate it into the OpenVLA pipeline (i.e., OpenVLA-FlowMatching) based on the original implementation [40]. We train for 150K gradient steps for all models using a batch size of 16 across 4 A100 GPUs, policies receive one third-person image, one wrist camera image, robot proprioceptive state, and language instruction as input. The detailed hyperparameters are shown in Table 6. Please note that our training settings are not consistent with the original paper. In the default settings of OpenVLA-OFT [32], they train for 150K steps, using a batch size of 64 across 8 GPUs. However, in our experiments, considering the computational overhead and resource capabilities, we did not follow the full training settings consistent with the original paper (for example, the results we show are all trained on 2.4M samples, while the original paper trained 9.6M samples or even more), but to show the compatibility and superiority with VLA in comparison with the multi-step policy baselines. In the test, we perform 50-steps inference for OpenVLA-DP, and 1-step and 10-steps inference for OpenVLA-FlowMatching. The inference speed is averaged over the first three episodes of LIBERO-spatial to estimate performance. We follow the default settings in OpenVLA-OFT⁴ for other configurations.

Table 6: **The hyperparameter settings for VLA experiments.** These values are kept consistent across all methods.

Parameter	Values
use_film	True
use_proprio	True
num_images_in_input	2
lora_rank	32
image_aug	True
num_steps_before_decay	100,000
max_steps	150,000
num_gpus	4
batch_size	4
learning_rate	5e-4
num_trials_per_task	50
num_inference_steps	1 or 10 or 50

Compare with OpenVLA-OFT. OpenVLA-OFT is an improved version of OpenVLA [33] that integrates parallel decoding, action chunking, continuous action representation, and an L1 regression-based learning objective. It achieves state-of-the-art performance on the LIBERO simulation benchmark, significantly improving the average success rate while increasing action generation throughput by 26 times. We reproduced OpenVLA-OFT [32] under the same training and testing settings, and the results are shown in Table 7. Specifically, our method leads or is on par with OpenVLA-OFT in all categories except LIBERO-10, and leads OpenVLA-OFT in average success (94.8% vs. 94.2%). Moreover, even though FreqPolicy leverages an additional policy head, the inference speed remains competitive. (6.05Hz vs. 6.15Hz).

⁴<https://github.com/moojink/openvla-oft>

Table 7: Comparison with OpenVLA-OFT on LIBERO simulation benchmark.

Method	NFE	Spatial (%)	Object (%)	Goal (%)	Long (%)	Average (%)	Speed (Hz)
OpenVLA-DP	50	92.0	75.0	93.4	11.8	68.1	0.32
OpenVLA-FlowMatching*	1	95.0	97.6	96.0	85.2	93.5	5.92
OpenVLA-FlowMatching*	10	96.0	97.2	97.8	83.6	93.7	1.26
OpenVLA-OFT	-	96.6	98.6	91.0	90.6	94.2	6.15
OpenVLA-FreqPolicy	1	97.0	98.6	96.0	87.6	94.8	6.05

A.5 More Details about Real-World Tasks

Implementation. The real-world experiments are based on the open-source LeRobot framework⁵. LeRobot is designed to support real-world robotics research by providing models, datasets, and tools in PyTorch. It includes state-of-the-art methods, such as Diffusion Policy, which we use as a baseline, that have demonstrated strong transfer capabilities to real-world settings, with a focus on imitation learning. We convert our collected real-world data into the LeRobot-supported format and integrate both the Flow Matching Policy and our proposed FreqPolicy into the framework. This enables training directly on real data, facilitating practical deployment and inference. Training was conducted on a single NVIDIA A100 GPU. To ensure a fair comparison, all policy models were trained with aligned hyperparameters provided in Table 8. We also provide additional demonstration images for the three tasks in Fig. 5, as well as demonstration videos in the supplementary materials to facilitate better understanding.

Table 8: The hyperparameter settings for real-world experiments. These values are kept consistent across all methods.

Parameter	Values
input_images_Franka	[camera_front, camera_wrist]
input_images_UR	[camera_left, camera_wrist]
use_robot_state	True
image_size	480 × 480
vision_backbone	ResNet-18
n_obs_steps	2
horizon	48
n_action_steps	48
num_inference_steps	1 or 10
batch_size	64
# of training steps	100,000
optimizer	Adam
learning_rate	1e-4
weight_decay	1e-6
grad_clip_norm	10.0

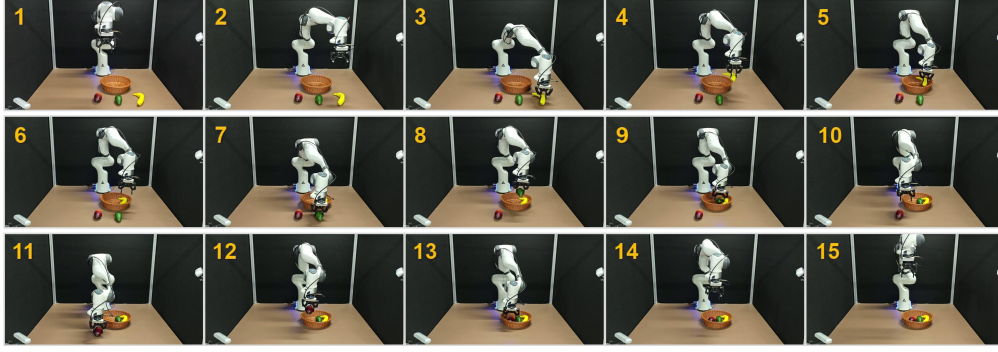
Evaluation Criteria. The initialization of the task environment plays a critical role, as it directly affects whether the robot can complete a task under the guidance of the policy model. Moreover, the conditions leading to task failure vary across different tasks. To facilitate a deeper understanding of the real-world experiments presented in this work, we provide target and object placement rules for the three tasks, along with detailed descriptions of the conditions under which task execution fails:

1) Fruit Sorting.

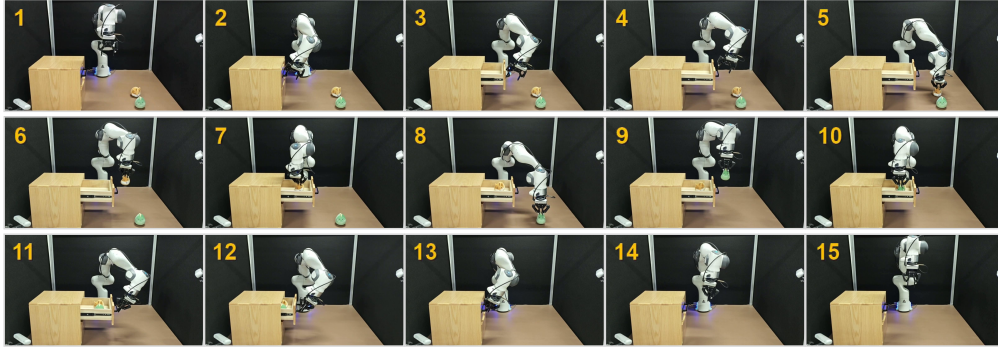
Target: Pick up bananas, avocados, and mangoes and put them into the basket in order.

Object placement rules: During execution, the fruit basket remains stationary in a fixed position. The spatial order of the three fruits—banana, avocado, and mango, from right to left—remains unchanged. However, in each trial, the exact placement of the fruits is randomly initialized within the front half of the basket.

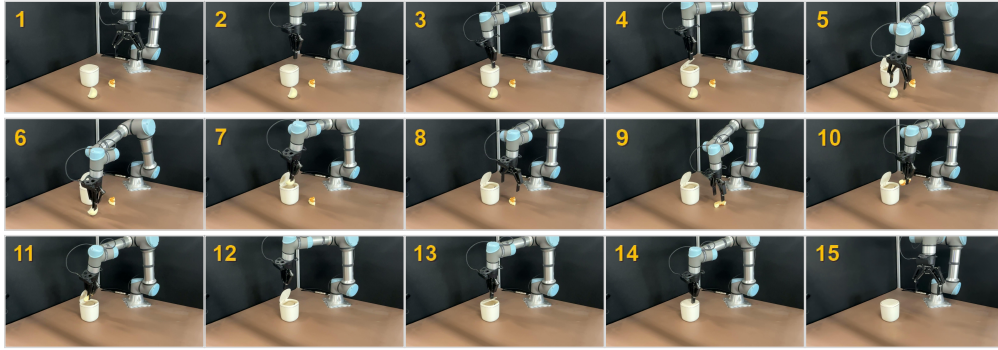
⁵<https://github.com/huggingface/lerobot>



(a) Demonstrations of real-world **Task_1**. This task is carried out on the Franka platform. The robot arm is tasked with sequentially picking up three different fruits—banana, avocado, and mango—and placing them into a basket. The fruits are positioned one at a time in random locations in front of the basket, and the task is deemed successful only if all three fruits are correctly picked up and placed in the basket.



(b) Demonstrations of real-world **Task_2**. This task is carried out on the Franka platform. The robot arm must first rotate to the appropriate angle, open its gripper, and insert it into the drawer handle to pull the drawer open. It then sequentially picks up two plush toys placed randomly on the table and places them inside the drawer. Finally, it closes the gripper and pushes the handle to shut the drawer. This requires the robot to accurately perceive the drawer handle’s start and end positions and be robust to the random positions of the toys.



(c) Demonstrations of real-world **Task_3**. This task is carried out on the UR platform. The robot arm first closes its gripper and attempts to tap a specific spot on the trash bin lid with the tip of the gripper to trigger it to pop open. Then, with the gripper open, the arm sequentially picks up two pieces of soft food waste—a half-steamed bun and a piece of bread—randomly placed on the table. Finally, the gripper closes again, moves behind the lid, and pushes it while tapping once more to fully close the bin. This process requires precise tapping actions and robustness to the random placement of the food waste.

Figure 5: **Demonstrations of three long-horizon real-world tasks.** Following a consistent protocol, 300 episodes are collected for each task for training, after which the trained policies are evaluated. The numbers in each image indicate the sequential steps in the task execution process.

Cases of task execution failed:

- If any fruit falls while picking it up, it will be considered a failure case.
- If the same fruit fails to be picked up more than 3 times in total, it will be considered a failure case.
- If all fruits are not completely put into the basket, it will be considered a failure case.

2) Toy Organization.

Target: Open the drawer, then put the two dolls into the drawer one by one, and close the drawer.

Object placement rules: During each execution, the cabinet remains in a fixed position, while two dolls are reinitialized with their relative spatial positions preserved.

Cases of task execution failed:

- If the drawer is not successfully opened, it is considered a failure case.
- If the drawer is not closed after all the dolls are placed, it is considered a failure case.
- If all the dolls are not placed in the drawer, it is considered a failure case.
- If the cumulative number of failures to pick up the same doll is 3 or more, it is considered a failure case.

3) Trash Disposal.

Target: Tap the trash can lid to make it pop open, put the food waste (steamed buns and bread) into the trash can one by one, and tap the lid again to close it.

Object placement rules: During the execution, the position of the trash can is fixed, the relative spatial positions of the two types of food waste remain unchanged, and the trash positions are reinitialized for each execution.

Cases of task execution failed:

- If the lid is not closed after picking up the food waste, it is considered a failure case.
- If the number of failed attempts to pick up the same food waste exceeds 3 times, the machine will be deemed as a failed case.
- If the lid is closed before all the food wastes are put into the trash can, it will be considered a failure case.
- If the lid is not opened and food waste is taken, it will be deemed as a failure case.