DISCOVERING PARAMETERIZED MOTOR SKILLS FOR VISION-LANGUAGE-ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision-Language-Action (VLA) models have shown strong potential for general-purpose robotic manipulation, yet their learned behaviors often remain tied to specific task contexts. A key observation is that semantically different manipulation tasks can share recurring short-horizon motor behaviors. Inspired by unsupervised skill discovery in reinforcement learning (RL), we formulate reusable behavior in VLA as discovering, parameterizing, and planning reusable motor skills. We introduce PLUGVLA, a framework that discovers recurring local motion patterns from multi-task demonstrations and converts them into detachable parameterized skill modules around a frozen task-adapted VLA. Specifically, a factorized vector-quantized variational autoencoder identifies recurring motion patterns, and each discovered skill is encoded as a low-rank weight update around the frozen supervised fine-tuning (SFT) VLA policy. This parameterization provides compact and detachable skill modules that can be selectively activated while preserving the frozen base policy and its native action interface. During execution, a vision-language router performs skill planning from the current observation and language instruction by choosing either a parameterized skill or the base policy. Experiments on LIBERO and RoboTwin 2.0 show consistent improvements over SFT references, while cross-task skill reuse and real-world evaluation demonstrate that learned skills can transfer across task contexts. Our results point toward a VLA paradigm in which increasingly capable high-level reasoning can leverage reusable motor skills, rather than relearning low-level behaviors for each new task. Code is available at `https://anonymous.4open.science/r/0D2F`.

1 INTRODUCTION

Vision-Language-Action (VLA) models have shown potential for robotic manipulation by mapping visual observations and language instructions directly to continuous actions (Kim et al., 2025b; Black et al., 2024; 2025). However, their performance can degrade under distribution shifts, in part because behaviors acquired during training do not always transfer effectively across task contexts (Fang et al., 2025; Hu et al., 2026). A closely related problem has been studied for decades in reinforcement learning (RL): rather than learning every task from primitive actions, skill discovery seeks reusable behavioral abstractions that can be invoked across different situations. From temporally extended options (Sutton et al., 1999), to unsupervised behavioral discovery (Eysenbach et al., 2019; Sharma et al., 2020), and later demonstration-driven skill learning for robotic manipulation (Pertsch et al., 2020; Ajay et al., 2021; Zhu et al., 2022), this line of research has shown that recurring behavioral structure can provide a useful abstraction for solving diverse and long-horizon tasks. VLA models face a conceptually similar opportunity: multi-task robot demonstrations often contain recurring local motions even when their language instructions and task semantics differ. This naturally raises the question of whether such recurring behaviors can serve as reusable skills within VLA models.

This question becomes particularly relevant as the high-level reasoning capabilities of foundation models continue to advance. Recent frontier models such as GPT-6 Astra demonstrate increasingly strong reasoning and multi-step task execution capabilities (OpenAI, 2026). As high-level reasoning becomes more capable, a reusable repertoire of low-level motor skills offers a natural interface between abstract task planning and continuous robotic control. Recent VLA research has already begun to introduce mechanisms that resemble such behavioral abstractions, although they are typically developed from different perspectives (Hu et al., 2026; Yang et al., 2026).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

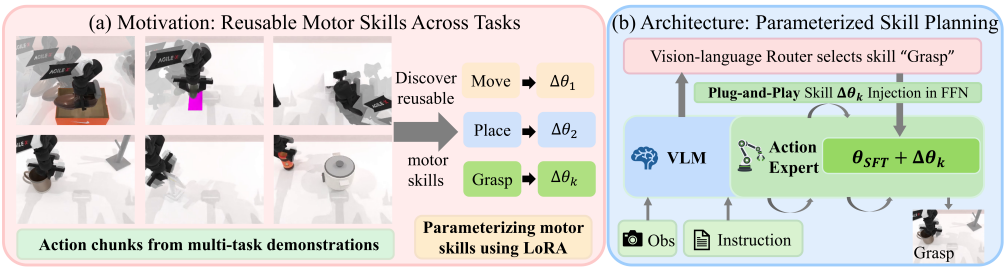

Figure 1: Motivation and overview of PLUGVLA. Recurring local motor patterns are discovered from multi-task demonstrations, represented as plug-and-play parameterized skills, and organized into a short-horizon plan by a vision-language router.

From the perspective of skill representation, these approaches can be grouped into three categories. *Pre-specified* approaches provide behavioral structure through language instructions, motion primitives, or retrieved memories (Jeong et al., 2026; Li et al., 2025). *Embedding-based* approaches encode behavioral variation through latent skill variables or interventions on internal representations (Hu et al., 2026; Khayatan et al., 2026). *Parameter-based* approaches represent behavioral specialization in model parameters, for example through low-rank adaptation or parameter merging (Luo et al., 2026; Fu et al., 2026). Despite these differences, a gap remains. Pre-specified approaches depend on externally defined behavioral structure, while embedding-based skills remain coupled to internal representations and are not exposed as detachable policy modules. Parameter-based approaches offer a natural interface for modular reuse, but existing methods mainly organize parameter specialization around task-specific adaptations or expert conditions, rather than discovering cross-task reusable motor skills and representing them as reusable parameterized skills. We therefore ask: *How can recurring motor skills be discovered from multi-task demonstrations, represented as reusable modules, and planned according to the current task context?*

We address this question with PLUGVLA, which organizes reusable motor behavior in VLA models through three stages: **skill discovery**, **skill representation**, and **skill planning**. First, we treat recurring short-horizon motion patterns in multi-task demonstrations as motor skills and discover them in a motion-centric representation space using a factorized vector-quantized variational autoencoder. Second, rather than introducing an additional action representation or modifying the shared policy, we encode each discovered skill as an independent low-rank parameter increment around a frozen task-adapted VLA obtained through SFT. Each increment therefore acts as a compact, detachable specialization that can be attached to the shared policy when needed, providing a plug-and-play interface for reusing discovered motor behaviors while preserving the model's shared knowledge and native action space. Third, during execution, a vision-language router performs context-dependent skill planning by selecting either one parameterized skill or the unchanged base policy from the current observation and language instruction. This design separates *what* reusable motor behaviors are available, *how* they are stored in the policy, and *when* they should be activated.

We evaluate PLUGVLA on LIBERO (Liu et al., 2023) and RoboTwin 2.0 (Chen et al., 2026a). Across benchmarks, parameterized skill discovery and planning improve manipulation performance over the SFT references. More importantly, task-held-out experiments show that parameterized skills can transfer beyond the task contexts in which their routing supervision is obtained: a skill learned in one context can be reused in another by adapting the context-dependent planner while keeping the skill parameters fixed. These results provide evidence that recurring local motions can serve as modular behavioral units within VLA models rather than remaining entangled with task policies.

Our contributions are:

- We formulate reusable local motor behavior in VLA models from the perspective of unsupervised skill discovery in RL, and organize the problem into three stages: **skill discovery**, **skill representation**, and **skill planning**.
- We introduce PLUGVLA, which discovers motion patterns from multi-task demonstrations and encodes each discovered skill as a detachable low-rank parameter specialization around a frozen

task-adapted VLA. A vision-language planner then selects these skills according to task context while retaining the unchanged base policy as a fallback.

- Experiments on LIBERO and RoboTwin 2.0 show improved manipulation performance over SFT references, while task-held-out simulation and real-robot evaluations further provide evidence that learned parameterized skills can be reused across task contexts by adapting the planner without modifying the skill parameters.

2 Related Work

2.1 Skill Discovery and Hierarchical Control

Temporal abstraction has been used to organize reusable behavior in RL. Options formalize temporally extended actions (Sutton et al., 1999), while DIAYN (Eysenbach et al., 2019) and DADS (Sharma et al., 2020) discover diverse behaviors without task-specific rewards. For robot learning, SPiRL (Pertsch et al., 2020) and OPAL (Ajay et al., 2021) learn reusable latent skills from offline experience, while BUDS (Zhu et al., 2022) discovers recurring skills from unsegmented multi-task demonstrations for long-horizon manipulation. Related imitation-learning approaches use latent plans (Lynch et al., 2020) or discrete action abstractions (Lee et al., 2024; Mete et al., 2024) to capture temporal structure in demonstrations. These works motivate our view that multi-task demonstrations contain reusable behavioral structure. Our setting differs in that the discovered unit is not introduced as a separate latent-conditioned policy or action decoder; instead, we expose each recurring local motion as a detachable parameter specialization of a task-adapted VLA.

2.2 Reusable Behavior and Parameter-Space Adaptation in VLA

VLA methods introduce reusable behavioral structure through several interfaces. Prompt- and context-based approaches use language feedback or motion primitives (Jeong et al., 2026; Chen et al., 2026b), and demonstrations or retrieved memories (Li et al., 2025) to guide a policy, while representation-based approaches encode behavioral variation through latent variables (Hu et al., 2026) or internal activation interventions (Häon et al., 2025; Khayatan et al., 2026). A complementary direction specializes model parameters through routed experts (Luo et al., 2026; Yang et al., 2026), low-rank updates (Guo et al., 2026; Jiang et al., 2026), or model merging (Fu et al., 2026). More broadly, parameter-efficient modules can serve as composable carriers of capabilities (Hu et al., 2022; Pfeiffer et al., 2021; Liu et al., 2025b), and Transformer analyses provide a theoretical connection between contextual effects and low-rank parameter updates (Goldwaser et al., 2026). This observation further motivates reusable behaviors: we encode reusable skill guidance directly in parameter space as detachable low-rank updates to a shared VLA.

3 Preliminaries

Skill learning in reinforcement learning (RL) studies reusable behavioral abstractions that can be invoked across different task contexts (Sutton et al., 1999; Eysenbach et al., 2019). Let $z \in \mathcal{Z}$ denote a skill variable and A_t^h an action chunk conditioned on observation o_t and language instruction ℓ:

$$A_t^h \sim \pi\left(A_t^h \mid o_t, \ell, z\right). \tag{1}$$

Under this abstraction, reusable skill learning can be viewed through three components.

Skill discovery determines which recurring behavioral patterns constitute the skill set $\mathcal{Z}$. Prior work discovers such skills from interaction or demonstrations through diversity, dynamics, or reconstruction based objectives (Sharma et al., 2020; Pertsch et al., 2020; Ajay et al., 2021; Zhu et al., 2022).

Skill representation determines how the skill variable z conditions the policy. Existing approaches represent skills through predefined behavioral primitives, learned latent representations, or parameter-space specializations (Jeong et al., 2026; Hu et al., 2026; Luo et al., 2026).

Skill planning determines which reusable skills should be invoked under the current task context. For a planning horizon of P skill decisions, a planner may produce

$$\mathbf{z}_t = (z_{t,0}, \ldots, z_{t,P-1}) \sim p_\psi\left(\mathbf{z}_t \mid o_t, \ell\right), \tag{2}$$

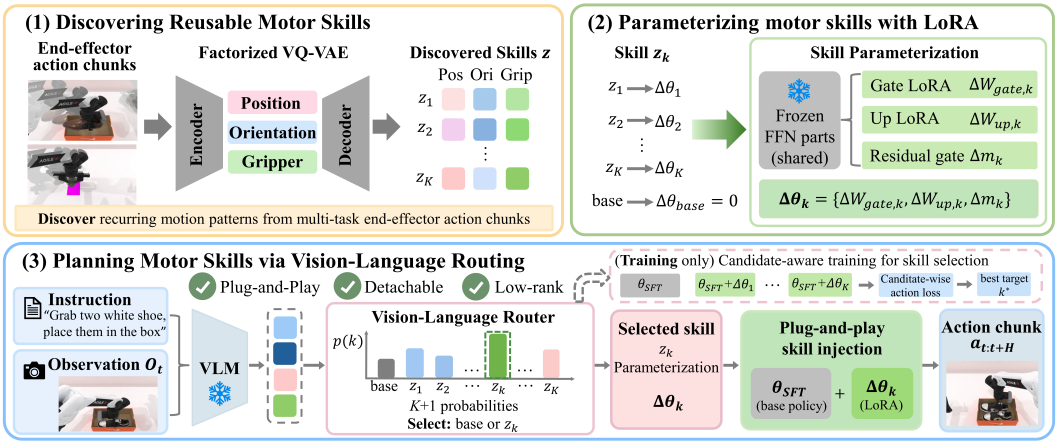

Figure 2: Overview of PLUGVLA. Recurring end-effector motion patterns are first discovered from multi-task demonstrations and represented as lightweight parameter updates around a frozen SFT VLA. A vision-language planner then predicts a short-horizon sequence over the parameterized skills and the unchanged base policy. Only the first planned choice is executed before replanning from a new observation, enabling closed-loop and plug-and-play skill reuse across task contexts.

where each $z_{t,j} \in \mathcal{Z}$ denotes a future-stage skill. These skills provide high-level behavioral abstractions, while the conditioned policy generates continuous robot actions.

4 METHODS

4.1 PROBLEM FORMULATION

Let $\mathcal{D} = \{\tau_i\}_{i=1}^N$ denote multi-task robot demonstrations, where each trajectory contains observations o_t, language instructions ℓ, and action chunks $A_t^h = (a_t, \ldots, a_{t+h-1})$. The observation o_t includes visual observations and robot proprioception. We obtain a task-adapted reference policy $\pi_{\theta_{\mathrm{SFT}}}$ through supervised fine-tuning and keep it frozen thereafter. Our goal is to discover recurring local motor patterns and represent them as reusable parameter specializations of this shared VLA. Suppose K motor skills are discovered. Each skill k is associated with a parameter update $\Delta\theta_k$. We define the available skill set as $\mathcal{K} = \{\mathrm{base}, 0, \ldots, K-1\}$, where $\Delta\theta_{\mathrm{base}} = 0$ denotes the unchanged SFT policy. Given the current observation and instruction, the router predicts skills for the next P action chunks, with the first skill conditioning the current action generation:

$$\hat{\mathbf{k}}_t = (\hat{k}_{t,0}, \ldots, \hat{k}_{t,P-1}) = g_\psi(o_t, \ell), \quad A_t^{H_{\mathrm{act}}} \sim \pi_{\theta_{\mathrm{SFT}}+\Delta\theta_{\hat{k}_{t,0}}}\left(A_t^{H_{\mathrm{act}}} \mid o_t, \ell\right), \quad \hat{k}_{t,j} \in \mathcal{K}. \tag{3}$$

Accordingly, the problem separates into three stages: discovering recurring local motor patterns, representing them as reusable parameter updates, and planning activation over future action chunks.

4.2 DISCOVERING REUSABLE MOTOR SKILLS

A reusable motor skill should capture a local motion pattern that can recur across semantically different tasks rather than encode a complete task identity. Motivated by prior work showing that reusable temporal structure can be recovered from unsegmented robot trajectories (Lynch et al., 2020; Zhu et al., 2022; Lee et al., 2024; Mete et al., 2024), we perform skill discovery in a motion-centric space separated from the VLA's native action representation.

For each demonstration trajectory, we extract fixed-length local motion segments using a sliding window over the end-effector position, orientation, and gripper-state sequences. Each segment captures the local motor behavior executed over a short temporal horizon.

We use a vector-quantized variational autoencoder (VQ-VAE) (van den Oord et al., 2017) to discretize these motion segments. To preserve the structure of different motion components, we introduce a

factorized quantization scheme with codebooks for position, orientation, and gripper motion. Each component is encoded into a latent sequence and quantized independently. For $c \in \{\mathrm{pos}, \mathrm{ori}, \mathrm{grip}\}$,

$$q_{t,n}^{c} = \arg\min_{q} \left\| \bar{h}_{t,n}^{c} - e_{q}^{c} \right\|_{2}^{2}, \qquad n = 0, \ldots, L_{\mathrm{vq}} - 1, \tag{4}$$

where $\bar{h}_{t,n}^{c}$ denotes the normalized latent representation and e_{q}^{c} a codebook entry for component c. The ordered codes from the three components jointly define a local motion pattern:

$$P_t = \left(q_{t,n}^{\mathrm{pos}}, q_{t,n}^{\mathrm{ori}}, q_{t,n}^{\mathrm{grip}} \right)_{n=0}^{L_{\mathrm{vq}}-1}, \quad \mathcal{P}_{\mathrm{sel}} = \{P^{(k)}\}_{k=0}^{K-1}, \quad z_t = \begin{cases} k, & P_t = P^{(k)} \in \mathcal{P}_{\mathrm{sel}}, \\ \mathrm{base}, & \text{otherwise.} \end{cases} \tag{5}$$

Here $\mathcal{P}_{\mathrm{sel}}$ contains the K most frequently recurring patterns in the demonstrations, and each selected pattern defines a reusable motor skill. Thus, $z_t \in \mathcal{K}$ is determined by recurring local motion structure rather than semantic task identity or manually specified subtasks. Appendix B for more details.

4.3 Parameterizing Motor Skills

We next convert each discovered motor skill into a detachable parameter specialization of the shared VLA. Analyses of Transformer blocks show that contextual changes can, under conditions, be expressed through parameter-space updates (Goldwaser et al., 2026). Motivated by this perspective and parameter-efficient adaptation (Hu et al., 2022), we realize each discovered skill $z = k$ as a parameter update $\Delta\theta_k$ around the frozen SFT policy. For each Action Expert layer l, skill k provides low-rank updates to the gate and up projections of the gated FFN. For $p \in \{\mathrm{gate}, \mathrm{up}\}$,

$$W_{p,k}^{(l)} = W_{p,\mathrm{SFT}}^{(l)} + \Delta W_{p,k}^{(l)}, \qquad \Delta W_{p,k}^{(l)} = \frac{\alpha}{r} B_{p,k}^{(l)} A_{p,k}^{(l)}. \tag{6}$$

Here r denotes the LoRA rank and α is the scaling factor. Each skill additionally introduces a residual-gate increment $\Delta m_{k}^{(l)}$. The complete parameterization of skill k is

$$\Delta\theta_k = \left\{ \Delta W_{\mathrm{gate},k}^{(l)}, \Delta W_{\mathrm{up},k}^{(l)}, \Delta m_{k}^{(l)} \right\}_{l=1}^{L}. \tag{7}$$

The down projections, attention modules, vision-language backbone, and action output projection remain shared and frozen. Skill specialization is restricted to gated feature formation and residual modulation, while the shared task-adapted parameters remain unchanged. This yields a plug-and-play parameter interface: skills can be selectively attached to or bypassed from the same frozen SFT policy. The gated-FFN computation and design rationale are provided in Appendix C.

For each discovered skill k, we collect the samples satisfying $z_t = k$ into $\mathcal{D}_k$, and let $\mathcal{D}_+ = \bigcup_{k=0}^{K-1} \mathcal{D}_k$ denote the union of all skill-associated samples. We retain the native flow-matching objective (Lipman et al., 2023) and optimize the skill library by

$$\mathcal{L}_{\mathrm{skill}} = \mathbb{E}_{(o_t, \ell, A_t, z_t) \sim \mathcal{D}_+} \left[\mathcal{L}_{\mathrm{FM}} \left(\theta_{\mathrm{SFT}} + \Delta\theta_{z_t}; o_t, \ell, A_t \right) \right]. \tag{8}$$

Only the update associated with the discovered label z_t is optimized for each sample; the SFT reference parameters remain frozen throughout. After fitting, the resulting skill library $\{\Delta\theta_k\}_{k=0}^{K-1}$ is fixed before skill planning. The full flow-matching formulation is provided in Appendix C.2.

4.4 Planning Motor Skills with a Vision-Language Router

The discovery label z_t identifies which recurring motion pattern a demonstration segment belongs to, but it does not indicate which parameterized skill should be activated under the current task context. We therefore learn a vision-language router that performs receding-horizon planning over the frozen skill library using the multimodal context as the VLA (ichter et al., 2023; Yang et al., 2026).

The router operates on frozen vision-language prefix features:

$$H_t^{\mathrm{prefix}} = f_{\theta_{\mathrm{SFT}}}^{\mathrm{prefix}}(o_t, \ell), \qquad p_{\psi,j} = \mathrm{softmax}\left(R_{\psi,j}(H_t^{\mathrm{prefix}}) \right), \quad j = 0, \ldots, P-1. \tag{9}$$

Here $p_{\psi,j}$ is a categorical distribution over $\mathcal{K}$ for planning anchor $\tau_{t,j} = t + jH_{\mathrm{act}}$. The first position determines the skill applied to the current policy call, while subsequent positions provide look-ahead predictions over future action chunks. Architectural details are provided in Appendix D.

A distinction is that discovering a motion pattern and deciding when its parameterization should be activated are different problems. Rather than training the router to reproduce the discovery label z_t, we evaluate the action fitting induced by candidate parameterizations. For each planning anchor τ,

$$C_{\tau,k} = \mathcal{L}_{\mathrm{FM}}\left(\theta_{\mathrm{SFT}} + \Delta\theta_k; o_\tau, \ell_\tau, A_\tau^{H_{\mathrm{train}}}\right), \qquad k_\tau^\star = \arg\min_{k\in\mathcal{K}} C_{\tau,k}. \tag{10}$$

The candidate-wise optimum $k_\tau^\star$ serves as the primary planning target, while the discovery label z_τ provides a weaker auxiliary constraint that preserves the connection to the discovered motion structure. The two labels need not coincide: z_τ describes the recurring motion pattern observed in the demonstration, whereas $k_\tau^\star$ identifies the frozen parameterization that best fits the desired action under the current context. The router is trained jointly over all valid planning anchors:

$$\mathcal{L}_{\mathrm{plan}}(\psi) = \mathbb{E}_t \sum_{j\in\mathcal{J}_t} w_j \left[\mathrm{CE}\left(p_{\psi,j}, k^\star_{\tau_{t,j}}\right) + \lambda_{\mathrm{src}}\,\mathrm{CE}\left(p_{\psi,j}, z_{\tau_{t,j}}\right)\right]. \tag{11}$$

Here, λ_{src} is a hyperparameter controlling the weight of the source-label supervision term. All candidate losses are precomputed with both the SFT reference and skill library frozen; only the planner parameters ψ are optimized. When $\hat{k}_{t,0} = \mathrm{base}$, we set $\Delta\theta_{\mathrm{base}} = 0$, bypassing skill-specific updates and reducing the computation to the frozen SFT reference policy. This formulation separates *what* reusable motor behaviors are available from *when* their activation should occur.

5 Experiments

We organize the evaluation around three questions. **Q1** asks whether parameterized motor skills improve manipulation performance and whether a learned skill library can be reused across task contexts. **Q2** examines what the frequently reused skills represent in terms of local robot motion and how their invocation affects local execution. **Q3** isolates the contributions of context-conditioned skill planning and the factorized motion representation.

Benchmarks. We evaluate PlugVLA in both simulated and real-world settings. For simulated manipulation, we use RoboTwin 2.0 (Chen et al., 2026a) and LIBERO (Liu et al., 2023). LIBERO contains four standard task suites, while RoboTwin 2.0 is evaluated on two task sets with different difficulty levels. We further validate PlugVLA on a RealMan dual-arm robot in a real-world manipulation environment. Task details and evaluation settings are provided in Appendix E.

Baselines. We compare PlugVLA with representative imitation-learning and VLA baselines, including Diffusion Policy (Chi et al., 2025), ACT (Zhao et al., 2023), RDT (Liu et al., 2025a), DP3 (Ze et al., 2024), OpenVLA (Kim et al., 2025b), GR00T-N1 (Bjorck et al., 2025), OpenVLA-OFT (Kim et al., 2025a), MemoryVLA (Shi et al., 2026), UniVLA (Wang et al., 2026), InternVLA-M1 (Chen et al., 2025), π_0-FAST (Pertsch et al., 2025), π_0 (Black et al., 2024), $\pi_{0.5}$ (Black et al., 2025), BehaviorVLA (Hu et al., 2026), and KinRT (Yang et al., 2026). Detailed implementation and training configurations are provided in Appendix A.

5.1 Performance and Cross-Task Skill Reuse

Q1 evaluates whether parameterized motor skills improve the task-adapted VLA and whether the learned skill library can be reused across task contexts. We evaluate PlugVLA on RoboTwin 2.0 and LIBERO, and further test cross-task skill library reuse between the two RoboTwin task groups.

Results on RoboTwin 2.0. For RoboTwin 2.0, we divide the evaluated tasks into two groups according to task difficulty: a 9-task set and a more challenging 8-task set, reported in Tables 1 and 2, respectively. PlugVLA outperforms the existing representative baselines, including BehaviorVLA and KinRT, on both task groups. We attribute the improvement to the complementary roles of the shared task-adapted policy and reusable local parameterizations: the SFT reference preserves general task knowledge, while the skill-specific updates specialize local motion regimes and the VLR selects them according to the current observation and instruction. This allows the model to refine local execution without replacing the shared policy with separate task-specific models.

Table 1: Results on the RoboTwin 2.0 benchmark on the 9-task set. Success rates are reported under the **Hard** setting with 100 rollouts per task. Best results are **bold**. *Single* denotes per-task training; † denotes results reproduced by us. Skill-reuse denotes cross-task transfer from the 8-task set.

Method	Adjust Bottle	Click Alarm	Dump Bigbin	Grab Roller	Move Card	Place A2B Right	Bread Basket	Burger Fries	Container Plate	Avg.
RDT (Single)	75	12	32	43	11	1	2	27	17	24.4
ACT (Single)	23	4	1	25	0	0	0	0	1	6.0
DP (Single)	0	5	0	0	0	0	0	0	0	0.6
DP3 (Single)	3	14	53	2	3	0	1	18	1	10.6
π_0 (Single)	56	11	24	80	22	6	4	4	45	28.0
$\pi^{\dagger}_{0.5}$	77	45	77	91	63	34	50	80	63	64.4
BehaviorVLA	83	**52**	77	90	41	25	36	61	62	58.6
Skill-reuse	76	48	80	97	**74**	**38**	**60**	**87**	**77**	**70.8**
PLUGVLA	**85**	49	**82**	**97**	68	37	56	86	73	70.3

Table 2: Results on the challenging RoboTwin 2.0 8-task set. Success rates are reported under the **Easy** | **Hard** setting with 100 rollouts per task. Best results are **bold**. *Single* denotes per-task training; † denotes results reproduced by us. Skill-reuse denotes cross-task transfer from the 9-task set.

Method	Hand Block	Hang Mug	Move Can	Open Laptop	Place Shoes	Place Pad	Rotate Qrcode	Turn Switch	Avg.
RDT (Single)	45 | 14	23 | 16	25 | 12	59 | 32	42 | 13	1 | 0	50 | 5	35 | 15	35.0 | 13.4
ACT (Single)	42 | 0	7 | 0	22 | 4	56 | 0	9 | 0	0 | 0	90 | 21	52 | 25	34.8 | 6.3
π_0 (Single)	45 | 8	93 | 13	58 | 21	85 | 46	15 | 0	7 | 1	68 | 15	27 | 23	49.8 | 15.9
$\pi^{\dagger}_{0.5}$	45 | 17	20 | 10	63 | 10	96 | 65	58 | 40	32 | 25	86 | 22	58 | 24	57.3 | 26.6
KinRT	34 | 22	12 | 2	38 | **28**	84 | **83**	28 | 24	18 | 10	40 | **34**	34 | 28	36.0 | 28.9
Skill-reuse	46 | 17	22 | 11	65 | 10	95 | 67	61 | 42	32 | 25	**93** | 25	56 | 23	58.8 | 27.5
PLUGVLA	**52** | **26**	**25** | **16**	**68** | 14	**97** | 72	**66** | **42**	**36** | **31**	90 | 27	**61** | **35**	**61.9** | **32.9**

Results on LIBERO. As shown in Table 3, PLUGVLA achieves an average success rate of 98.0%, outperforming $\pi_{0.5}$ and OpenVLA-OFT, and matching the performance of the current state-of-the-art BehaviorVLA. The improvement is most pronounced on LIBERO-Long, where PLUGVLA improves over $\pi_{0.5}$ by 2.4 percentage points. We attribute this gain to the ability of the parameterized skills to improve local motion execution, while the VLR dynamically replans skill activation across multiple stages of long-horizon tasks.

Table 3: Success rates on LIBERO with 500 rollouts per suite. Best results are bolded.

Method	Spatial	Object	Goal	Long	Avg.
Diffusion Policy	78.5	87.5	73.5	64.8	76.1
GR00T-N1	94.4	97.6	93.0	90.6	93.9
OpenVLA-OFT	97.6	98.4	97.9	94.5	97.1
MemoryVLA	98.4	98.4	96.4	93.4	96.7
UniVLA	95.4	98.8	93.6	94.0	95.4
InternVLA-M1	98.0	99.0	93.8	92.6	95.9
π_0-FAST	96.4	96.8	88.6	60.2	85.5
π_0	96.8	98.8	95.8	85.2	94.2
$\pi_{0.5}$	98.8	98.2	98.0	92.4	96.9
BehaviorVLA	**99.2**	**99.4**	**98.8**	94.6	**98.0**
PLUGVLA	**99.2**	**99.4**	98.6	**94.8**	**98.0**

Cross-task skill reuse. The **Skill-reuse** results in Tables 1 and 2 evaluate cross-task skill reuse by exchanging skill libraries between the two RoboTwin task groups. For each transfer direction, the skill library learned from the source group is frozen, and only the VLR is adapted using demonstrations from the target group. Compared with $\pi_{0.5}$, Skill-reuse improves the target-task success rate by 6.4 and 0.9 percentage points in the two transfer directions, respectively. These results show that the learned skill library remains useful under changing task contexts without refitting the underlying skill parameters. Notably, transferring the skill library learned from the more challenging 8-task set yields the best performance on the 9-task set, suggesting that more challenging source tasks may expose reusable local motion patterns that transfer particularly well to new task contexts.

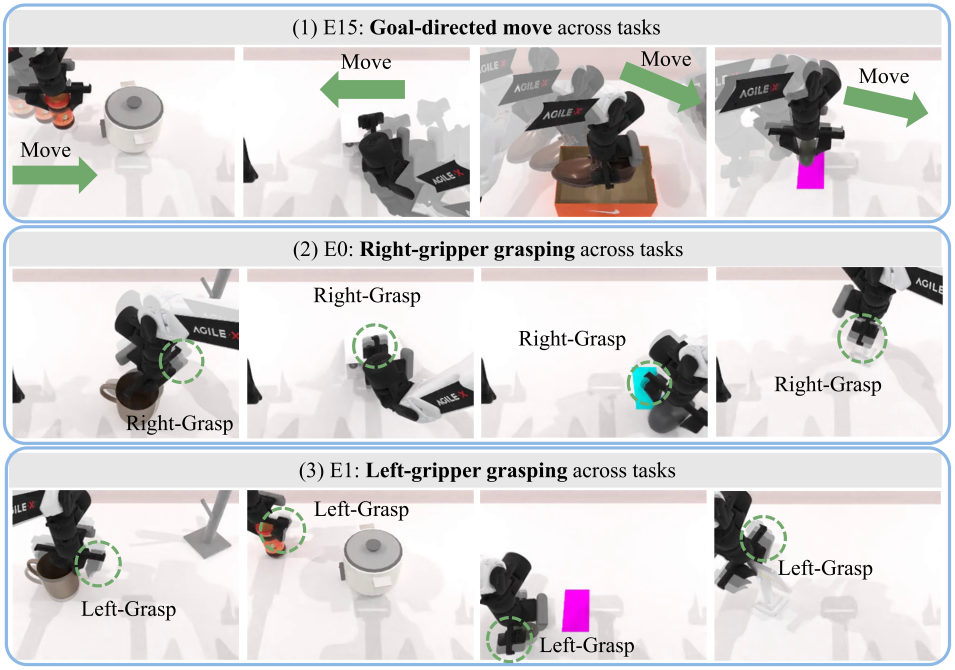

Figure 3: Cross-task consistency of discovered motor skills. Representative activations show that (1) E15 corresponds to goal-directed motion, (2) E0 to right-gripper grasping, and (3) E1 to left-gripper grasping across different RoboTwin tasks.

5.2 Analysis of Discovered Motor Skills

Q2 examines whether the discovered parameterizations acquire consistent local motor semantics across tasks, and whether activating them produces meaningful changes in action execution. We therefore analyze the learned skills from three complementary perspectives: cross-task consistency, local behavioral effects, and reuse under an unseen real-robot task composition.

Discovered skills exhibit consistent local motor semantics across tasks. Figure 3 shows that the same learned skills are recurrently selected under similar local motion contexts despite changes in object identity and task semantics. This consistency indicates that the resulting parameterizations are not simply tied to individual task labels, but instead capture recurring structure in short-horizon robot motion. The observation complements the task-level performance results in Q1 by showing that the learned skill library organizes behavior around reusable local motion patterns rather than memorized task identities. Additional activation statistics and cross-task patterns are provided in Appendix F.

Skill parameterizations induce targeted changes in local execution. Figure 4 examines how selected skills alter the behavior of the shared policy within contexts. The comparisons reveal that skill activation does not merely change the routing label: it changes the action execution in task-relevant ways. In particular, the differences occur during the local intervals associated with the selected skills, supporting the interpretation of $\Delta\theta_k$ as a behavior-specific specialization around the shared SFT policy. This provides a mechanism-level explanation for why dynamic skill selection can improve task success over the unchanged reference policy. See detailed sequences in Appendix F.1.

Real-robot illustration of cross-task skill reuse. We examine whether parameterized skills learned in one task context can be reused in another task context on a RealMan dual-arm robot. We train on two tasks, Task A and Task B, and retain their skill parameterizations when executing a held-out Task C that involves behaviors observed during training. No Task-C-specific skill parameters are introduced. As shown in Figure 5, skills acquired from the training tasks can be selected under the new task context and contribute to the execution of Task C. This example complements the cross-task activation analysis by showing that learned skill parameterizations can remain applicable beyond the task contexts in which they were obtained. Detailed task descriptions are provided in Appendix G.

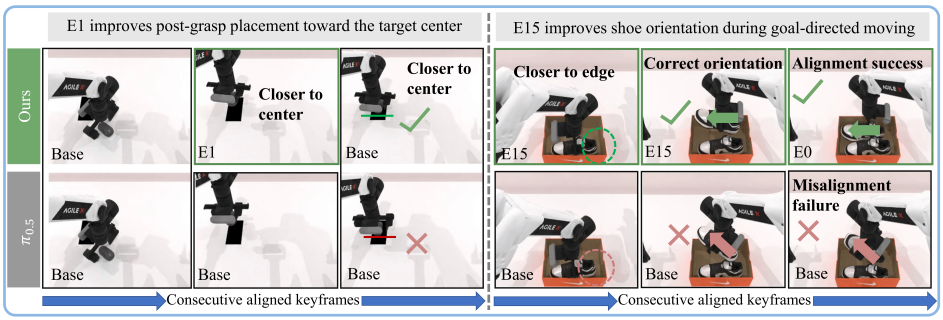

Figure 4: Effects of parameterized skills on local action execution, comparing PLUGVLA with the SFT reference $\pi_{0.5}$. Left: activating E1 during the post-grasp interval shifts object placement closer to the target center. Right: activating E15 during goal-directed moving produces a more appropriate shoe orientation before placement. Aligned keyframes highlight the local execution intervals associated with the analyzed skill activations.

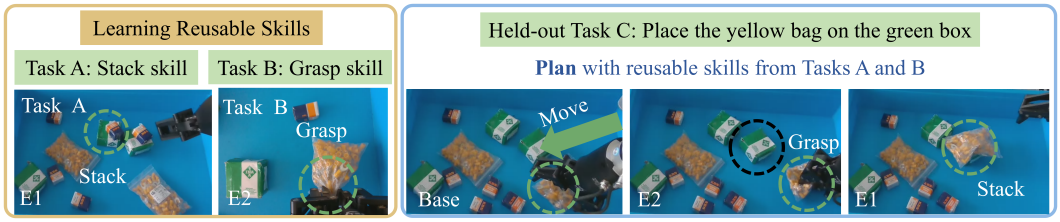

Figure 5: Real-robot validation of cross-task skill reuse under unseen task composition. Parameterized skills learned from Training Tasks A and B are retained and reused to execute the held-out Task C without learning Task-C-specific skill parameters. The planner selects reusable behaviors according to the evolving task context, illustrating that learned motor skills can be recomposed beyond the task contexts in which they were acquired.

5.3 Ablation Studies

Q3 examines the contributions of the VLR and the EEF-based representation for skill discovery. We compare the full model with two controlled variants: *Random Routing*, which replaces the VLR with a frequency-matched random selector while preserving the same skill library, and *Joint VQ*, which performs skill discovery using native joint-space trajectories instead of EEF trajectories. As shown in Table 4, random routing reduces success rates by 2.0, 7.2, and 7.1 percentage points, respectively, demonstrating the benefit of context-conditioned skill selection. Joint-space discovery leads to drops of 1.0, 4.6, and 3.5 points, indicating that EEF trajectories provide a more suitable representation for discovering reusable local motion patterns.

Table 4: Ablation of skill planning and motion representation. RT denotes RoboTwin.

Method	LIBERO	RT 2.0-8	RT 2.0-9
PLUGVLA	**98.0**	**47.4**	**70.3**
w/ Random	96.0	40.2	63.2
w/ Joint VQ	97.0	42.8	66.8

6 Conclusion

We introduced PLUGVLA, a framework that discovers recurring local motion patterns from multi-task demonstrations, parameterizes them as reusable low-rank skill updates, and plans their activation with a vision-language router around a frozen task-adapted VLA. Across LIBERO, RoboTwin 2.0, and real-robot task composition, the learned skills improve local action execution and can be reused across different task contexts without refitting the underlying skill parameters. The real-robot results further show that skills learned from separate training tasks can be recombined in an unseen task context. Cross-task transfer on RoboTwin also reveals an asymmetric reuse pattern, where skills learned from the more challenging task set transfer particularly well to the other task group, suggesting that skill transfer may depend on the diversity of local motions encountered during discovery.

AI USE STATEMENT

We use LLMs for polish writing. Specifically, LLMs assist in refining the grammar, clarity, and overall presentation of the paper, ensuring that the text is clear and professionally written. No experimental results or core content were generated by LLMs.

REFERENCES

Anurag Ajay, Aviral Kumar, Pulkit Agrawal, Sergey Levine, and Ofir Nachum. OPAL: offline primitive discovery for accelerating offline reinforcement learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL https://openreview.net/forum?id=V69LGwJ0lIN.

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.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. pi_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, brian ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *Proceedings of The 9th Conference on Robot Learning*, pp. 17–40, 2025.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, Weiliang Deng, Yubin Guo, Tian Nian, Xuanbing Xie, Qiangyu Chen, Kailun Su, Tianling Xu, Guodong Liu, Mengkang Hu, Huan ang Gao, Kaixuan Wang, Zhixuan Liang, Yusen Qin, Xiaokang Yang, Ping Luo, and Yao Mu. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. In *Forty-third International Conference on Machine Learning*, 2026a. URL https://openreview.net/forum?id=itonej9GIV.

William Chen, Jagdeep Singh Bhatia, Catherine Glossop, Nikhil Mathihalli, Ria Doshi, Andy Tang, Danny Driess, Karl Pertsch, and Sergey Levine. Steerable vision-language-action policies for embodied reasoning and hierarchical control. *arXiv preprint arXiv:2602.13193*, 2026b.

Xinyi Chen, Yilun Chen, Yanwei Fu, Ning Gao, Jiaya Jia, Weiyang Jin, Hao Li, Yao Mu, Jiangmiao Pang, Yu Qiao, et al. Internvla-m1: A spatially guided vision-language-action framework for generalist robot policy. *arXiv preprint arXiv:2510.13778*, 2025.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL https://openreview.net/forum?id=SJx63jRqFm.

Irving Fang, Juexiao Zhang, Shengbang Tong, and Chen Feng. From intention to execution: Probing the generalization boundaries of vision-language-action models. *arXiv preprint arXiv:2506.09930*, 2025.

Yuxia Fu, Zhizhen Zhang, Yuqi Zhang, Zijian Wang, Zi Huang, and Yadan Luo. Mergevla: Cross-skill model merging toward a generalist vision-language-action agent. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22335–22347, June 2026.

Adrian Goldwaser, Michael Munn, Javier Gonzalvo, and Benoit Dherin. Equivalence of context and parameter updates in modern transformer blocks. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=aZ5Di8Ii4X.

Yufei Guo, Yinan Wu, Haoran Duan, Guiguang Ding, and Jungong Han. Phaselora: Control-regime-conditioned low-rank adaptation for continuous-action vision-language-action policies. *arXiv preprint arXiv:2608.15285*, 2026.

Bear Häon, Kaylene Caswell Stocking, Ian Chuang, and Claire Tomlin. Mechanistic interpretability for steering vision-language-action models. In Joseph Lim, Shuran Song, and Hae-Won Park (eds.), *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pp. 2743–2762. PMLR, 27–30 Sep 2025.

Bing Hu, Zaijing Li, Rui Shao, Junda Chen, April Hua Liu, Wei-Shi Zheng, and Liqiang Nie. From abstraction to instantiation: Learning behavioral representation for vision-language-action model. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=2NRrsz4nUB.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

brian ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, Dmitry Kalashnikov, Sergey Levine, Yao Lu, Carolina Parada, Kanishka Rao, Pierre Sermanet, Alexander T Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Mengyuan Yan, Noah Brown, Michael Ahn, Omar Cortes, Nicolas Sievers, Clayton Tan, Sichun Xu, Diego Reyes, Jarek Rettinghouse, Jornell Quiambao, Peter Pastor, Linda Luu, Kuang-Huei Lee, Yuheng Kuang, Sally Jesmonth, Nikhil J. Joshi, Kyle Jeffrey, Rosario Jauregui Ruano, Jasmine Hsu, Keerthana Gopalakrishnan, Byron David, Andy Zeng, and Chuyuan Kelly Fu. Do as i can, not as i say: Grounding language in robotic affordances. In Karen Liu, Dana Kulic, and Jeff Ichnowski (eds.), *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 287–318. PMLR, 14–18 Dec 2023. URL https://proceedings.mlr.press/v205/ichter23a.html.

Hyun Joe Jeong, Gokul Swamy, and Andrea Bajcsy. Learning what to say to your vla: Mostly harmless vision language action model steering. *arXiv preprint arXiv:2606.12299*, 2026.

Yuhua Jiang, Junjie Lu, Xinyao Qin, Xiaoyu Chen, Kaixin Wang, Feifei Gao, and Li Zhao. Vla-gse: Boosting parameter-efficient fine-tuning in vla with generalized and specialized experts. *arXiv preprint arXiv:2605.06175*, 2026.

Pegah Khayatan, Sara Meziane, Jayneel Parekh, and Matthieu Cord. Dimas: Distribution matching for steering vision-language-action models. *arXiv preprint arXiv:2607.14280*, 2026.

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

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. In *Proceedings of The 8th Conference on Robot Learning*, pp. 2679–2713, 2025b.

Seungjae Lee, Yibin Wang, Haritheja Etukuru, H. Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 26991–27008. PMLR, 21–27 Jul 2024.

Runhao Li, Wenkai Guo, Zhenyu Wu, Changyuan Wang, Haoyuan Deng, Zhenyu Weng, Yap-Peng Tan, and Ziwei Wang. Map-vla: Memory-augmented prompting for vision-language-action model in robotic manipulation. *arXiv preprint arXiv:2511.09516*, 2025.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 44776–44791. Curran Associates, Inc., 2023. doi: 10.52202/075280-1939. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/8c3c666820ea055a77726d66fc7d447f-Paper-Datasets_and_Benchmarks.pdf.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. RDT-1b: a diffusion foundation model for bimanual manipulation. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=yAzN4tz7oI.

Tenglong Liu, Jianxiong Li, Yinan Zheng, Haoyi Niu, Yixing Lan, Xin Xu, and Xianyuan Zhan. Skill expansion and composition in parameter space. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=GLWf2fq0bX.

Yuankai Luo, Woping Chen, Tong Liang, and Zhenguo Li. Coral: Scalable multi-task robot learning via lora experts. *arXiv preprint arXiv:2603.09298*, 2026.

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In Leslie Pack Kaelbling, Danica Kragic, and Komei Sugiura (eds.), *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pp. 1113–1132. PMLR, 30 Oct–01 Nov 2020.

Atharva Mete, Haotian Xue, Albert Wilcox, Yongxin Chen, and Animesh Garg. Quest: Self-supervised skill abstractions for learning continuous control. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 4062–4089. Curran Associates, Inc., 2024. doi: 10.52202/079017-0133. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/076c3e48fa502c660902105965fdd9f6-Paper-Conference.pdf.

OpenAI. Gpt-6 astra: A new generation of intelligence, 2026. URL https://openai.com/index/gpt-6-astra/. OpenAI Blog.

Karl Pertsch, Youngwoon Lee, and Joseph J. Lim. Accelerating reinforcement learning with learned skill priors. In Jens Kober, Fabio Ramos, and Claire J. Tomlin (eds.), *4th Conference on Robot Learning, CoRL 2020, 16-18 November 2020, Virtual Event / Cambridge, MA, USA*, volume 155 of *Proceedings of Machine Learning Research*, pp. 188–204. PMLR, 2020. URL https://proceedings.mlr.press/v155/pertsch21a.html.

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.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. AdapterFusion: Non-destructive task composition for transfer learning. In Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty (eds.), *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 487–503, Online, April 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.39. URL https://aclanthology.org/2021.eacl-main.39/.

Archit Sharma, Shixiang Gu, Sergey Levine, Vikash Kumar, and Karol Hausman. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=HJgLZR4KvH.

Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. Memoryvla: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 18567–18602, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/1fa3d6ccbcd15f7285fee666b2bd57be-Paper-Conference.pdf.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6309–6318, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Yuqi Wang, Xinghang Li, Wenxuan Wang, Junbo Zhang, Yingyan Li, Yuntao Chen, Xinlong Wang, and Zhaoxiang Zhang. Unified vision-language-action model. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=PklMD8PwUy.

Tianhang Yang, Yanze Zheng, Junjie Wang, Wei-Bin Kou, Ruotong Li, and Yujiu Yang. Route by kinematics, act by observation: Kinematics-supervised expert routing in moe-augmented vla. *arXiv preprint arXiv:2607.26807*, 2026.

Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. *arXiv preprint arXiv:2403.03954*, 2024.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

Yifeng Zhu, Peter Stone, and Yuke Zhu. Bottom-up skill discovery from unsegmented demonstrations for long-horizon robot manipulation. *IEEE Robotics and Automation Letters*, 7(2):4126–4133, 2022.

APPENDIX CONTENTS

- **A.** Implementation and Training Details
- **B.** Motor-Skill Discovery
- **C.** Skill Parameterization and Fitting
- **D.** Vision–Language Skill Planning
- **E.** Evaluation Protocols
- **F.** Skill Activation and Executed Sequences
- **G.** Real-Robot Task Composition
- **H.** Discussion

This appendix supports the three linked stages of PLUGVLA: discovering recurring local motion patterns, instantiating them as detachable parameter specializations, and planning their activation from vision–language context. The central distinction is that a discovered pattern is not used only as a discrete label. It indexes an independently activatable update to a shared VLA. We provide the implementation details, evaluation protocols, and additional behavioral analyses needed to interpret this design.

A IMPLEMENTATION AND TRAINING DETAILS

A.1 TRAINING STAGES

Training proceeds in four stages. First, supervised fine-tuning (SFT) adapts the shared $\pi_{0.5}$ policy to the benchmark demonstrations. Second, the factorized VQ-VAE is trained on local action windows and then used to assign motion-pattern labels. Third, each selected pattern trains one parameter increment while the SFT policy remains frozen. Finally, the skill bank and SFT policy are frozen, and only the vision–language router is optimized. This ordering prevents router training from changing either the shared generator or the behaviors represented by the fitted increments. Once fitted, each update can be activated or bypassed independently without modifying the shared SFT policy. This separation provides the plug-and-play interface used in both within-benchmark evaluation and cross-task skill-bank reuse.

Table 5: Trainable and frozen components at each stage.

Stage	Trainable component	Frozen component or output
Base adaptation	Shared VLA	Produces θ_{SFT}
Skill discovery	Factorized VQ-VAE	Produces motion-pattern labels
Skill fitting	Assigned $\Delta\theta_k$	θ_{SFT} and all non-assigned updates
Router fitting	Router R_ψ	θ_{SFT} and the complete skill bank

A.2 BENCHMARK-SPECIFIC IMPLEMENTATION

For LIBERO, physical actions contain three translational, three rotational, and one gripper component. The policy uses the standard 32-dimensional $\pi_{0.5}$ action interface, with unused dimensions padded consistently with the reference implementation. Skill discovery and skill fitting use 16-step windows. The skill bank contains $K = 16$ updates, each with LoRA rank $r = 16$ and scaling $\alpha = 16$.

For RoboTwin 2.0, the implementation uses the Aloha–AgileX embodiment. Each observation contains the high camera and both wrist cameras. The physical state and action are 14-dimensional joint vectors, embedded in the 32-dimensional model interface. The policy predicts 50-step action chunks at 15 Hz. Skill discovery uses three ordered temporal slots over each 50-step window and a factorized codebook of size $5 \times 5 \times 4$ for end-effector position, orientation, and gripper state. The router predicts the current and next chunk and replans after each executed 50-step chunk. These settings follow the benchmark-specific policy interface; they do not imply that the LIBERO and RoboTwin motion dictionaries share code identities.

A.3 Ablation implementation

The *Random Routing* ablation keeps the trained policy and skill bank fixed but replaces the learned router with a frequency-matched random selector. The *Joint VQ* ablation replaces end-effector motion with native joint-space trajectories during skill discovery while retaining the subsequent skill-fitting and routing pipeline. Thus, the two ablations respectively test context-conditioned selection and the choice of discovery representation.

B Motor-Skill Discovery

B.1 Factorized motion representation

For LIBERO, the discovery model receives a normalized 16-step window with seven delta-end-effector and gripper channels. Type and temporal positional embeddings are followed by a temporal convolution with kernel and stride four, yielding four ordered subwindow representations. Separate heads encode translation, rotation, and gripper motion with latent dimensions 64, 32, and 16 and codebook sizes 5, 3, and 4, respectively. The codebooks are updated by exponential moving average with decay 0.8.

The decoder is used only to train the representation. Its conditioning inputs, including the initial robot state and first action, are never available to the deployed router. Let M and $\hat{M}$ denote a motion window and its reconstruction. The discovery objective is

$$\mathcal{L}_{\mathrm{VQ}} = \sum_{d=1}^{7} \gamma_d \,\mathrm{MSE}(\hat{M}_{:,d}, M_{:,d}) + \beta_{\mathrm{com}} \mathcal{L}_{\mathrm{com}} - \beta_{\mathrm{ent}} \sum_{b} \mathcal{H}(\bar{p}^b), \tag{12}$$

where $\gamma = (1, 1, 1, 5, 5, 5, 5)$, $\beta_{\mathrm{com}} = 0.25$, and $\beta_{\mathrm{ent}} = 0.1$. The commitment term trains the encoder toward its selected entries, while the entropy term discourages premature codebook collapse. Straight-through quantization supplies encoder gradients. After training, the decoder is discarded.

B.2 Pattern indexing and selection

At each temporal position, the component tuple is mapped to one atomic index by

$$q^{\mathrm{atomic}} = 12q^{\mathrm{pos}} + 4q^{\mathrm{ori}} + q^{\mathrm{grip}}. \tag{13}$$

The ordered indices from all temporal positions define a whole-window pattern. We count exact pattern occurrences in the demonstration collection and retain the $K = 16$ most frequent patterns for skill fitting. An unselected pattern is mapped to the base branch. This frequency-based selection retains a compact set of well-supported recurring motion patterns for subsequent parameterization. Whether these selected patterns are useful as reusable skills is evaluated downstream through closed-loop performance, the Joint VQ ablation, and cross-task skill-bank reuse. Frequency determines which patterns are parameterized; it does not by itself determine their control value.

RoboTwin uses the same factorized principle but applies it to its 50-step, dual-arm end-effector motion representation. Because the embodiments and action conventions differ, pattern IDs are defined independently for each benchmark and are not compared across skill banks.

B.3 Qualitative structure of the discovered motion codes

We first inspect the gripper component of the factorized codebook. As shown in Figure 6, the four codes capture distinct local command profiles around the two saturated gripper states. Codes 0 and 2 approach stable full-closure and full-opening commands, respectively, whereas Codes 1 and 3 move away from those command limits. The shaded variation indicates that the codes summarize recurring transition tendencies rather than prescribing a single deterministic trajectory. Because these are action commands, the curves do not by themselves establish contact, grasp success, or applied force.

Then, Figure 7 colors the same type of local action segments by their source LIBERO suite. Suite labels are broadly intermixed within local neighborhoods, although some regions retain suite-dependent density. Thus, the embedding is consistent with related local motions recurring across task families instead of being partitioned entirely by suite identity.

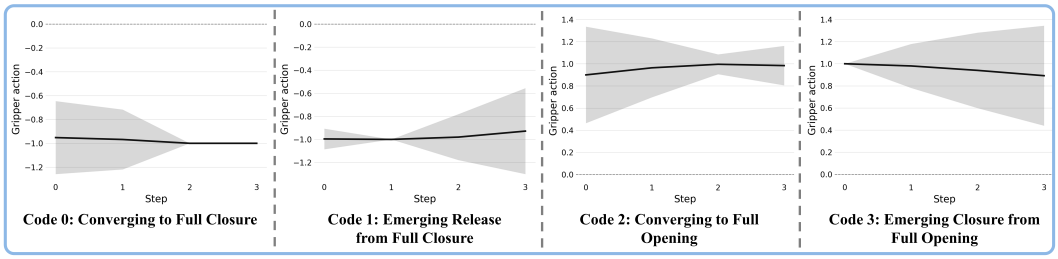

Figure 6: Factorized VQ captures structured gripper-action transitions. Negative and positive values denote closing and opening commands, respectively. Codes 0 and 2 converge toward full-closure and full-opening commands, while Codes 1 and 3 capture transitions away from these saturated command limits. Solid curves show the mean action at each step, and shaded regions denote one standard deviation.

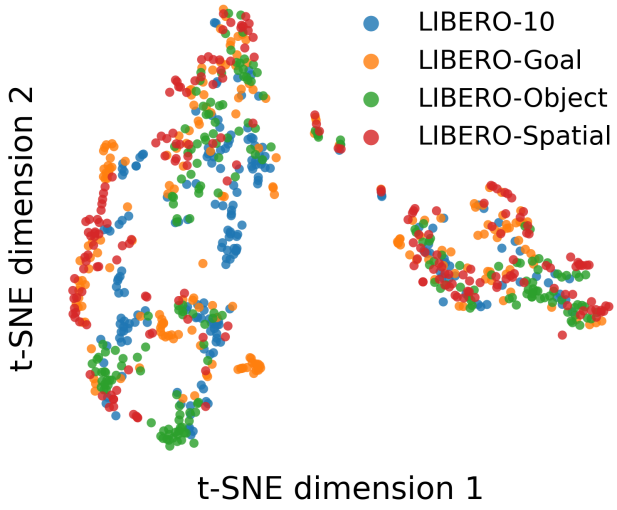

Figure 7: Local motion structure is shared across LIBERO task suites. Each point represents an action segment and is colored by its source suite. The broad mixing of suite labels within local neighborhoods is consistent with recurring local motions shared across task families, while localized density differences show that the distributions are not identical.

C Skill Parameterization and Fitting

C.1 Action-expert updates

In our formulation, the discrete motion pattern is not left as a latent code or routing label. Each selected pattern indexes an independent parameter specialization that can alter action generation when invoked. Each discovered pattern is therefore instantiated as an independently activatable parameter specialization of the same frozen task-adapted policy.

We insert skill-specific updates into all 18 action-expert Transformer blocks of $\pi_{0.5}$. Each skill modifies the gate and up projections of the gated feed-forward network through rank-16 LoRA factors and adds a vector increment $\Delta m_k^{(l)}$ to the residual gate. A block can be written schematically as

$$h^+ = h + \left(m^{(l)}(u) + \Delta m_k^{(l)}\right) \odot W_{\text{down}}^{(l)} \left[\sigma\left((W_{\text{gate}}^{(l)} + \Delta W_{\text{gate},k}^{(l)})\tilde{h}\right) \odot \left((W_{\text{up}}^{(l)} + \Delta W_{\text{up},k}^{(l)})\tilde{h}\right)\right]. \tag{14}$$

Here u is flow time. The down projection, attention layers, vision–language backbone, normalization parameters, and action projections remain shared and frozen. With action-expert width 1,024, MLP width 4,096, $K = 16$, and 18 blocks, the complete skill bank contains 47,480,832 trainable parameters. A single selected skill activates 2,967,552 of these parameters.

This restriction concentrates adaptation on gated feature formation and residual modulation while preserving a shared output projection. The resulting skill modules specialize local action generation while leaving the shared task-adapted policy unchanged. Only one 2.97M-parameter update is selected for a policy call, whereas the complete 16-skill bank contains 47.48M parameters.

C.2 Skill fitting objective

We retain the native flow-matching objective of $\pi_{0.5}$ (Black et al., 2025; Lipman et al., 2023). For action chunk A, flow time u, and Gaussian noise ϵ,

$$A_u = (1-u)A + u\epsilon, \qquad v^\star = \epsilon - A, \qquad \mathcal{L}_{\mathrm{FM}} = \|v_\vartheta(A_u, u, o, \ell) - v^\star\|_2^2 . \tag{15}$$

For a sample assigned to pattern k, we set $\vartheta = \theta_{\mathrm{SFT}} + \Delta\theta_k$ and update only $\Delta\theta_k$. The shared SFT policy and all other skill increments remain fixed. This allocation fits each update to its assigned motion subset but does not require two updates to induce mutually exclusive behaviors.

The base route sets every increment to zero. Consequently, at fixed inputs, sampling noise, precision, and integration settings, the augmented model on the base route recovers the SFT computation exactly:

$$\hat{A}(x, \epsilon; \theta_{\mathrm{SFT}} + \Delta\theta_{\mathrm{base}}) = \hat{A}(x, \epsilon; \theta_{\mathrm{SFT}}), \qquad \Delta\theta_{\mathrm{base}} = 0. \tag{16}$$

This identity gives the router an unchanged reference-policy branch in addition to the 16 parameterized skills.

D Vision–Language Skill Planning

D.1 Router inputs and architecture

Skill discovery determines what reusable local motions are available, whereas the router determines when their fitted parameterizations should be activated from the current task context. It therefore conditions selection on the current visual, linguistic, and robot-state context rather than replaying the discovery label.

The router receives only information available at deployment. Multi-view images are encoded into visual tokens, while the instruction and discretized robot state are included in the prefix prompt. The frozen $\pi_{0.5}$ prefix stream therefore already combines visual, linguistic, and state information. The router reads its final 2,048-dimensional token representations and the valid-token mask; it does not receive future observations, demonstration actions, discovery labels, candidate losses, or success outcomes.

Sixteen learned queries attend to the valid prefix tokens through 256-dimensional key and value projections. The pooled features pass through a $4096 \to 512 \to 256$ fusion stack. Learned horizon embeddings, a causal self-attention layer, and a feed-forward layer produce two planning positions. Each position has 17 logits: 16 skill updates and one explicit base class. The deployed model applies only the first prediction; the second is an auxiliary look-ahead prediction and is recomputed at the next policy call.

D.2 Candidate-wise supervision

Before router training, we evaluate the frozen base and all 16 frozen updates on each valid demonstration window. All candidates use the same target action and paired flow-time/noise samples. The lowest-loss candidate defines $k^\star_\tau$ in Eq. (10). The original discovery label z_τ is retained as weaker auxiliary supervision. For planning position j, the classification loss is

$$\ell_{t,j} = \mathrm{CE}(p_{\psi,j}, k^\star_{\tau_{t,j}}) + 0.1\,\mathrm{CE}(p_{\psi,j}, z_{\tau_{t,j}}). \tag{17}$$

The two positions are weighted by $w_0 = 1$ and $w_1 = 0.8$. Candidate losses are used only to form fixed class targets; they are not deployment inputs and are not interpreted as task returns.

D.3 Receding-horizon execution

Let H_{act} be the generated action-chunk length, P the number of skill-plan positions, and E the executed prefix length. The planning anchors are $t, t + H_{\mathrm{act}}, \ldots, t + (P-1)H_{\mathrm{act}}$. At each policy

call, the router predicts all P positions, the action generator uses the first selected route for the entire flow integration, and the environment executes at most E actions. Any remaining actions and unused plan positions are then discarded. A new observation produces a new plan. RoboTwin uses $H_{\text{act}} = E = 50$ and $P = 2$; LIBERO follows the action horizon used by its matched evaluation configuration.

E Evaluation Protocols

E.1 Common comparison protocol

For each benchmark, skill discovery, skill fitting, and router fitting use the designated demonstration collection. The SFT reference and PlugVLA share the same task-adapted base checkpoint. Comparisons match observation inputs, action normalization, generation and execution horizons, episode limits, and task initializations. Reported success rates use the benchmark's binary success criterion and are macro-averaged over the stated tasks or suites. Literature baselines in the main tables retain their published protocols; reproduced $\pi_{0.5}$ results are marked separately in the main paper.

E.2 LIBERO

We evaluate the four standard LIBERO suites: Spatial, Object, Goal, and Long (Liu et al., 2023). Each suite contains ten tasks, and the main table reports 500 rollouts per suite. Spatial varies target locations, Object varies the manipulated objects, Goal varies task goals under related scenes, and Long contains multi-stage, long-horizon instructions. The four suite means are averaged with equal weight.

E.3 RoboTwin 2.0

RoboTwin 2.0 evaluates dual-arm manipulation with the Aloha–AgileX embodiment (Chen et al., 2026a). Training uses clean demonstrations. The Easy setting uses clean scenes and seen instructions, whereas the Hard setting uses randomized scenes and unseen instruction variants. The main tables report 100 rollouts per task. The 9-task table is evaluated under Hard; the challenging 8-task table reports Easy and Hard separately.

Table 6: RoboTwin 2.0 tasks used in the main evaluation. Descriptions summarize the commanded objective; success is determined by the benchmark evaluator.

Task set	Tasks
9-task set	Adjust Bottle (adjust a bottle pose); Click Alarm (press the alarm clock); Dump Bigbin (empty an item into a large bin); Grab Roller; Move Card; Place A2B Right (right-arm transfer); Bread Basket; Burger Fries; Container Plate.
8-task set	Handover Block; Hang Mug; Move Can; Open Laptop; Place Dual Shoes; Place Mouse Pad; Rotate QR Code; Turn Switch.

The reported RoboTwin evaluations use only the 9-task and 8-task sets listed above. *Click Bell* is retained solely as an auxiliary qualitative case for interpreting E5 in Section F.2; it does not define an additional evaluation set, and no *Click Bell* outcome is included in the reported 9-task average.

E.4 Cross-task skill-bank reuse

For each transfer direction, the skill bank learned on the source RoboTwin task set is frozen. Only the router is trained on demonstrations from the target set, after which the resulting policy is evaluated on target tasks. The target demonstrations therefore provide context for learning when to invoke source skills, but they do not update the transferred skill parameters. This protocol tests whether frozen skill parameters remain reusable under a new target-task context after adapting only the planner; it does not test zero-shot task transfer. As reported in the main paper, this protocol improves over $\pi_{0.5}$ by 6.4 and 0.9 percentage points in the two transfer directions, providing direct evidence that the transferred skill parameters remain useful without refitting.

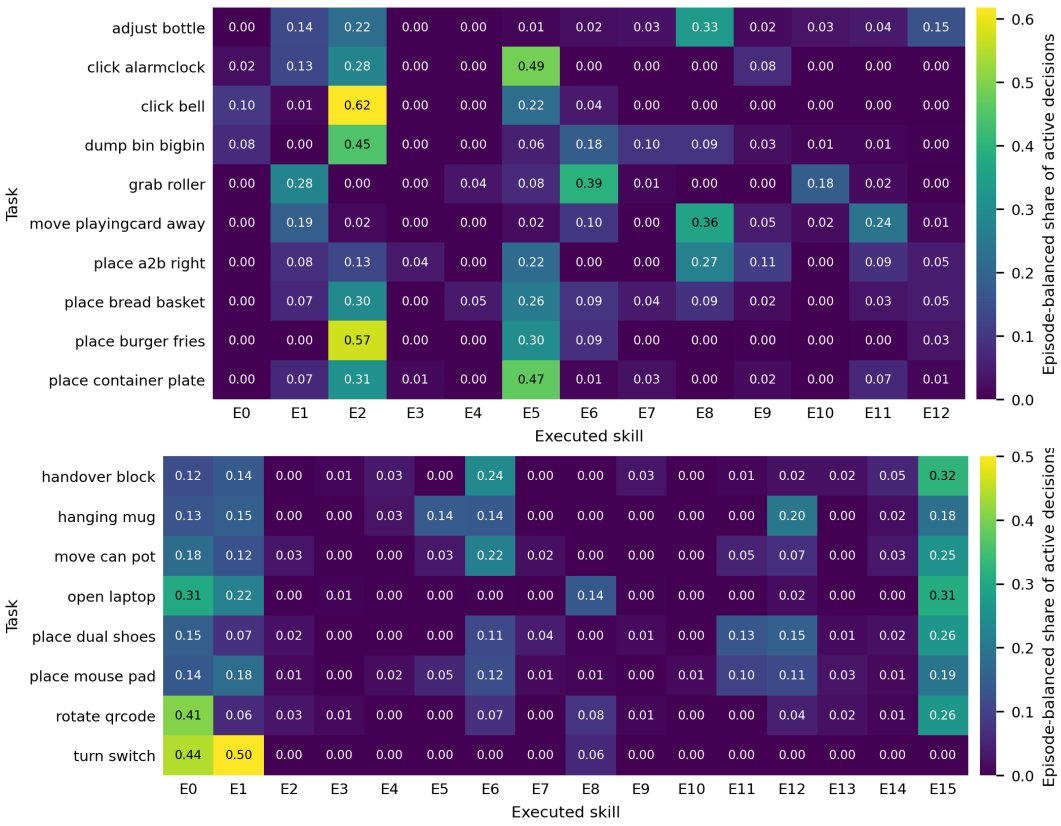

Figure 8: Task-conditioned execution frequencies of non-base motor skills for the 9-task and 8-task RoboTwin sets. The top panel reports the 9-task set and additionally displays *Click Bell* as the auxiliary E5 diagnostic used in Section F.2; this row is not part of the 9-task evaluation. The bottom panel reports the 8-task set. Each row is normalized over active skill decisions; base selections are excluded.

E.5 Ablations

The full model and both ablations use the same benchmark task sets and rollout budgets. Random Routing preserves the bank but removes observation- and instruction-conditioned selection. Joint VQ preserves the downstream model and optimization stages but changes the discovery input. As reported in Table 4, Random Routing reduces success by 2.0, 7.2, and 7.1 percentage points on LIBERO, RoboTwin-8, and RoboTwin-9, respectively, while Joint VQ reduces success by 1.0, 4.6, and 3.5 points. These controlled comparisons support the contributions of context-conditioned planning and the EEF-based discovery representation, respectively; they do not isolate every architectural choice within the router or the VQ-VAE.

F Skill Activation and Executed Sequences

Figure 8 summarizes which non-base skills were executed in each RoboTwin task. Rows are normalized over active decisions, so the plots show the distribution among selected skills rather than the fraction of time spent on the base route. Repeated use of the same update across tasks directly shows cross-task reuse at the routing level; its local motion semantics are examined separately through the trajectory analyses below.

F.1 Executed skill sequences

Figures 9 and 10 retain the order of the first-choice route executed at successive replanning steps. `B` denotes the unchanged base route and `E#` denotes a skill update. These are realized routing traces,

not manually annotated task phases. They complement marginal activation frequencies by showing that multiple parameterized skills and the base route are invoked sequentially within task execution.

F.2 Interpreting an executed parameterized skill

Cross-episode motion pattern associated with E5. To examine whether an executed update has consistent local motion semantics, we inspect end-effector trajectories associated with E5 across different episodes. Figure 11 shows that E5 is repeatedly activated after gripper closure during segments in which the end effector moves vertically downward. Although the surrounding task contexts and trajectories differ, the activated segments share this local motion tendency. This cross-episode consistency supports interpreting E5 as a recurring downward-motion pattern, rather than as an identifier tied to one episode. It does not imply that E5 has an exclusive semantic meaning in every state where it may be selected.

Representative execution with E5. We further examine a representative *Click Bell* episode in Figure 12. The base policy fails to complete the press in this episode, whereas our routed policy succeeds. Following the transition from E3 to E5, the routed execution exhibits the downward motion identified in the cross-episode analysis and reaches the bell. The alignment between the selected update and the ensuing trajectory provides complementary evidence that E5 is not merely an activation label: the routed execution exhibits an observable local change after E5 is selected. This example illustrates the mechanism in one rollout and is not used as a substitute for the aggregate success rates reported in the main paper.

G Real-Robot Task Composition

G.1 Platform and tasks

The physical experiments use a RealMan dual-arm lift platform with two RM65-B arms. Task A trains a stacking behavior, and Task B trains grasping of a yellow bag. The held-out Task C instructs the robot to place the yellow bag on a green box. Task C therefore tests whether the planner can compose behaviors from the shared base policy and skill updates learned from Tasks B and A. No Task-C-specific skill parameters are fitted.

G.2 Interpretation

The real-robot result is a task-composition study. It shows that skill parameters learned from the two training tasks can be reused under the held-out Task C without fitting Task-C-specific skill parameters. Because the study uses one platform and a limited set of objects, it provides evidence for cross-task skill reuse under this held-out composition rather than for general zero-shot or open-world manipulation.

H Discussion.

The experiments instantiate the framework on $\pi_{0.5}$ with skill banks derived from fixed demonstration collections. The current study focuses on recurring motion patterns that are sufficiently represented in these demonstrations. Extending the framework to broader behavior coverage, additional VLA backbones, larger task sets, and more diverse real-world environments remains an important direction for future work.

We do not assume that each discovered skill corresponds to a unique human-interpretable semantic category. Instead, the discovered skills are defined by recurring motion patterns observed in the demonstrations, without requiring predefined behavior labels. The resulting skill IDs therefore serve as data-driven indices of reusable motor patterns, whose functional roles can be examined afterward through activation statistics and trajectory analysis. This design supports our goal of discovering reusable motor structure from multi-task demonstrations without imposing a manually designed skill taxonomy.

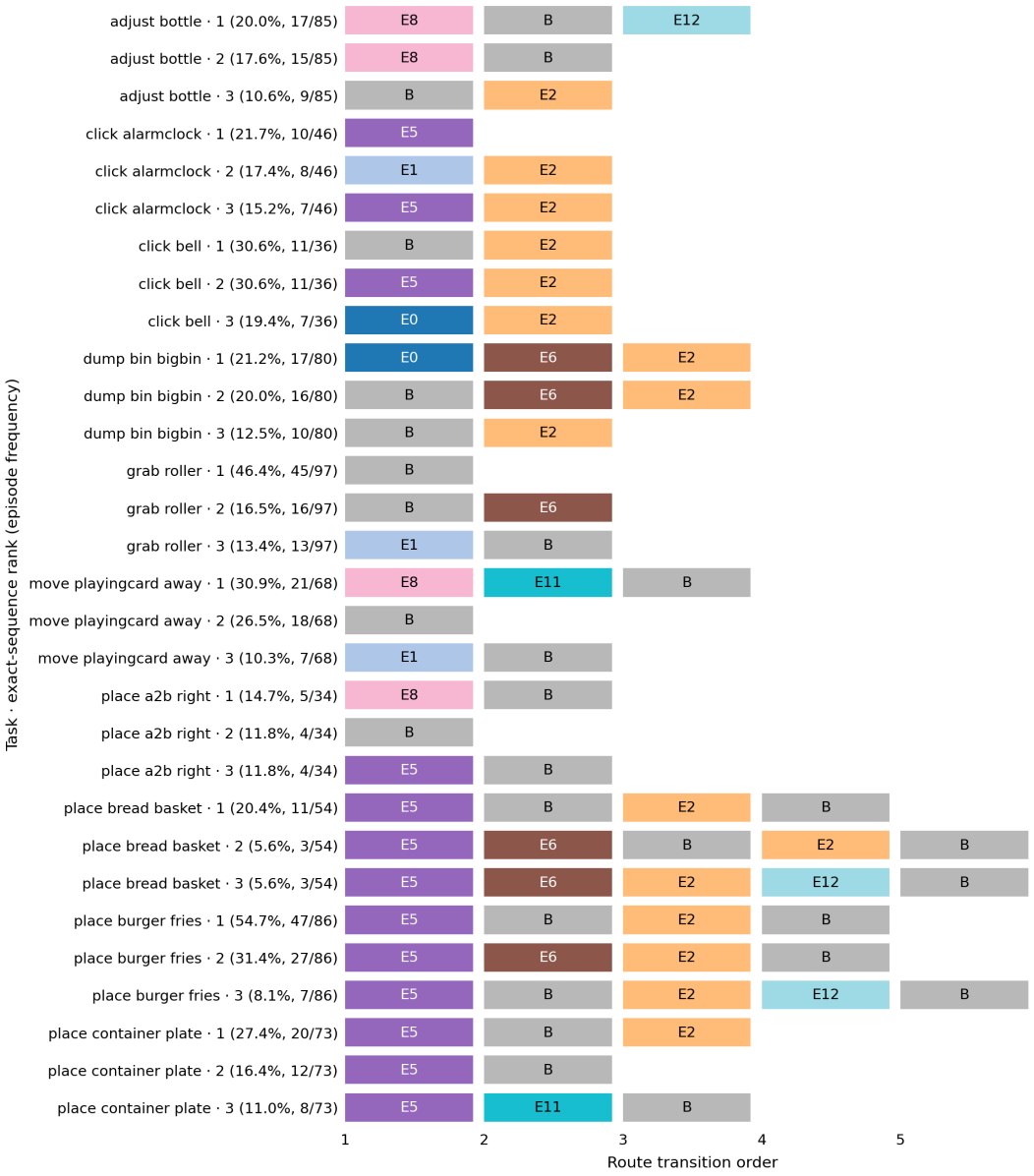

Figure 9: Most frequent executed routing sequences for the 9-task RoboTwin set, with *Click Bell* included only as an auxiliary E5 diagnostic. Parentheses report the observed episode frequency and count; the *Click Bell* traces are excluded from all 9-task aggregate results.

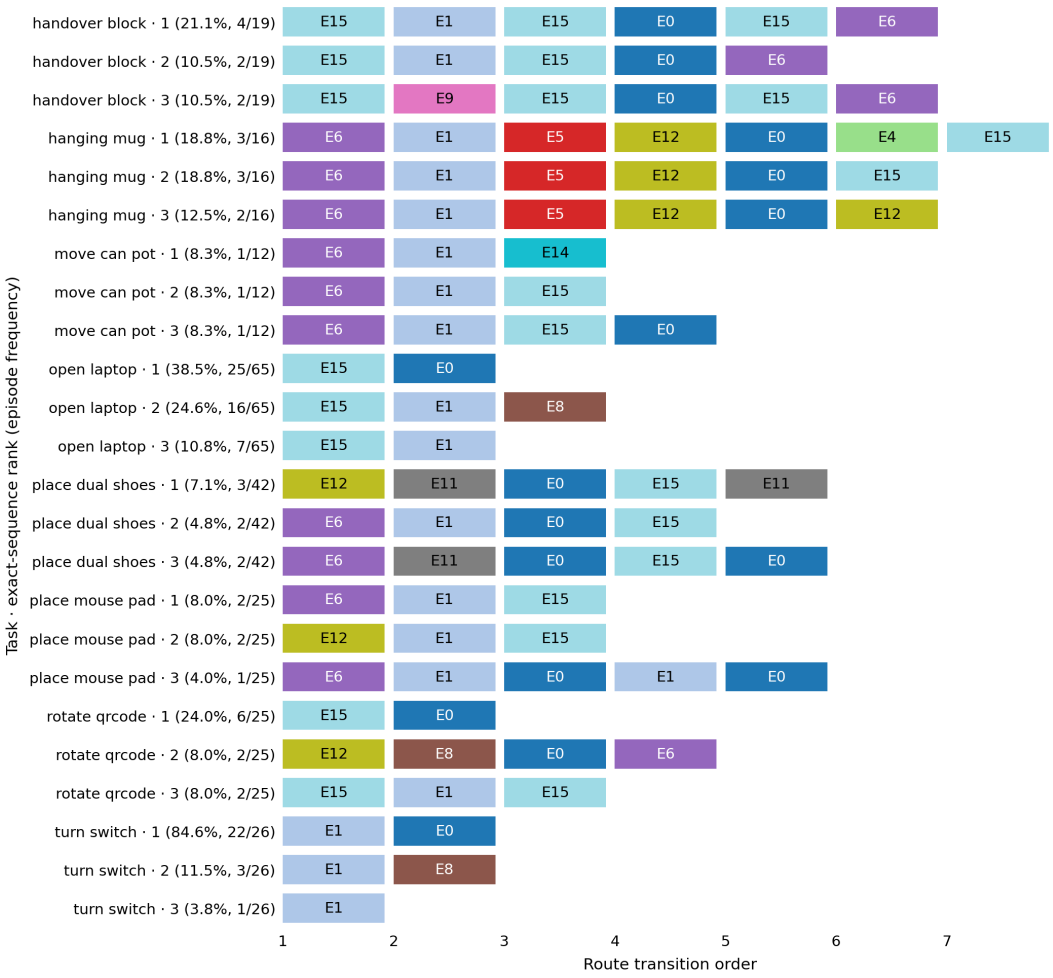

Figure 10: Most frequent executed routing sequences in the challenging 8-task RoboTwin evaluation. Parentheses report the observed episode frequency and count.

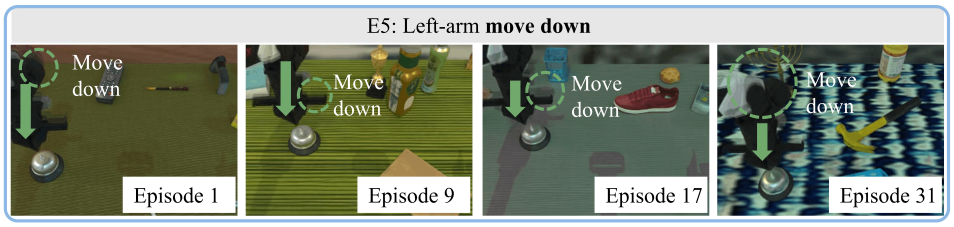

Figure 11: Cross-episode motion pattern associated with E5. Across the shown episodes, E5 is repeatedly activated during vertical downward motion following gripper closure. The highlighted trajectory segments exhibit a consistent local tendency despite differences in the surrounding task context.

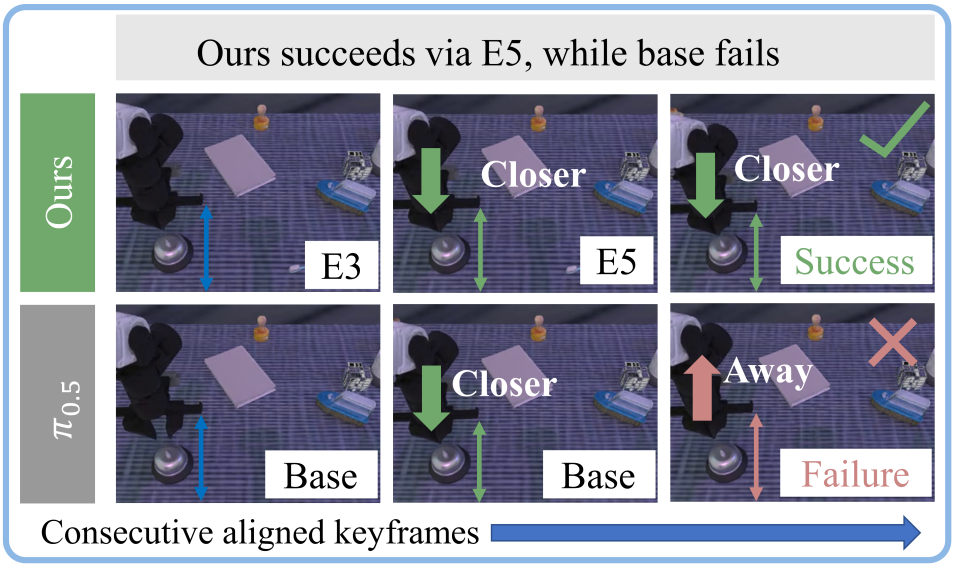

Figure 12: Representative execution with E5 on *Click Bell*. Our routed policy completes the bell press, whereas the base policy fails in the shown episode. Following the transition from E3 to E5, the routed execution exhibits the characteristic downward motion observed across episodes and reaches the bell.