WHAT DOES A SKILL VOCABULARY COST? A CLOSED-LOOP DECOMPOSITION OVER DEMONSTRATED MOTIONS

Maximilian Xiling Li[1], **Nils Blank**[1], **Paul Mattes**[1],
Pankhuri Vanjani[1] **& Rudolf Lioutikov**[1,2]
[1]Intuitive Robots Lab, Karlsruhe Institute of Technology (KIT), Germany
[2]Robotics Institute Germany (RIG)
{maximilian.li, lioutikov}@kit.edu

ABSTRACT

Discrete skill policies execute motions drawn from a finite vocabulary. In most current methods, this vocabulary is a latent codebook learned jointly with a selector and a decoder. When such a policy fails in closed loop, the failure cannot be traced to the vocabulary, to the choice of skill or to its execution. We present Movement Prototype Quantization (MPQ), which places a library of demonstrated motions after a trained policy instead of inside it. MPQ replaces each predicted action chunk with its nearest library element, corrects that element toward the prediction within a bound, and trains no parameters. Across six architectures, including a 7B vision-language-action model, the wrapped policy is statistically equivalent to the original. A reference policy executed through the library is therefore a baseline for any selector over the same library. Substituting a selector's components one at a time attributes its shortfall to them. On LIBERO-90, with selectors on three backbones, quantization costs nothing and the choice of skill costs at most 2 points under the reference's placement and correction. The selectors' own placement and correction account for nearly all of the remaining gap, up to 12 points. Offline reconstruction predicts none of these differences and misjudges library size, design and transfer across benchmark suites. Project page: https://intuitive-robots.github.io/mpq.

1 INTRODUCTION

A policy that predicts action chunks can be given a finite vocabulary of motions in one of two places. The vocabulary can sit inside the policy, as a latent codebook learned jointly with a selector and a decoder (Lee et al., 2024; Mete et al., 2024; Zheng et al., 2024). It can also sit after the policy and be applied to the predictions of a continuous policy, which then trains as usual while every chunk it executes belongs to a finite set. Either placement allows a pretrained vision-language model to emit actions as tokens (Pertsch et al., 2025), but the two differ in what the vocabulary costs. This paper measures that cost.

Placing the vocabulary inside the policy has two consequences. First, a codebook trained jointly with its selector and decoder entangles the costs of the vocabulary, the selection and the execution, so closed-loop success reports only their sum. Second, a code becomes a trajectory only through the learned decoder, so the library is judged by offline reconstruction rather than by closed-loop success.

We separate these costs by placing the vocabulary after the policy. If a continuous reference policy keeps its success when executed through an explicit library, the library covers the motions the task requires. The reference executed through the library, the *library reference*, is then a baseline for any selector over that library. A trained selector's shortfall against it can be split into the cost of its choice and the cost of its placement and correction (§3.1). This requires two things. Every library element must execute without a learned decoder, and the layer that applies the library must cost the reference nothing.

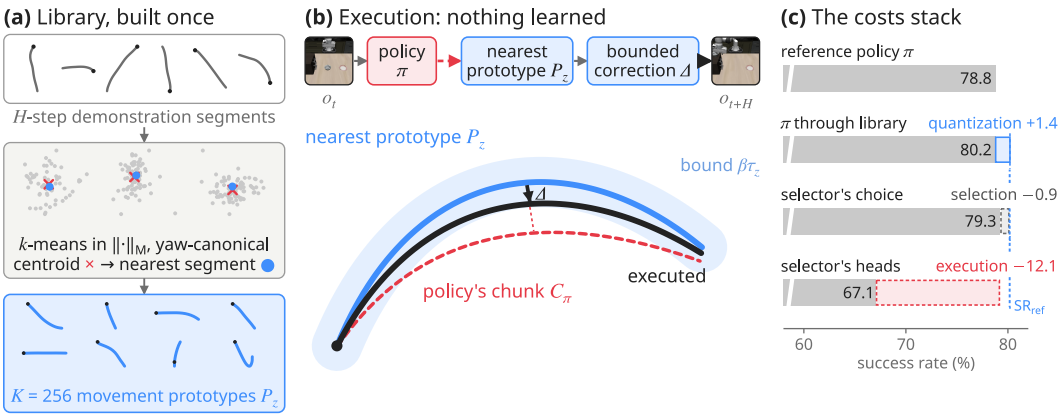


Figure 1: Movement Prototype Quantization. **(a)** Library construction. Demonstration segments are fit to B-spline control points, yaw-canonicalized, clustered under the Gram metric and projected onto real segments (LIBERO-90). **(b)** Execution. The chunk C_π of a frozen policy is matched to its nearest movement prototype P_z, placed in the chunk's frame and corrected toward C_π within the deviation bound (Eq. 5). Blue: demonstrated. Red: predicted. Black: executed. **(c)** Decomposition. A reference policy executed through the library ($\mathrm{SR}_{\mathrm{ref}}$, dashed) sets the baseline. Substituting a selector's choice, then its placement and correction, splits the selector's shortfall into selection and execution (LIBERO-90, Table 3).

Movement Prototype Quantization (MPQ) provides such a layer. Its elements, *movement prototypes*, are demonstration segments stored as B-spline control points (Zhou et al., 2025). A fixed linear map decodes them, so every element is an executable trajectory. At execution time, MPQ replaces each predicted chunk with its nearest movement prototype and corrects it toward the prediction, within a bound set by the demonstrations' own spread (Fig. 1). Nothing on the execution path is trained.

The layer costs the policies it wraps almost nothing (§4). Pooled over 13 combinations of policy and benchmark on LIBERO (Liu et al., 2023) and RoboCasa (Nasiriany et al., 2024), covering six architectures including a 7B vision-language-action model, MPQ changes success by $+0.6$ points (95% CI $[-0.8, +1.9]$). Its cost does not grow under observation shift, and on a real robot it runs with the bound selected in simulation.

Relative to the library reference, selectors on three backbones lose almost all of their gap in placing and correcting the chosen element (§5). On LIBERO-90 with 256 movement prototypes, quantization costs nothing. A selector's choice often differs from the nearest movement prototype, yet with the reference's placement and correction it costs at most 2 points. The selectors' own placement and correction cost between 0 and 12 points.

Offline reconstruction, the standard criterion for validating a library, predicts none of these differences (§6). It keeps improving after closed-loop success has saturated, it ranks library designs differently than closed-loop success does, and it misses a transfer loss of 4 to 8 points. As with policy checkpoints (Mandlekar et al., 2021), offline criteria therefore cannot select a library.

Contributions.

- A closed-loop decomposition that splits a discrete skill policy's shortfall into quantization, selection and execution, measured against the library reference (§3.1).
- MPQ, an execution layer that keeps every executed chunk within a bounded distance of a demonstrated segment and is statistically equivalent to the unwrapped policy, within the noise of repeated evaluation (§3, §4).
- The finding that on three backbones, a selector's gap to the library reference arises in executing the chosen element rather than in the vocabulary or the choice (§5).
- Evidence that offline reconstruction misjudges library size, design and transfer (§6).

2 Related work

Discrete skill policies and how they are measured. VQ-BeT (Lee et al., 2024), QueST (Mete et al., 2024) and PRISE (Zheng et al., 2024) discretize action chunks into latent codebooks learned as vector-quantized autoencoders (van den Oord et al., 2017). FAST (Pertsch et al., 2025) and its successors (Liu et al., 2026; Kang et al., 2026) instead tokenize chunks so that autoregressive vision-language-action models can predict them. In the codebook methods, a learned head selects the element, and a code becomes a trajectory only through a learned decoder. The codebook is therefore validated offline, by reconstruction. In imitation learning, Mandlekar et al. (2021) showed that selecting checkpoints by validation loss yields policies far below the best, and evaluated them in closed loop instead. The library reference applies the same principle to the vocabulary (§6).

Selectors with learned offsets. BeT (Shafiullah et al., 2022) bins single-step actions and adds a learned offset to the selected bin. This combination of bin and offset makes it the closest structural relative of a selector over MPQ's library. It differs from MPQ in three respects, however. Its bins are means of raw single-step actions, its selector is a trained classifier, and its offsets are unbounded. Concurrent work pairs a learned selector with a residual that is free to overwrite the selected code (Lei et al., 2026; Wang et al., 2026). For such a design, Eq. 1 measures the cost of the choice and the cost of the heads separately. In MPQ the residual is bounded, so the executed segment keeps the identity of the selected element.

Motion primitives and prototypes. Our construction, which clusters demonstration segments and projects the centroids onto real ones, has classical relatives in libraries of motion primitives (Ijspeert et al., 2013; Paraschos et al., 2013; Li et al., 2023a; Lödige et al., 2025) and in skills segmented from demonstrations (Niekum et al., 2012; Konidaris et al., 2012; Lioutikov et al., 2017; Blank et al., 2024; 2026). These libraries are sequenced by planners or grammars (Lioutikov et al., 2018; 2020) rather than placed after a chunk-predicting policy. The tolerance of a movement prototype plays the role that the variance plays in a probabilistic movement primitive (Paraschos et al., 2013), except that MPQ uses it as a bound on execution rather than as a distribution to sample from. We represent segments as B-spline control points following BEAST (Zhou et al., 2025), which tokenizes chunks without a learned encoder. Other work places primitives (Scheikl et al., 2024) or splines (Tian et al., 2026; Han et al., 2026) inside the policy head. The projection of cluster centroids onto real examples follows ProtoPNet (Chen et al., 2019). We introduce no new primitive. Our contribution lies in where the library sits and in the measurement this placement enables.

Execution-time constraints. Retrieval-based policies match *observations* to demonstrations and leave the executed action unconstrained (Pari et al., 2022; Du et al., 2023; Odonchimed et al., 2025). MPQ instead matches *actions* at execution time, so that each executed chunk has an identity and a deviation bound. Two further lines of work act on the chunk at execution time. Bidirectional decoding selects among chunks sampled from a policy (Liu et al., 2025), and residual policies learn corrections to a fixed controller or a reference trajectory (Silver et al., 2018; Johannink et al., 2019; Huang et al., 2025). Like a residual policy, MPQ corrects a fixed base, but the base is a demonstrated segment and the correction is bounded rather than learned. Safety shields (Alshiekh et al., 2018) and control barrier functions (Ames et al., 2019) likewise project actions, but onto an implicit safe set rather than an explicit library.

3 Movement Prototype Quantization

3.1 Measuring a vocabulary against a library reference

A vocabulary can be measured in closed loop only against a policy that already succeeds without it. Let $\mathrm{SR}(\cdot)$ denote closed-loop success rate, and let π be a continuous reference policy that succeeds on the benchmark. A *library* $\mathcal{P} = \{P_k\}_{k=1}^{K}$ is a finite set of executable chunks, and an execution layer (§3.5) wraps π through this library. For each chunk that π predicts, the layer selects a library element, places it in the frame of the prediction and applies a bounded correction.

The *library reference* $\mathrm{SR}_{\mathrm{ref}}$ is the success rate of π executed through this layer. Since π succeeds without the layer, any drop in success can be attributed to the library. $\mathrm{SR}_{\mathrm{ref}}$ is a baseline rather

than an upper bound, since a selector that chooses or corrects better could exceed it. A discrete skill policy L over the same library replaces the layer's decisions with learned heads. A choice head selects the element, and further heads place and correct it. To isolate what its choice costs, we define a hybrid L_{idx} that executes L's choice with the reference's placement and correction (§3.5). The shortfall of L then decomposes as

$$\underbrace{\mathrm{SR}(\pi) - \mathrm{SR}(L)}_{\text{total gap}} = \underbrace{\mathrm{SR}(\pi) - \mathrm{SR}_{\mathrm{ref}}}_{\text{quantization}} + \underbrace{\mathrm{SR}_{\mathrm{ref}} - \mathrm{SR}(L_{\mathrm{idx}})}_{\text{selection}} + \underbrace{\mathrm{SR}(L_{\mathrm{idx}}) - \mathrm{SR}(L)}_{\text{execution}}. \tag{1}$$

The quantization term is what the library cannot represent. The selection term is what L's choice costs when the reference supplies the placement and correction, and the execution term is what L's learned placement and correction lose relative to the reference's. Eq. 1 substitutes L's choice before its placement and correction, and we call it the *choice-first* order. In the *correction-first* order, a hybrid L_{exec} executes the reference's choice with L's placement and correction. §5.2 reports both.

The decomposition holds only if library elements execute without a learned decoder, since otherwise part of any drop could come from the decoder. A latent codebook does not meet this requirement (Appendix F). The layer must also cost the reference nothing, which we verify in §4.

3.2 Representing action chunks

We represent an action chunk $A \in \mathbb{R}^{H\times d}$ of horizon H over d geometric degrees of freedom as a B-spline curve (Zhou et al., 2025) with B control points $C \in \mathbb{R}^{B\times d}$. Evaluating the B basis functions at the H timesteps gives the *collocation matrix* $\Phi \in \mathbb{R}^{H\times B}$. Decoding control points to a trajectory is then the linear map $A \approx \Phi\, C$, and fitting control points to a trajectory is its closed-form least-squares inverse. Since neither operation is learned, every point of $\mathbb{R}^{B\times d}$ decodes to a valid trajectory. The gripper is fit on the same basis as a separate channel, with control points $c^{\mathrm{g}} \in \mathbb{R}^{B}$.

Euclidean distance between control points does not equal the distance between the trajectories they produce. The *Gram matrix* $M = \Phi^{\top}\Phi$ relates the two through

$$\|\Phi(C-P)\|_F^2 \;=\; \mathrm{tr}\big((C-P)^{\top} M\,(C-P)\big) \;=:\; \|C-P\|_M^2, \tag{2}$$

so the trajectory distance can be computed on the coefficients without decoding. We call $\|\cdot\|_M$ the *Gram metric* and compute it over geometric channels, comparing the gripper separately (Eq. 3).

3.3 Canonicalizing orientation

The same motion performed in different directions should match the same library element. Because actions are end-effector deltas in a fixed base frame, position is already factored out, and only the direction the motion faces remains. We remove this direction with a *canonicalizing yaw rotation* $\hat{R}_C$, which turns the net horizontal displacement of chunk C to $+x$ and rotates position and axis-angle deltas alike. For chunks with negligible horizontal displacement, the rotation falls back to the principal horizontal axis (Appendix A.1). Library elements are stored in this canonical frame. Since the Gram metric acts along time, rotating a trajectory leaves its norm unchanged, $\|\hat{R}\,C\|_M = \|C\|_M$. The distance to a library element P with gripper control points p^{g} is

$$d(C,P) \;=\; \big\|\hat{R}_C\,C - P\big\|_M^2 \;+\; w_g^2\,\big\|c^{\mathrm{g}} - p^{\mathrm{g}}\big\|_M^2, \tag{3}$$

where the weight w_g separates an approach with the gripper opening from a lift with it closing.

We factor out only yaw, because gravity distinguishes vertical from horizontal motion. Under full 3D rotation, a vertical lift and a horizontal push would map to the same skill. It also requires rotation to act linearly on the actions, which holds for end-effector deltas but not in joint space.

3.4 Building the library

The library is extracted from demonstrations. Each demonstration is split into overlapping H-step segments, which are canonicalized and then clustered by k-means under the distance of Eq. 3. Whitening by the Cholesky factor of M turns the Gram metric into Euclidean distance, so standard k-means applies. Each centroid is then projected onto the nearest real segment, which makes

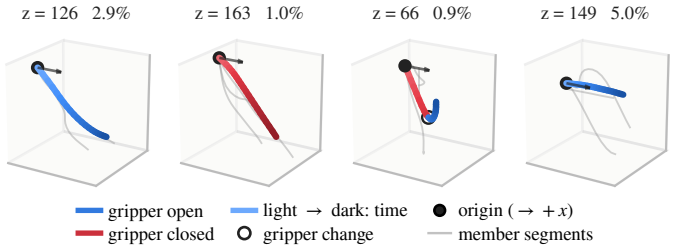


Figure 2: Four movement prototypes of the LIBERO-90 library, left to right a reach, a closed-gripper transport, a gripper transition and a near-idle motion. Headers: index z and share of closed-loop usage by wrapped ACT. Full gallery in Appendix I.

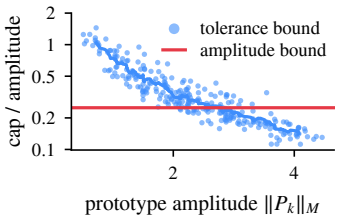


Figure 3: Cap relative to amplitude, one point per LIBERO-90 movement prototype. Blue line: running mean of the tolerance bound.

every library element a prototype of its cluster (Chen et al., 2019; Li et al., 2024). The projection is needed because a cluster average can blend two valid motions into one that is neither (Fig. 2).

Each movement prototype also carries a *tolerance* τ_k, defined as the q-quantile (q=0.75) of the Gram distances from its cluster members to P_k. The tolerance captures how much the demonstrations vary for that motion, and it is far from uniform. On LIBERO-90, small-amplitude movement prototypes have tolerances of $1.4\times$ their norm, whereas large-amplitude ones have $0.35\times$ (Fig. 3). Small, precise motions such as aligning a gripper or correcting a grasp vary more relative to their size than long transports across the workspace do. An *amplitude bound*, proportional to each element's norm, therefore gives these motions the least room to correct, while our *tolerance bound*, set by τ_k, gives them the most (§4.3). We choose K by the library reference rather than by reconstruction (§6.1).

3.5 Wrapping a policy

MPQ attaches to any policy with a continuous chunk head. We write this policy as π and treat it as a frozen black box. Its predicted chunk is fit to control points C_π and gripper control points c^{g}_π. For each predicted chunk, the layer *selects* the nearest movement prototype P_z under Eq. 3, *places* it in the frame of the prediction as $R^{-1}P_z$ with $R = \hat{R}_{C_\pi}$, and *corrects* it toward the prediction by an offset clipped to the movement prototype's tolerance τ_z scaled by a budget β. With $e = C_\pi - R^{-1}P_z$ denoting the gap between the prediction and the placed movement prototype, the correction and the executed chunk are

$$\Delta = e \cdot \min\Big(1,\ \frac{\beta\,\tau_z}{\|e\|_M}\Big) \qquad \text{(clipped correction)}, \tag{4}$$

$$C^{\mathrm{exec}} = R^{-1}P_z + \Delta \qquad \text{(executed chunk)}. \tag{5}$$

The executed chunk therefore satisfies the *deviation bound* $\|C^{\mathrm{exec}} - R^{-1}P_z\|_M \le \beta\,\tau_z$. The amplitude bound of §3.4 instead caps $\|\Delta\|_M \le \beta\|P_z\|_M$. At β=0 the policy only chooses which element runs, and as $\beta \to \infty$ the layer reduces to the spline fit of the prediction. For β>0 the gripper passes through unchanged ($c^{\mathrm{g,exec}} = c^{\mathrm{g}}_\pi$), so the policy decides when the gripper closes and the library where the hand goes. Nothing on this path is learned.

The hybrid L_{idx} of Eq. 1 replaces z by another model's choice z_L and keeps the reference's placement and correction,

$$C^{\mathrm{exec}}_{L_{\mathrm{idx}}} \;=\; R_\pi^{-1}\,P_{z_L} + \Delta_\pi(z_L). \tag{6}$$

This isolates the choice from placement and correction.

4 Validating the execution layer

The library reference measures a vocabulary only if the execution layer costs a succeeding policy nothing. We test it across architectures, benchmarks, observation shift and a real robot.

Table 1: MPQ on six architectures. Success (%) bare and through the in-suite library (K=256, β=0.5). H_π: chunk length. Mean ± sd over three seeds, Δ paired per seed. †: released weights, one seed, Δ paired per task with 95% bootstrap CI. Pooled: 95% t-interval over cells (Appendix D.1).

policy	H_π	bare	+MPQ	Δ
LIBERO-90				
ACT	20	78.8 ± 2.3	80.2 ± 1.0	+1.4 ± 2.1
ACT, language-conditioned	20	72.1 ± 0.8	71.6 ± 0.8	−0.6 ± 0.1
Diffusion Policy	16	91.7 ± 0.6	92.9 ± 1.2	+1.2 ± 1.8
QueST	32	85.0 ± 0.7	82.9 ± 2.1	−2.1 ± 1.5
VQ-BeT	5	69.6 ± 6.1	68.3 ± 5.1	−1.3 ± 1.2
Flow-matching DiT	16	81.9 ± 1.5	83.4 ± 1.4	+1.5 ± 1.6
LIBERO-10				
OpenVLA-OFT (7B)†	8	86.8	86.4	−0.4 [−3.6, +2.6]
ACT	20	64.7 ± 3.4	70.3 ± 1.9	+5.6 ± 2.9
Diffusion Policy	16	76.2 ± 2.6	79.5 ± 2.5	+3.3 ± 2.0
Flow-matching DiT	16	72.7 ± 5.0	70.6 ± 4.4	−2.1 ± 3.1
pooled, 13 cells incl. RoboCasa (Table 5)				+0.6 [−0.8, +1.9]

4.1 Setup

Our main benchmark is LIBERO-90 (Liu et al., 2023), whose 90 tasks we evaluate at 10 episodes each from fixed initial states, with a cap of 600 steps per episode. LIBERO-10 and RoboCasa (Nasiriany et al., 2024) are evaluated at 50 episodes per task (Appendix A.3). Unless stated otherwise, the library has K=256 movement prototypes of H=20 steps, with w_g=5 and β=0.5. These values were selected once for ACT on a disjoint split of LIBERO-90 initial states (Appendix A.4) and are used unchanged for every policy, every suite and the real robot. An *in-suite* library is built from the benchmark's own demonstrations, and a *bare* policy runs without the layer at the same replanning interval. Chunks longer than the library horizon are passed to the layer as their first H steps, which contain the executed prefix (Appendix D.3). Confidence intervals are 95% bootstrap intervals over tasks, and three-seed results are paired per seed. Because repeated evaluation of one checkpoint varies by up to 3.5 points (Appendix A.4), we do not interpret unpaired differences below 3 points.

4.2 The layer costs the reference nothing

Table 1 evaluates ACT (Zhao et al., 2023), Diffusion Policy (Chi et al., 2023), a flow-matching DiT (Appendix A.2), QueST (Mete et al., 2024), VQ-BeT (Lee et al., 2024), and the 7B OpenVLA-OFT (Kim et al., 2025) with its released LIBERO-10 weights. QueST and VQ-BeT are themselves discrete skill policies, so for them the layer quantizes the output of their decoders.

Cost of the layer. No policy loses more than evaluation noise. Since the spline fit alone costs nothing (Appendix D.2), any cost comes from quantization, even though the bound clips 67 to 93% of chunks. The largest losses are 1 to 2 points for QueST and VQ-BeT on LIBERO-90 and 2.1 ± 3.1 for the flow DiT on LIBERO-10. Pooled over 13 cells with RoboCasa (Table 5), success changes by $+0.6$ points (95% CI $[-0.8, +1.9]$). Two one-sided tests show equivalence within ± 3 points, the noise floor of §4.1 ($p<0.001$, Appendix D.1). The layer also preserves smoothness (Appendix D.6).

Removing movement prototypes. Removing the most-used movement prototype or five heavily used ones leaves wrapped ACT within noise (seed 0, Appendix D.4). Neighboring movement prototypes substitute for each other within the correction, so the library reference measures the whole library.

Imposing a constraint. Removing movement prototypes can also impose a new constraint on a trained policy without retraining it. As an example, we cap the commanded speed, the largest translation a chunk commands per control step, at the 75th percentile of the library and delete the 64 movement prototypes above it. Bare ACT exceeds the cap in 28.4% of its chunks. With the edited

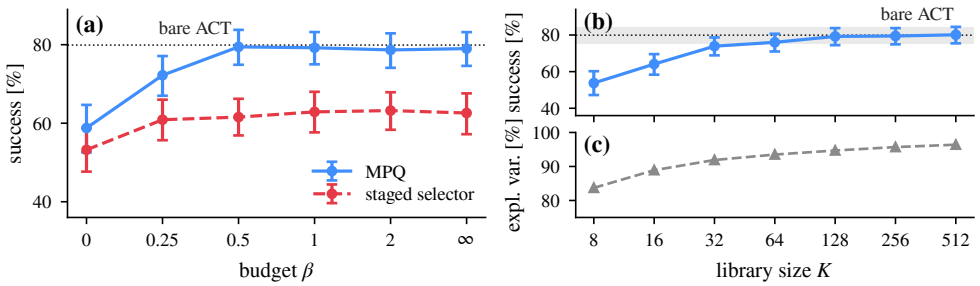


Figure 4: Wrapped ACT on LIBERO-90 (seed 0). Error bars: 95% bootstrap CIs over tasks. **(a)** Success versus the budget β for MPQ and the staged selector (§5.3), both with the K=256 library. Dotted: bare ACT. **(b)** Success versus library size K at β=0.5. Band: bare ACT with its 95% CI. **(c)** Offline explained variance of the same libraries.

library, wrapped ACT exceeds it in 11.9% at unchanged success (79.3 ± 1.2 against 79.0 ± 0.6). Halving the bound limits how much speed the correction can add and lowers violations to 4.5%, at a cost of 10 points (Appendix G.4).

4.3 The deviation bound

Fig. 4a varies β for ACT (seed 0, bare 79.9). Success rises from 58.8 at β=0, where the policy only selects the segment, to 79.4 at β=0.5, and stays flat for larger budgets. Even at β=0.5, the tightest value on this plateau, the bound clips 88% of chunks. The median displacement from the placed movement prototype is then 7 cm, against 39 cm commanded per one-second chunk (Appendix D.5).

The form of the bound matters as well. The relative spread of a movement prototype falls with its amplitude, with a correlation of -0.81 (Fig. 3), so the tolerance bound gives small motions more room than an amplitude bound does. Compared with the amplitude bound and a gripper scaled by the clip, the tolerance bound with the gripper passed through raises the paired difference to the bare policy on each of the nine three-seed LIBERO rows of Table 1, by 3.0 points on average (Appendix B).

4.4 Execution on a real robot

We run a fine-tuned $\pi_{0.5}$ (Physical Intelligence et al., 2025) on a Franka Panda in the DROID configuration (Khazatsky et al., 2024) (Appendix C). The task is to stack one of two cylinders on whichever of four colored cubes the instruction names. The library is built from the 400 fine-tuning demonstrations with the defaults of §4.1, and the bound is the one selected in simulation. Since the policy commands absolute poses, the layer converts them to deltas and back. The bare policy succeeds in 15 of 15 attempts and the wrapped policy in 13 of 15 (Fisher exact test, p=0.48). Both failures are missed grasps in which the gripper stops above the cylinder. With fifteen attempts per arm, this comparison can resolve only large differences. Offline, on held-out demonstrations of the same task, the tolerance bound executes 81% of a 1 cm lateral correction before the release, compared with 63% for the amplitude bound (Appendix C).

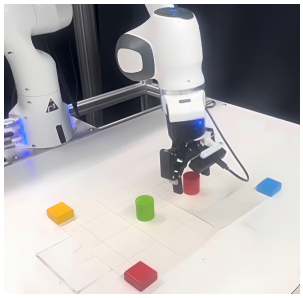


Figure 5: Real-robot task. Stacking a cylinder on the instructed cube.

4.5 Robustness under observation shift

LIBERO-Plus (Fei et al., 2026) perturbs the LIBERO-10 tasks along six axes, namely viewpoint, initial state, noise, layout, texture and lighting. We evaluate the three LIBERO-10 policies, which were trained without augmentation, on 100 paired variants per axis.

Under shift, the layer costs no more than in distribution (Table 2). On no single axis does any policy show a paired loss whose interval excludes zero (Appendix D.7). This holds although shift

Table 2: Observation shift on LIBERO-Plus, LIBERO-10 policies (K=256 in-suite library, β=0.5). In distribution: ten unperturbed tasks, 10 initial states each (Table 1: 50). Shifted: 6 axes $\times$ 100 variants. Mean $\pm$ sd over three seeds, Δ paired per seed. Per axis in Appendix D.7.

	in distribution			six shifted axes		
policy	bare	+MPQ	Δ	bare	+MPQ	Δ
Diffusion Policy	85.8 $\pm$ 1.0	84.3 $\pm$ 1.5	-1.5 ± 2.3	36.6 $\pm$ 1.4	37.9 $\pm$ 1.5	$+1.3 \pm 2.4$
ACT	67.7 $\pm$ 9.0	77.0 $\pm$ 6.6	$+9.3 \pm 6.8$	29.5 $\pm$ 1.8	33.5 $\pm$ 1.6	$+4.0 \pm 2.1$
Flow-matching DiT	81.0 $\pm$ 3.6	80.7 $\pm$ 3.8	-0.3 ± 3.2	33.7 $\pm$ 2.1	33.1 $\pm$ 4.4	-0.6 ± 3.4

Table 3: Decomposition of the selectors' gaps (LIBERO-90, K=256, β=0.5). Heads: how the selector generates placement and correction. SR(π) and quantization as in Table 1. Mean $\pm$ sd over three seeds.

					choice first		correction first	
reference	heads	SR(π)	SR(L)	quant.	selection	execution	execution	selection
ACT	regression	78.8	67.1 $\pm$ 3.5	+1.4	-0.9 ± 1.2	$\mathbf{-12.1} \pm 1.6$	$\mathbf{-12.4} \pm 2.1$	-0.6 ± 2.1
flow DiT	flow	81.9	83.0 $\pm$ 0.0	+1.5	-0.4 ± 1.0	$\mathbf{+0.0} \pm 2.0$	$\mathbf{-3.1} \pm 1.1$	$+2.7 \pm 1.2$
Diff. Policy	diffusion	91.7	82.0 $\pm$ 2.0	+1.2	-0.5 ± 1.1	$\mathbf{-10.3} \pm 1.9$	$\mathbf{-9.0} \pm 1.5$	-1.8 ± 2.5

moves bare ACT's chunks farther from the demonstrations on every axis, even in successful episodes (Appendix D.5). The layer keeps each chunk within the bound of a demonstrated one.

5 Decomposing a discrete selector's gap

Since the execution layer costs a succeeding policy nothing (§4), the gap between a continuous reference and a discrete selector decomposes into quantization, selection and execution as in Eq. 1. We decompose selectors on three references, ACT, the flow DiT and Diffusion Policy.

5.1 Selectors

Every selector L replaces a policy's continuous action output by the skill heads of §3.1. The heads are trained on demonstrations labeled with their nearest movement prototype, after the library is built and frozen (Appendix E). Each selector keeps the trained image and state encoder of its reference frozen and trains only a readout and the skill heads (Table 3). Selector and reference thus see the same features, and the decomposition isolates the cost of the discrete output. The heads generate placement and correction as the reference generates actions. ACT's selector regresses them, the flow DiT's samples them by flow matching and Diffusion Policy's by diffusion.

5.2 Selection and execution terms

Table 3 shows that the selectors' gaps arise in executing the chosen element, not in choosing it. Quantization is within noise for all three references with this library, although it grows for smaller ones (§6.1). In the choice-first order, selection changes success by at most 0.9 points. Execution costs the selectors of ACT and Diffusion Policy 12.1 and 10.3 points, nearly their entire gaps, whereas the flow DiT's selector loses nothing in execution and reaches its library reference (83.0 against 83.4). The correction-first order moves at most 3.1 points between the two terms. Appendix E gives bootstrap intervals for all terms.

A small selection term could also mean that the reference's correction compensates for any choice. We

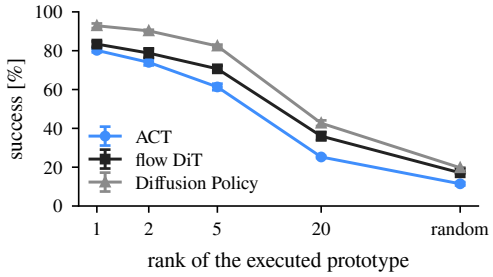


Figure 6: Success when the reference executes its r-th nearest movement prototype (random: expected rank 128). LIBERO-90, K=256, β=0.5, three seeds.

therefore force the layer to execute the r-th nearest
movement prototype to the reference's chunk, or a random one, under the reference's placement and correction (Fig. 6). Success falls steeply with rank. For ACT, the fifth-nearest movement prototype costs 18.9 ± 2.6 points and a random one 68.7 ± 1.7. The flow DiT and Diffusion Policy follow the same curve. Without any correction (β=0, Appendix E), the fifth-nearest movement prototype costs about 25 points, while the choices of the staged ACT selector (§5.3) cost 2.9. The selection term is therefore small because the selectors choose well, not because the correction hides their choices.

5.3 Source of the execution loss

ACT's execution loss follows from how placement and correction are computed. MPQ derives both from the reference's continuous prediction, so the executed chunk follows that prediction as closely as the bound allows, whichever element was chosen. ACT's selector instead regresses yaw and correction from the observation alone, with a single offset that does not depend on the chosen element. The analyses below use the *staged* selector, ACT's network with these heads trained end to end. It loses 14.6 points in execution, against 12.1 for the frozen selector of Table 3, so training the encoder on the selection objective changes the loss little. On held-out chunks, its predicted yaw has a median error of 11° (Appendix E).

Element-conditioned heads. The staged selector's placement and correction heads do not see the chosen movement prototype, which could by itself explain the execution loss. We therefore condition these heads on an embedding of the chosen movement prototype and compare them with heads of equal capacity without it (Appendix E). At execution the heads see the selector's own choice. Conditioning raises success from 65.6 to about 70, whether the heads are trained on the labeled movement prototype or on the selector's own choices. The selector still falls about 10 points short of the library reference, and execution accounts for 9 to 10 of them. The element-independent heads therefore explain only a small part of the execution loss. Diffusion Policy's selector, whose heads see the chosen element, still loses 10.3 points in execution.

Controls. Removing the bound does not help. Without it, the staged selector plateaus at 61 to 63 while MPQ reaches 79 (Fig. 4a), so its loss does not come from the bound. A correction network of Diffusion Policy's own width raises its selector from 82.0 to 85.8, a third of its gap, but execution still costs 6.3 points (Appendix E). Selectors that co-adapt the movement prototypes under a curriculum (Li et al., 2023b; Blessing et al., 2023) or execute a sparse mixture of them, as mixture-of-experts policies do (Zhou et al., 2024; Reuss et al., 2025), score 21 to 33 points below the staged selector (Appendix E). QueST scores 85.0, above bare ACT, but cannot be decomposed, because its codes need a learned decoder (Appendix F).

6 Closed-loop versus offline library evaluation

Libraries are typically validated offline, by the share of demonstration-action variance they reconstruct (Zhou et al., 2025; Lee et al., 2024; Mete et al., 2024). The library reference instead evaluates them in closed loop. We compare the two criteria on library size, library design and transfer.

6.1 Library size and design

Wrapped ACT matches bare ACT from K=128 onward, while reconstruction keeps improving by another 1.7 points over two further doublings (Fig. 4b and 4c). Selecting K by reconstruction would therefore choose a library several times larger than the task requires.

The two criteria also disagree on which design choices matter (Table 4). Dividing out amplitude in addition to yaw costs 5 points in closed loop, yet reconstruction ranks the resulting library highest. Yaw canonicalization itself adds about 3 points of success at K=256 and 10 at K=64 (Appendix G.1). The Gram metric, the clustering algorithm and the projection onto real segments each change success by less than the noise at K=256, and our k-means library comes within 3 points of executing through all 118,525 segments. Leaving out that last row, reconstruction varies by 3.1 points across the libraries of Table 4, while closed-loop success varies by 5.5.

Table 4: Library ablations on LIBERO-90. ACT success and offline explained variance (%) (K=256, w_g=5, β=0.5, seed 0, bare 79.9). Brackets: 95% bootstrap CIs over tasks. Ours: k-means, projected.

library	success	offline
Euclidean	74.6 [69.7, 79.1]	93.6
Gram metric (Eq. 2)	76.7 [71.9, 81.1]	94.0
+ yaw (ours)	79.4 [74.9, 83.8]	95.6
+ amplitude	74.4 [69.2, 79.4]	95.9
EM-GMM, projected	77.6 [72.8, 82.0]	95.1
256 random segments	73.9 [68.9, 78.8]	92.8
all 118,525 segments	82.1 [78.0, 85.9]	100.0

Table 5: Library transfer. Bare success and its change through the target suite's own library and through the LIBERO-90 library (K=256, β=0.5). Mean ± sd over three seeds, Δ paired per seed.

		Δ through library	
policy	bare	own	LIBERO-90
RoboCasa			
ACT	43.6 ± 4.2	−0.9 ± 2.4	−7.9 ± 1.2
Diffusion Policy	43.4 ± 0.3	−0.1 ± 3.9	−4.2 ± 1.8
Flow-matching DiT	39.8 ± 1.3	+1.7 ± 2.3	−6.5 ± 1.5
LIBERO-10			
ACT	64.7 ± 3.4	+5.6 ± 2.9	−0.2 ± 0.6

6.2 Transfer across suites

Transfer separates the two criteria most clearly (Table 5). On RoboCasa, the LIBERO-90 library costs the three policies 4 to 8 points, whereas the in-suite library costs nothing within noise. Reconstruction nevertheless rates them within 3 points (89.0% against 92.0%). On LIBERO-10, the LIBERO-90 library executes ACT at parity with the bare policy (-0.2 ± 0.6) but misses the 5.6-point gain that LIBERO-10's own library provides. Reconstruction again rates the two within 0.1%.

Coverage distinguishes the two cases. The RoboCasa demonstrations effectively occupy only 100 of the 256 movement prototypes in the LIBERO-90 library, against 167 of their own. The LIBERO-10 demonstrations occupy 165 and 186 (Appendix G). The RoboCasa loss is therefore a coverage failure. The LIBERO-10 loss is not, and neither reconstruction nor coverage detects it.

A coverage failure can be repaired without touching the policy. Re-clustering both corpora together at K=256 brings RoboCasa within 1.2 points of its own library (seed 0, $[-5.5, +2.7]$) and leaves LIBERO-90 unchanged (-0.4 ± 0.6, three seeds). Appending RoboCasa movement prototypes to the LIBERO-90 library recovers less (Appendix G).

The two criteria thus answer different questions. Reconstruction asks whether a library can represent the demonstrations, the library reference whether the element executed in a state is the right one.

7 Conclusion

A discrete skill policy combines a vocabulary of motions, a learned choice among its elements and learned heads that execute it. Measured separately in closed loop, their costs differ widely. For selectors on three backbones, the vocabulary costs little, and so does the choice when the reference places and corrects it. Most of the loss lies in the learned placement and correction. Conditioning a regression head on the chosen element recovers little of this loss. When a discrete, editable library is required, wrapping a trained continuous policy is therefore at least as effective as learning a discrete head.

Placing the vocabulary after the policy has further consequences. Every executed chunk stays within a bound of a demonstrated motion, and the library can be built, evaluated and edited while the policy stays frozen. Offline reconstruction cannot guide such edits, because it does not resolve closed-loop differences across library sizes, designs or domains. The library reference resolves them at the cost of one evaluation run per design. For a vision-language-action model that emits an index over such a library, the execution term isolates what the model must supply beyond the index.

Limitations. Our experiments use one arm type and a single real-robot task. The decomposition applies to other selector architectures, but we tested only selectors built on a reference's network or features. Finally, the library reference is relative to a reference policy rather than an optimal one, and the bound constrains the executed motion but not its effect on the scene.

ACKNOWLEDGMENTS

The work was funded by the German Research Foundation (DFG) – 448648559. The authors gratefully acknowledge the support of the Robotics Institute Germany (RIG).

AI USE STATEMENT

In this work, we used generative AI tools (Anthropic's Claude models) to implement methods and to provide feedback on experiments. We have not used generative AI tools to generate synthetic data sets, to develop theoretical models or conceptual frameworks, to formulate mathematical claims, to propose or refine hypotheses, to clean or reformat data sets, or to interpret results. Proofs, translation, and qualitative or thematic data analysis are not applicable to this work.

Additionally, we used generative AI tools to create and debug software code, including figure plotting code, to search for and identify relevant literature, to format references, to draft parts of the research paper and to edit the paper for readability.

We have reviewed all AI-assisted work. In particular, the authors reviewed generated code and unit tests, traced every number reported in the paper to a logged evaluation run, and checked every citation against its published version.

We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

All code for library construction, execution and evaluation is available at https://intuitive-robots.github.io/mpq. Appendix A details the fixed evaluation protocols for all three benchmarks, the seed and noise-floor conventions, and all hyperparameters.

All experiments ran on desktop workstations with consumer GPUs (NVIDIA RTX 3070, 4080, and 5090) and CPUs (Intel Core i7 and AMD Ryzen 9). The project used roughly 500 GPU hours in total, including exploratory runs. Closed-loop evaluation is CPU-bound and uses the GPU only for policy inference.

REFERENCES

Riad Ahmed and Momotaz Begum. TAIL-Safe: Task-agnostic safety monitoring for imitation learning policies. In *Robotics: Science and Systems*, 2026. doi: 10.15607/rss.2026.xxii.207. arXiv:2605.01195.

Anurag Ajay, Aviral Kumar, Pulkit Agrawal, Sergey Levine, and Ofir Nachum. OPAL: Offline primitive discovery for accelerating offline reinforcement learning. In *International Conference on Learning Representations*, 2021. arXiv:2010.13611.

Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *AAAI Conference on Artificial Intelligence*, volume 32, 2018. doi: 10.1609/aaai.v32i1.11797. arXiv:1708.08611.

Aaron D. Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *European Control Conference*. IEEE, 2019. doi: 10.23919/ecc.2019.8796030. arXiv:1903.11199.

Sivakumar Balasubramanian, Alejandro Melendez-Calderon, Agnes Roby-Brami, and Etienne Burdet. On the analysis of movement smoothness. *Journal of NeuroEngineering and Rehabilitation*, 12(1), 2015. doi: 10.1186/s12984-015-0090-9.

Nils Blank, Moritz Reuss, Marcel Rühle, Ömer Erdinç Yağmurlu, Fabian Wenzel, Oier Mees, and Rudolf Lioutikov. Scaling robot policy learning via zero-shot labeling with foundation models. In *Conference on Robot Learning*, 2024. arXiv:2410.17772.

Nils Blank, Paul Mattes, Maximilian Xiling Li, Jakub Suliga, Thomas Roth, Moritz Reuss, Pankhuri Vanjani, and Rudolf Lioutikov. SPARC: Reliable spatial annotations from robot demonstrations at scale. In *Conference on Robot Learning*, 2026. URL https://openreview.net/forum?id=GX5xey9ade. arXiv:2606.13497.

Denis Blessing, Onur Celik, Xiaogang Jia, Moritz Reuss, Maximilian Xiling Li, Rudolf Lioutikov, and Gerhard Neumann. Information maximizing curriculum: A curriculum-based approach for learning versatile skills. In *Advances in Neural Information Processing Systems*, 2023. doi: 10.52202/075280-2244. arXiv:2303.15349.

Remi Cadene, Simon Alibert, Alexander Soare, Quentin Gallouedec, Adil Zouitine, Steven Palma, Pepijn Kooijmans, Michel Aractingi, Mustafa Shukor, Dana Aubakirova, Martino Russi, Francesco Capuano, Caroline Pascal, Jade Choghari, Khalil Meftah, Maxime Ellerbach, Jess Moss, and Thomas Wolf. LeRobot: State-of-the-art machine learning for real-world robotics in PyTorch. https://github.com/huggingface/lerobot, 2024.

Chaofan Chen, Oscar Li, Chaofan Tao, Alina Jade Barnett, Jonathan Su, and Cynthia Rudin. This looks like that: Deep learning for interpretable image recognition. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Robotics: Science and Systems*, 2023.

Maximilian Du, Suraj Nair, Dorsa Sadigh, and Chelsea Finn. Behavior retrieval: Few-shot imitation learning by querying unlabeled datasets. In *Robotics: Science and Systems*, 2023. doi: 10.15607/rss.2023.xix.011.

Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, et al. LIBERO-Plus: A progressive robustness benchmark for visual-language-action models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

Emilio Frazzoli, Munther A. Dahleh, and Eric Feron. Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE Transactions on Robotics*, 21(6), 2005. doi: 10.1109/TRO.2005.852260.

Xiaoshen Han, Haoyu Xiong, Haonan Chen, Chaoqi Liu, Antonio Torralba, Yuke Zhu, and Yilun Du. B-spline policy: Accelerating manipulation policies via B-spline action representations. *arXiv preprint arXiv:2607.09648*, 2026. doi: 10.48550/arXiv.2607.09648.

Xi Huang, Hongyi Zhou, Ge Li, Yucheng Tang, Weiran Liao, Björn Hein, Tamim Asfour, and Rudolf Lioutikov. MoRe-ERL: Learning motion residuals using episodic reinforcement learning. *IEEE Robotics and Automation Letters*, 10(10), 2025. doi: 10.1109/LRA.2025.3607229. arXiv:2508.01409.

Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, 25 (2), 2013. doi: 10.1162/NECO_a_00393.

Tobias Johannink, Shikhar Bahl, Ashvin Nair, Jianlan Luo, Avinash Kumar, Matthias Loskyll, Juan Aparicio Ojea, Eugen Solowjow, and Sergey Levine. Residual reinforcement learning for robot control. In *IEEE International Conference on Robotics and Automation*, 2019. doi: 10.1109/ICRA.2019.8794127.

Xirui Kang, Yanpei Shi, Lucy Liang, Roy Gan, Dongxiu Liu, Pushi Zhang, Danpeng Chen, Xiao Qin, Yinan Zheng, Jinliang Zheng, et al. X-Tokenizer: A multimodal action tokenizer for vision-language-action pretraining. *arXiv preprint arXiv:2606.14752*, 2026. doi: 10.48550/arXiv.2606.14752.

Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, et al. DROID: A large-scale in-the-wild robot manipulation dataset. In *Robotics: Science and Systems*, 2024. arXiv:2403.12945.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *Robotics: Science and Systems*, 2025. doi: 10.15607/rss.2025.xxi.017.

George Konidaris, Scott Kuindersma, Roderic Grupen, and Andrew Barto. Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research*, 31(3), 2012. doi: 10.1177/0278364911428653.

Seungjae Lee, Yibin Wang, Haritheja Etukuru, H. Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. In *International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, 2024.

Xiaohan Lei, Min Wang, Wengang Zhou, Xingyu Lu, and Houqiang Li. Primary-fine decoupling for action generation in robotic imitation. In *International Conference on Learning Representations*, 2026.

Ge Li, Zeqi Jin, Michael Volpp, Fabian Otto, Rudolf Lioutikov, and Gerhard Neumann. ProDMP: A unified perspective on dynamic and probabilistic movement primitives. *IEEE Robotics and Automation Letters*, 8(4), 2023a. doi: 10.1109/LRA.2023.3248443. arXiv:2210.01531.

Maximilian Xiling Li, Onur Celik, Philipp Becker, Denis Blessing, Rudolf Lioutikov, and Gerhard Neumann. Curriculum-based imitation of versatile skills. In *IEEE International Conference on Robotics and Automation*. IEEE, 2023b. doi: 10.1109/ICRA48891.2023.10160543. arXiv:2304.05171.

Maximilian Xiling Li, Korbinian Franz Rudolf, Paul Mattes, Nils Blank, and Rudolf Lioutikov. An overview of prototype formulations for interpretable deep learning. *arXiv preprint arXiv:2410.08925*, 2024. doi: 10.48550/arXiv.2410.08925.

Rudolf Lioutikov, Gerhard Neumann, Guilherme Maeda, and Jan Peters. Learning movement primitive libraries through probabilistic segmentation. *The International Journal of Robotics Research*, 36(8), 2017. doi: 10.1177/0278364917713116.

Rudolf Lioutikov, Guilherme Maeda, Filipe Veiga, Kristian Kersting, and Jan Peters. Inducing probabilistic context-free grammars for the sequencing of movement primitives. In *IEEE International Conference on Robotics and Automation*, 2018. doi: 10.1109/ICRA.2018.8460190.

Rudolf Lioutikov, Guilherme Maeda, Filipe Veiga, Kristian Kersting, and Jan Peters. Learning attribute grammars for movement primitive sequencing. *The International Journal of Robotics Research*, 39(1), 2020. doi: 10.1177/0278364919868279.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, 2023. doi: 10.52202/075280-1939. arXiv:2306.03310.

Chaoqi Liu, Xiaoshen Han, Jiawei Gao, Yue Zhao, Haonan Chen, and Yilun Du. OAT: Ordered action tokenization. In *Robotics: Science and Systems*, 2026. doi: 10.15607/rss.2026.xxii.075. arXiv:2602.04215.

Yuejiang Liu, Jubayer Ibn Hamid, Annie Xie, Yoonho Lee, Maximilian Du, and Chelsea Finn. Bidirectional decoding: Improving action chunking via guided test-time sampling. In *International Conference on Learning Representations*, 2025. arXiv:2408.17355.

Paul Werner Lödige, Maximilian Xiling Li, and Rudolf Lioutikov. Use the force, bot! – force-aware ProDMP with event-based replanning. In *IEEE International Conference on Robotics and Automation*, 2025. doi: 10.1109/ICRA55743.2025.11128718. arXiv:2409.11144.

Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning*, 2021. arXiv:2108.03298.

Marius Memmel, Jacob Berg, Bingqing Chen, Abhishek Gupta, and Jonathan Francis. STRAP: Robot sub-trajectory retrieval for augmented policy learning. In *International Conference on Learning Representations*, 2025. arXiv:2412.15182.

Atharva Mete, Haotian Xue, Albert Wilcox, Yongxin Chen, and Animesh Garg. QueST: Self-supervised skill abstractions for learning continuous control. In *Advances in Neural Information Processing Systems*, 2024. doi: 10.52202/079017-0133. arXiv:2407.15840.

Yao Ming, Panpan Xu, Huamin Qu, and Liu Ren. Interpretable and steerable sequence learning via prototypes. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2019. doi: 10.1145/3292500.3330908. arXiv:1907.09728.

Soroush Nasiriany, Tian Gao, Ajay Mandlekar, and Yuke Zhu. Learning and retrieval from prior data for skill-based imitation learning. In *Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, 2022. arXiv:2210.11435.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. RoboCasa: Large-scale simulation of everyday tasks for generalist robots. In *Robotics: Science and Systems*, 2024. doi: 10.15607/rss.2024.xx.050. arXiv:2406.02523.

Scott Niekum, Sarah Osentoski, George Konidaris, and Andrew G. Barto. Learning and generalization of complex tasks from unstructured demonstrations. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012. doi: 10.1109/IROS.2012.6386006.

Sodtavilan Odonchimed, Tatsuya Matsushima, Simon Holk, Yusuke Iwasawa, and Yutaka Matsuo. Retrieve-augmented generation for speeding up diffusion policy without additional training. *arXiv preprint arXiv:2507.21452*, 2025. doi: 10.48550/arXiv.2507.21452.

Alexandros Paraschos, Christian Daniel, Jan Peters, and Gerhard Neumann. Probabilistic movement primitives. In *Advances in Neural Information Processing Systems*, 2013.

Jyothish Pari, Nur Muhammad Mahi Shafiullah, Sridhar Pandian Arunachalam, and Lerrel Pinto. The surprising effectiveness of representation learning for visual imitation. In *Robotics: Science and Systems*, 2022. doi: 10.15607/rss.2022.xviii.010.

Karl Pertsch, Youngwoon Lee, and Joseph J. Lim. Accelerating reinforcement learning with learned skill priors. In *Conference on Robot Learning*, 2020. arXiv:2010.11944.

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. In *Robotics: Science and Systems*, 2025. doi: 10.15607/rss.2025.xxi.012.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: A vision-language-action model with open-world generalization. In *Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, 2025.

Moritz Reuss, Jyothish Pari, Pulkit Agrawal, and Rudolf Lioutikov. Efficient diffusion transformer policies with mixture of expert denoisers for multitask learning. In *International Conference on Learning Representations*, 2025. arXiv:2412.12953.

Paul Maria Scheikl, Nicolas Schreiber, Christoph Haas, Niklas Freymuth, Gerhard Neumann, Rudolf Lioutikov, and Franziska Mathis-Ullrich. Movement primitive diffusion: Learning gentle robotic manipulation of deformable objects. *IEEE Robotics and Automation Letters*, 9(6), 2024. doi: 10.1109/LRA.2024.3382529. arXiv:2312.10008.

Nur Muhammad Mahi Shafiullah, Zichen Jeff Cui, Ariuntuya Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. In *Advances in Neural Information Processing Systems*, 2022. doi: 10.52202/068431-1668. arXiv:2206.11251.

Tom Silver, Kelsey Allen, Josh Tenenbaum, and Leslie Kaelbling. Residual policy learning. *arXiv preprint arXiv:1812.06298*, 2018.

Martin Stolle and Christopher G. Atkeson. Policies based on trajectory libraries. In *IEEE International Conference on Robotics and Automation*, 2006.

Mengze Tian, Yiming Li, Sichao Liu, Auke Ijspeert, and Sylvain Calinon. Spline policy: A structured representation for robot policies. *arXiv preprint arXiv:2606.07386*, 2026. doi: 10.48550/arXiv.2606.07386.

Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, 2017. arXiv:1711.00937.

Yuntian Wang, Zesheng Jia, Yuhui Duan, Qibing Wang, Yang Liu, Song Wang, Siao Liu, and Jin Wang. ARP: Enhancing quantized skill abstractions via visual alignment and iterative refinement for robotic manipulation. *arXiv preprint arXiv:2606.22480*, 2026. doi: 10.48550/arXiv.2606.22480.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems*, 2023. doi: 10.15607/rss.2023.xix.016. arXiv:2304.13705.

Ruijie Zheng, Ching-An Cheng, Hal Daumé III, Furong Huang, and Andrey Kolobov. PRISE: LLM-style sequence compression for learning temporal action abstractions in control. In *International Conference on Machine Learning*, 2024. arXiv:2402.10450.

Hongyi Zhou, Denis Blessing, Ge Li, Onur Celik, Xiaogang Jia, Gerhard Neumann, and Rudolf Lioutikov. Variational distillation of diffusion policies into mixture of experts. In *Advances in Neural Information Processing Systems*, 2024. doi: 10.52202/079017-0405. arXiv:2406.12538.

Hongyi Zhou, Wei-Ran Liao, Xi Huang, Yucheng Tang, Fabian Otto, Xiaogang Jia, Xinkai Jiang, Simon Hilber, Ge Li, Qian Wang, et al. BEAST: Efficient tokenization of B-splines encoded action sequences for imitation learning. In *Advances in Neural Information Processing Systems*, 2025. doi: 10.52202/085713-5755. arXiv:2506.06072.

A EXPERIMENTAL DETAILS

A.1 CHUNKS, SPLINE FIT AND LIBRARY CONSTRUCTION

Demonstrations are cut into windows of $H{=}20$ actions at stride 5, which gives 118,525 segments on LIBERO-90. Each window is fit to $B{=}10$ B-spline control points with BEAST's knot configuration (Zhou et al., 2025), all channels on the same basis. The fit is near-lossless (residual MSE $1.29{\times}10^{-3}$, per-step action variance 2.01×10^{-1}).

Canonicalization rotates each segment about the gravity axis until its net horizontal displacement points along $+x$. For displacements shorter than 10^{-3} action units, the first principal axis of the per-step horizontal displacements is aligned instead, and if these also vanish the yaw is zero. The gripper channel is never rotated. The same rule canonicalizes the policy's chunks at execution time. k-means runs on Cholesky-whitened coefficients, with the whitened gripper channel scaled by w_g, and each centroid is projected onto its nearest segment. Building a library takes about two minutes on a desktop workstation.

A.2 POLICIES

ACT, Diffusion Policy and the flow-matching policy are implemented in LeRobot (Cadene et al., 2024).

ACT and all selectors share one training recipe. They use an ImageNet-pretrained ResNet-18 with frozen BatchNorm and an ACT-style transformer (Zhao et al., 2023), and they train for 50k steps at batch size 128 with AdamW (learning rate and weight decay 10^{-4}) and a cosine schedule. Task-id conditioning indexes an embedding table, and language conditioning projects frozen CLIP ViT-B/16 sentence embeddings of the instruction.

Diffusion Policy (Chi et al., 2023) uses a ResNet-18 per camera and a 1D-UNet denoiser over 16-step chunks with one observation step, with the task one-hot appended to proprioception. It is trained for 100k steps at batch size 64 (AdamW, lr 10^{-4}, 500-step warmup, cosine decay) with 112×112 crops and DDPM with 100 noise steps. We evaluate it with the EMA weights (decay 0.9995) and 10 DDIM steps. Proprioception enters unnormalized, which affects absolute success but not the paired differences.

The **flow-matching policy** is an 8-layer diffusion transformer (hidden size 768, 12 heads, conditioning MLP of width 512) over an ImageNet-pretrained ResNet-18. It predicts 16-step chunks and is trained for 100k steps at batch size 64 without crops.

QueST and VQ-BeT are trained through their official pipelines at the default configuration. For VQ-BeT, this means 5-step chunks, with the codebook and the head trained jointly.

OpenVLA-OFT is the released LIBERO-10 checkpoint, served through the authors' inference server with their settings (L1-regression head, proprioception, center crop, agentview and wrist images, 8-step chunks, open loop), evaluated at a single seed. It scores 86.8 against the reported 94.5. Our protocol differs in settling steps (5 vs. 10) and step cap (600 vs. 520), and the bare and wrapped cells share it.

A.3 BENCHMARKS AND EVALUATION PROTOCOL

LIBERO-90 has 90 tasks, each evaluated for 10 episodes from the fixed initial states (900 rollouts per cell) with a 600-step cap. **LIBERO-10** (LIBERO-Long) has 10 long-horizon tasks, evaluated at 50 initial states each (500 rollouts). For **RoboCasa** we use the 17 arm-driven atomic tasks (Nasiriany et al., 2024) with the mobile base idle, at 50 episodes per task (850 rollouts). The RoboCasa policies and libraries use all 500 machine-generated demonstrations per task. Both simulators drive a Franka Panda through an operational-space controller, so transfer between them crosses domains but not embodiments.

Replanning and latency. ACT and OpenVLA-OFT execute their full chunks of 20 and 8 steps. The Diffusion Policy and the flow-matching policy execute 8 of 16 steps, QueST 8 of 32 and VQ-BeT 2 of 5, both bare and wrapped. The layer quantizes the full predicted chunk before the executed prefix

runs. This adds 0.3 ms per chunk on one CPU thread. The overhead is flat in K from 64 to 512, is dominated by the spline fit and amounts to about 0.03% of a chunk's execution time at 20 Hz.

A.4 Seeds, noise floor and selection split

Three evaluations of wrapped ACT (seed 0) through the default library gave 80.0, 79.4 and 79.0. Two evaluations of the Euclidean cell of Table 4 gave 71.1 and 74.6, the largest spread we observed. We therefore treat differences under 3 points as noise.

The design sweeps (Tables 4 and 16, the β and K sweeps, and §5.3) evaluate seed-0 checkpoints against the seed-0 reference of 79.9. The headline results (Tables 1, 3 and 5) average three seeds, with the library reference and the selectors evaluated per seed. Table 6 gives the per-seed values behind Table 1.

Selection split. Every LIBERO-90 cell uses the first 10 of the 50 initial states per task, and hyperparameters are selected on states 10 to 19. On these states, wrapped ACT (seed 0) scores 72.4 / 80.4 / 79.4 for $\beta \in \{0.25, 0.5, 1\}$ at q=0.75 and 74.6 / 80.3 / 76.6 at q=0.9 (task-level intervals about $\pm$4.5), so β=0.5 is best at either quantile. K and w_g were selected on the same split under the amplitude bound, where all cells with $K \geq 256$ lie within 3 points, and kept for the tolerance bound.

B Amplitude and tolerance bounds per policy

Table 7 compares the layer of Table 1 with an amplitude-bound configuration on every three-seed LIBERO row. That configuration caps the correction at $0.25\,\|P_z\|_M$ and scales the gripper channel with the same clip factor, whereas the layer of Table 1 uses the tolerance bound and passes the gripper through (§3.5). At about equal tightness on ACT (seed 0), the tolerance bound at q=0.9 clips 74% of chunks and scores 80.6, the amplitude bound clips 78% and scores 78.0, and a constant cap (95% clipped) scores 78.1. These single cells differ by less than the noise floor, but the direction holds across the policies of Table 7.

C Real-robot setup

The robot is a Franka Panda with a Robotiq 2F-85 gripper and a wrist-mounted ZED Mini camera, in the DROID configuration (Khazatsky et al., 2024). We fine-tune $\pi_{0.5}$ on 400 teleoperated demonstrations recorded at 20 Hz. They cover two cylinders, each stacked on one of four colored cubes, with 50 demonstrations per pair. The library is built from the same demonstrations with the defaults of §4.1 (K=256, H=20, w_g=5, β=0.5, q=0.75).

The policy commands absolute end-effector poses. The layer therefore converts each predicted chunk to per-step deltas in the base frame, starting from the last commanded pose, quantizes it and integrates the result back from the same pose. Consecutive commanded targets define the deltas both in the library and at execution. Position and rotation deltas are rescaled to the action statistics of LIBERO-90, where the defaults were selected. The bare and the wrapped policy share the checkpoint, the replanning interval and the scene configurations. An attempt counts as a success if the cylinder rests on the named cube when the arm retracts.

Offline replay. We replay held-out demonstrations through the layer in five folds over episodes, stratified by task, with each library built from the other four folds. Every step starts a 20-step chunk, converted and quantized as on the robot. Unperturbed, the layer moves the release position by a median of 0.5 mm under the tolerance bound and 1.5 mm under the amplitude bound, while the demonstrators' own placements spread by 5 to 8 mm. To test a correction that the demonstrations do not contain, we add to each chunk that contains the release a horizontal offset perpendicular to the transport direction, ramped in over 10 steps. At the release, the tolerance bound executes 82 / 81 / 71 / 49% of offsets of 0.5 / 1 / 2 / 4 cm, and the amplitude bound 62 / 63 / 54 / 37%.

Table 6: Per-seed values behind Table 1, including its RoboCasa rows (K=256, β=0.5, q=0.75). Brackets: 95% bootstrap CIs over tasks. *: ablation of Appendix D.3, not part of Table 1.

policy	seed	bare	+MPQ
LIBERO-90			
ACT, task-id	0	79.9 [75.2, 84.2]	79.4 [74.9, 83.8]
	1	80.3 [75.9, 84.4]	81.3 [77.3, 85.0]
	2	76.1 [70.9, 81.1]	79.8 [75.1, 84.1]
ACT, language	0	71.3 [66.1, 76.3]	70.7 [64.8, 76.2]
	1	72.9 [67.7, 77.9]	72.3 [66.8, 77.7]
	2	72.2 [66.8, 77.2]	71.7 [66.0, 77.1]
Diffusion Policy	0	92.2 [90.1, 94.1]	92.1 [89.7, 94.3]
	1	91.8 [89.4, 93.9]	92.2 [89.7, 94.6]
	2	91.0 [88.8, 93.0]	94.2 [92.0, 96.2]
Flow-matching DiT	0	83.3 [79.3, 87.0]	84.8 [80.8, 88.4]
	1	80.3 [76.2, 84.2]	83.4 [79.3, 87.1]
	2	82.0 [77.9, 85.8]	82.0 [77.7, 85.9]
QueST	0	84.8 [80.2, 89.0]	81.2 [76.0, 86.0]
	1	85.8 [82.2, 89.0]	85.2 [81.4, 88.8]
	2	84.4 [80.7, 87.9]	82.3 [77.8, 86.6]
QueST, full 32-step chunks*	0	83.0 [78.4, 87.1]	69.8 [63.4, 75.7]
	1	87.2 [84.3, 89.9]	72.9 [66.9, 78.6]
	2	85.6 [81.8, 89.1]	72.6 [66.4, 78.3]
VQ-BeT	0	63.1 [56.6, 69.6]	63.1 [56.9, 69.2]
	1	75.1 [70.6, 79.4]	73.3 [68.7, 77.8]
	2	70.6 [65.0, 76.0]	68.3 [62.2, 74.2]
LIBERO-10			
OpenVLA-OFT	0	86.8 [79.4, 93.2]	86.4 [79.4, 92.6]
ACT, task-id	0	63.6 [54.2, 74.0]	71.8 [63.4, 80.2]
	1	68.6 [60.4, 78.6]	71.0 [62.8, 79.2]
	2	62.0 [51.0, 74.0]	68.2 [60.6, 76.0]
Diffusion Policy	0	78.8 [68.2, 88.6]	82.2 [73.2, 90.4]
	1	73.6 [62.4, 84.8]	78.8 [69.0, 88.2]
	2	76.2 [66.6, 85.0]	77.4 [67.4, 87.2]
Flow-matching DiT	0	74.2 [61.8, 86.0]	68.6 [55.0, 82.2]
	1	67.2 [53.8, 79.8]	67.6 [55.2, 80.2]
	2	76.8 [66.0, 87.2]	75.6 [65.8, 84.8]
RoboCasa (in-suite library, K=256)			
ACT	0	48.1 [37.4, 59.4]	44.7 [34.5, 55.5]
	1	42.9 [31.9, 54.7]	42.4 [31.9, 53.5]
	2	39.8 [28.7, 51.6]	41.2 [30.0, 53.3]
Diffusion Policy	0	43.5 [31.3, 56.4]	42.1 [31.5, 53.6]
	1	43.1 [31.6, 55.1]	39.9 [27.9, 52.9]
	2	43.6 [32.7, 55.4]	47.9 [36.1, 60.1]
Flow-matching DiT	0	40.5 [29.4, 52.2]	40.1 [29.5, 51.4]
	1	38.4 [28.2, 49.4]	42.6 [31.8, 53.8]
	2	40.6 [30.1, 51.5]	41.9 [31.2, 52.7]

D VALIDATING THE LAYER

D.1 POOLED EFFECT AND EQUIVALENCE

Each of the 13 in-suite cells of Table 1 (six LIBERO-90, four LIBERO-10 and three RoboCasa rows) contributes its seed-averaged paired difference. With cells as the unit, evaluation noise, seed variation and differences between policies all enter the interval. The mean is $+0.55$ points (sd 2.21), with a 95% t-interval of $[-0.78, +1.89]$ and a 90% interval of $[-0.54, +1.64]$. Two one-sided tests at α=0.05 establish equivalence at the noise-floor margin of ± 3 points (p<0.001) and at any margin of 1.64 points or more (p=0.018 at ± 2). LIBERO-90 alone gives $+0.01$ $[-1.63, +1.66]$. Adding QueST with its full 32-step chunks (Appendix D.3) gives -0.45 $[-2.94, +2.04]$, still equivalent at ± 3 (p=0.023).

Table 7: Amplitude versus tolerance bound. Paired difference to the bare policy under the amplitude bound (gripper scaled with the clip) and under the tolerance bound of Table 1 (β=0.5, q=0.75, as in Table 1). Mean $\pm$ sd over three seeds, Δ paired per seed.

policy	amplitude	tolerance	change
LIBERO-90			
ACT	-2.0 ± 2.3	$+1.4 \pm 2.1$	$+3.4$
ACT, language-conditioned	-1.9 ± 0.6	-0.6 ± 0.1	$+1.3$
Diffusion Policy	-1.4 ± 1.5	$+1.2 \pm 1.8$	$+2.6$
QueST	-5.2 ± 0.6	-2.1 ± 1.5	$+3.1$
VQ-BeT	-1.8 ± 0.8	-1.3 ± 1.2	$+0.5$
Flow-matching DiT	-2.2 ± 1.7	$+1.5 \pm 1.6$	$+3.7$
LIBERO-10			
ACT	-1.1 ± 2.4	$+5.6 \pm 2.9$	$+6.7$
Diffusion Policy	-0.2 ± 4.7	$+3.3 \pm 2.0$	$+3.5$
Flow-matching DiT	-4.5 ± 5.1	-2.1 ± 3.1	$+2.4$
mean			$+3.0$

D.2 Fit-only control

Fitting each predicted chunk to control points and decoding it again, without a library, separates the representation from quantization. For ACT this is the identity, since its head predicts control points, and chunks shorter than B are interpolated exactly. Where the fit does change the chunk, it costs nothing. The Diffusion Policy moves from 92.2 [90.1, 94.1] to 91.9 [89.6, 94.0], and QueST from 83.0 to 81.6 [76.4, 86.1].

D.3 Chunk length

The spline basis is defined on normalized time. A chunk of H_π steps is therefore fit on a uniform grid over the whole support, compared with the library in the Gram metric of H=20, and decoded back at H_π steps. In effect, it is matched to movement prototypes rescaled in time by H_π/H. Chunks shorter than B, which are the 8-step chunks of OpenVLA-OFT and the 5-step chunks of VQ-BeT, are fit by minimum-norm exact interpolation.

Rescaling alone costs little. OpenVLA-OFT (factor 0.4, all 8 steps executed) changes by -0.4 and VQ-BeT (factor 0.25, 2 of 5 executed) by -1.3 ± 1.2, while the Diffusion Policy and the flow-matching policy (factor 0.8, 8 of 16 executed) are at parity. QueST's native 32-step chunks, of which 8 execute, cost 13.5 ± 0.7 points (19.8 ± 1.9 under the amplitude bound). A library built from 32-step windows (stride 8, K=256, tolerances and Gram metric at H=32) removes the rescaling, yet still costs 8.1 ± 2.0. Passing only the first 20 steps, which contain the 8 executed ones, reduces the cost to 2.1 ± 1.5. Rescaling thus accounts for about 5 of the 13.5 points, and the rest arises from matching the element and its correction to a chunk whose last 24 steps never execute.

D.4 Element removal

We define a movement prototype's failure enrichment as its usage share in failed episodes divided by its share in successful ones (wrapped ACT, LIBERO-90, seed 0). Movement prototype 149, a near-idle motion and the most used one (5.0% of chunks), is enriched $10\times$ under the default configuration (K=256, β=0.5, q=0.75). When a movement prototype is removed, its chunks are redirected to the nearest remaining one. Despite its enrichment, removing 149 leaves success at 79.2 [74.6, 83.7] against 79.4. Removing five heavily used movement prototypes (149, 126, 10, 33 and 94, together 12% of chunks) gives 79.3 [74.6, 83.8]. Without correction (β=0), the reduced libraries score 60.7 and 57.6 against 58.8.

D.5 The bound in physical units

LIBERO's operational-space controller maps a unit action to 5 cm of commanded end-effector displacement per step, or 0.5 rad in orientation. A deviation of Gram norm c therefore changes a

chunk's commanded net displacement by at most $0.05\,\mathrm{m} \cdot \sqrt{H}\,c$, which a constant offset attains. Under the default configuration, the caps allow at most 14 cm per 20-step chunk as the median over movement prototypes, and 20 cm at the 90th percentile.

The logged rollouts of wrapped ACT (seed 0, 900 episodes, 10,724 full chunks) give the realized values. The log does not record the placement R, so we compare under the yaw that minimizes the Gram distance, which lower-bounds the distance under R. Every chunk stays within its cap under this yaw (largest ratio 1.0004, from the decoder's clamp to $[-1, 1]$). A chunk's commanded net displacement differs from that of its movement prototype by a median of 7.1 cm (90th percentile 12.5 cm), against 39 cm commanded per chunk. Since the arm achieves about 20% of each commanded displacement, these correspond to about 1.4 and 8 cm of executed motion.

Distance to the demonstrations. We compare each executed chunk with the nearest of the 118,525 canonicalized LIBERO-90 demonstration segments (bare ACT, seed 0, logged re-evaluation at 76.8%). The chunks of bare ACT lie at a median Gram distance of 0.50 (99th percentile 1.18, maximum 1.74), and those of wrapped ACT at 0.41 (0.75, 1.06). Chunks from failed episodes lie farther than those from successful ones (median 0.57 against 0.43). Under LIBERO-Plus (Table 8), bare ACT's chunks move away from the demonstrations on every axis, in successful episodes as well, and the share that the layer would clip rises from 86% to between 90 and 92%.

Table 8: Distance of bare ACT's chunks to the demonstrations under LIBERO-Plus (Table 2). Distance: Gram distance of each executed 20-step chunk to the nearest LIBERO-10 demonstration segment. Clipped: share of chunks farther from their nearest movement prototype than its cap, which the layer would clip (K=256 in-suite library, β=0.5, q=0.75). Mean over three seeds.

		distance to nearest demonstration			
axis	chunks	median	99th pct.	median, successes	clipped (%)
in distribution	1870	0.55	1.50	0.47	85.6 ± 2.3
camera viewpoint	2891	0.71	1.62	0.60	91.3 ± 1.9
robot initial state	2304	0.67	1.60	0.51	90.4 ± 0.9
sensor noise	2679	0.73	1.67	0.53	92.1 ± 1.7
object layout	2240	0.63	1.59	0.51	90.0 ± 0.5
background texture	2545	0.68	1.68	0.54	90.6 ± 1.2
lighting	2453	0.67	1.56	0.51	89.9 ± 1.2

D.6 Trajectory quality

The LIBERO-90 rollouts of ACT, the Diffusion Policy and the flow-matching policy under the default configuration log the executed action and the end-effector pose at every step. For each episode we compute the spectral arc length (SPARC) of the end-effector speed profile (Balasubramanian et al., 2015) and the mean action discontinuity $\|a_t - a_{t-1}\|$, both at replanning seams and inside chunks. Failed episodes are rougher than successful ones on every measure, so the paired statistic uses only episodes in which both the bare and the wrapped policy succeed (Table 9). No wrapped policy differs from its bare counterpart in SPARC. Jumps inside chunks shrink for all three policies. Jumps at seams shrink for the Diffusion Policy and the flow-matching policy and grow by 0.007 for ACT.

D.7 Observation shift per axis

Table 10 gives the per-axis success behind Table 2 at seed 0.

E Learned selectors

Architecture and training. The ACT-based selectors are the ACT network of Appendix A.2 with the coefficient head replaced by a K-way choice head, a yaw head and a residual head, all of which read the decoder's query token. The staged selector is trained in a single run against the frozen library. Each demonstrated chunk is fit, yaw-canonicalized and assigned to its nearest movement prototype under Eq. 3 (w_g=5), which gives a label z^* and a yaw θ^*. The choice head is trained

Table 9: Trajectory quality on LIBERO-90 (β=0.5, q=0.75, seed 0, 900 episodes per arm). Rows: mean $\pm$ sd over successful episodes, and over 10 demonstrations per task. Δ: wrapped minus bare, paired per task and initial state over the episodes in which both arms succeed, with 95% bootstrap CIs. Bold: CI excludes zero. The Diffusion Policy's fit-only arm differs from bare on no measure.

	SPARC	seam jump	within-chunk jump
demonstrations	-2.38 $\pm$ 0.39	0.065 $\pm$ 0.025	0.064 $\pm$ 0.016
ACT, bare	-2.41 $\pm$ 0.55	0.212 $\pm$ 0.092	0.052 $\pm$ 0.015
ACT +MPQ	-2.45 $\pm$ 0.61	0.220 $\pm$ 0.091	0.046 $\pm$ 0.012
Δ (n=617)	-0.01 [-0.05, $+0.02$]	$\mathbf{+0.007}$ [$+0.000$, $+0.014$]	$\mathbf{-0.005}$ [-0.006, -0.005]
Diffusion Policy, bare	-2.44 $\pm$ 0.53	0.210 $\pm$ 0.044	0.054 $\pm$ 0.014
Diffusion Policy, fit only	-2.44 $\pm$ 0.50	0.211 $\pm$ 0.045	0.054 $\pm$ 0.014
Diffusion Policy +MPQ	-2.41 $\pm$ 0.52	0.201 $\pm$ 0.041	0.049 $\pm$ 0.012
Δ (n=786)	$+0.03$ [-0.01, $+0.07$]	$\mathbf{-0.009}$ [-0.012, -0.005]	$\mathbf{-0.005}$ [-0.005, -0.005]
Flow-matching DiT, bare	-2.44 $\pm$ 0.63	0.183 $\pm$ 0.054	0.058 $\pm$ 0.015
Flow-matching DiT +MPQ	-2.44 $\pm$ 0.67	0.177 $\pm$ 0.041	0.052 $\pm$ 0.013
Δ (n=688)	$+0.02$ [-0.03, $+0.06$]	$\mathbf{-0.006}$ [-0.009, -0.002]	$\mathbf{-0.006}$ [-0.006, -0.005]

Table 10: LIBERO-Plus success (%) per axis (seed 0, K=256 in-suite library, β=0.5, q=0.75, 600-step cap). Shifted axes: the fixed 100-variant subset of each axis. In distribution: the ten unperturbed tasks at 10 initial states. Δ paired per variant.

	Diffusion Policy			ACT			Flow-matching DiT		
axis	bare	+MPQ	Δ	bare	+MPQ	Δ	bare	+MPQ	Δ
in distribution	86	86	+0	67	71	+4	82	78	-4
camera viewpoint	30	31	+1	8	12	+4	6	5	-1
robot initial state	47	44	-3	44	43	-1	46	38	-8
sensor noise	25	29	+4	20	28	+8	19	16	-3
object layout	65	64	-1	44	50	+6	60	50	-10
background texture	16	13	-3	19	28	+9	26	21	-5
lighting	36	36	+0	32	44	+12	44	44	+0
all shifted axes (600)	37	36	-0	28	34	+6	34	29	-5

by cross-entropy on z^* with hard labels. The yaw head predicts $(\sin\theta, \cos\theta)$, regressed onto θ^*, together with a log-amplitude whose target is zero. The residual head predicts a single offset for all control points, gripper included, regressed onto the chunk minus P_{z^*} placed at θ^*. The bound is not applied in training, so β enters only at execution. The three losses are weighted 1, 1 and 10. All parameters train jointly from ImageNet initialization with the ACT recipe rather than from the reference's weights.

The selectors on the flow DiT and Diffusion Policy read the vector that the frozen reference computes from the observation to condition its denoiser, namely the 512-dimensional input to the DiT's adaptive layer norms and the 227-dimensional global conditioning of Diffusion Policy's U-Net. Both combine ResNet-18 keypoints from the two cameras, the robot state and the task. A linear layer and two pre-norm residual MLP blocks (width 512, hidden 2560) map this vector to a readout, from which a K-way choice head selects the element as in the staged selector.

Placement and correction, that is the yaw, the log-amplitude and the residual of each control point, are then generated given an embedding of the chosen element, as the reference generates actions. On the DiT, a DiT-style transformer (width 256, five layers, adaptive layer norm on the readout, the element and the flow time) attends over one placement token and one token per control point and is trained by rectified flow, with 10 Euler steps at execution. On Diffusion Policy, the placement is a single vector rather than a sequence, so a diffusion MLP (three layers of width 512) samples it first. Diffusion Policy's conditional 1D U-Net, with down dimensions 104 and 208 instead of 512, 1024 and 2048, then denoises the correction's control points, with FiLM conditioning on the readout, the element and the sampled placement. Both are trained as DDPMs with 100 steps and sampled with 10 DDIM steps, without sample clipping since their targets are standardized.

The readouts and choice heads have 5.7M and 5.5M parameters and the generators 6.3M and 6.5M, so both selectors total 12.0M. A full-width variant of Diffusion Policy's selector uses Diffusion Policy's first two levels (512 and 1024), since a third does not fit the ten control points, and has a 72.7M generator. All selectors are trained for 50k steps (batch 128, learning rate 10^{-4}) on the reference's own 16-step chunks, with every 50th demonstration per task held out. Every sampled correction passes through the bound of Eq. 4.

Agreement and yaw error. Agreement is the fraction of chunks whose top choice is the labeled movement prototype z^*. We train the staged selector (seed 0) with every 50th demonstration held out. It agrees on 0.96 of held-out timesteps of its training demonstrations but on only 0.30 of the 11,653 chunks of unseen demonstrations (0.66 within its top five). On these chunks, the predicted yaw differs from θ^* by a median of 0.19 rad (11°), and by 0.93 rad (53°) at the 90th percentile.

Frozen and end-to-end encoder. The frozen ACT selector of Table 3 starts from each seed's bare ACT, freezes the ResNet trunk, the embeddings and the transformer encoder, and trains only the one-layer decoder, its query and the three heads (5.6M parameters, same data and budget as the staged selector). Offline it fits held-out timesteps slightly worse than the staged selector (action MSE 0.013 against 0.012), yet its success is 3.6 ± 3.9 points higher, within seed noise. The two differ mainly in execution (Tables 14 and 15), so training the encoder on the selection objective changes the decomposition little.

Rank of the selectors' choices. In closed loop we log the rank of each executed choice in the reference's ordering, where rank 1 is the movement prototype nearest to the reference's chunk. Table 11 reports these ranks under the reference's placement and correction (L_{idx}), together with a rank-matched rule. The rule draws each chunk's rank independently from the selector's per-chunk ranks at the same seed and budget and executes the movement prototype at that rank. The staged selector's ranks have a long tail, with 28 to 29% of its choices beyond rank 20 and a 90th percentile of 112 to 140. Table 12 reports the forced ranks of Fig. 6. Without any correction (β=0), the staged selector's choices lie farther from the reference's chunk, yet they cost far less than the fifth-nearest movement prototype.

Table 11: Rank of the selectors' choices in the reference's ordering, and the rank-matched rule (LIBERO-90, K=256). $\mathrm{SR}(L_{\mathrm{idx}})$ is re-evaluated with the ranks logged and lies within repeat noise of Table 14. The rank-matched rule executes movement prototypes at ranks drawn from the selector's own rank distribution. Mean ± sd over three seeds, differences paired per seed. –: not evaluated.

		ACT selector	
		staged	frozen
β=0.5	choices at rank 1 [%]	26	75
	choices in top 5 [%]	52	92
	median rank	4–5	1
	$\mathrm{SR}(L_{\mathrm{idx}})$	78.7 ± 1.9	78.3 ± 2.0
	rank-matched rule	45.0 ± 3.2	71.2 ± 1.8
	difference	−33.6 ± 1.9	−7.1 ± 2.0
β=0	choices at rank 1 [%]	19	–
	$\mathrm{SR}_{\mathrm{ref}}$	59.2 ± 1.0	–
	$\mathrm{SR}(L_{\mathrm{idx}})$	56.3 ± 1.0	–
	$\mathrm{SR}(L)$	56.0 ± 2.6	–
	rank-matched rule − $\mathrm{SR}(L_{\mathrm{idx}})$	−41.7 ± 2.0	–

Element-conditioned heads. The control replaces the staged selector's linear yaw and residual heads by two-layer MLPs (width 512, GELU, 563k parameters against 37k). The conditioned variant adds a linear embedding of the chosen movement prototype's control points to the decoder token (36k parameters, +0.10% of the model). It is trained on the label z^* and executes the selector's own choice. The third variant has the same architecture, but after a 10k-step warm-up it conditions on the selector's own detached choice, and its residual target is computed against that element. All variants use the recipe and bound of the staged selector. Table 13 reports their decomposition. Relative

Table 12: Forced choices under the reference's placement and correction. Change in success when the layer executes the r-th nearest movement prototype to the reference's chunk, or a uniformly random one, instead of the nearest (LIBERO-90, K=256). Mean ± sd over three seeds, paired per seed.

	β=0.5					β=0		
reference	$\mathrm{SR}_{\mathrm{ref}}$	r=2	r=5	r=20	random	$\mathrm{SR}_{\mathrm{ref}}$	r=5	random
ACT	80.2 ± 1.0	−6.2 ± 0.5	−18.9 ± 2.6	−54.9 ± 1.5	−68.7 ± 1.7	59.2 ± 1.0	−27.4 ± 0.6	−57.7 ± 1.0
flow DiT	83.4 ± 1.4	−4.6 ± 1.1	−12.7 ± 1.5	−47.4 ± 3.3	−66.3 ± 1.2	62.6 ± 1.9	−25.0 ± 0.8	−60.2 ± 1.6
Diff. Policy	92.9 ± 1.2	−2.6 ± 0.9	−10.4 ± 0.7	−50.2 ± 2.3	−73.1 ± 1.3	74.9 ± 1.8	−24.5 ± 3.4	−71.9 ± 2.2

to the control, the execution term changes by $+2.2 \pm 3.4$ with conditioning and $+3.0 \pm 3.6$ with supervision on the own choice. At seed 0 the bound clips 62% and 59% of the conditioned heads' chunks, against 34% for the control, so their gain does not come from a looser bound.

Table 13: Selector heads with and without the chosen element (ACT staged, LIBERO-90, K=256, β=0.5). Execution in both substitution orders. Cosine: median cosine in the Gram metric between the learned correction and the demonstrated one on held-out frames (ACT's own correction: 0.74). Mean ± sd over three seeds, paired per seed.

			execution		
heads	$\mathrm{SR}(L)$	$\mathrm{SR}(L_{\mathrm{idx}})$	choice first	correction first	cosine
linear (staged)	63.6 ± 1.7	78.1 ± 0.6	−14.6 ± 1.5	−19.0 ± 0.4	0.47
MLP control	65.6 ± 1.5	77.5 ± 1.2	−11.9 ± 2.1	−20.6 ± 2.2	0.52
conditioned, trained on label	69.7 ± 1.6	79.4 ± 0.8	−9.7 ± 1.4	−10.9 ± 0.5	0.66
conditioned, trained on own choice	69.9 ± 1.1	78.9 ± 0.5	−9.0 ± 1.6	−12.7 ± 1.2	0.65

Offline diagnosis of the learned correction. We compare corrections on each seed's validation split, a random 1/50 of timesteps (11,580 frames) that ACT and the selector both hold out. Each frame is executed with the evaluator's code at β=0.5. For the selector's element placed at the demonstrated yaw, the target e^* is the demonstrated chunk minus that element. We report medians over frames, averaged over seeds (seed sd at most 0.01). The linear heads' correction is poorly aligned with e^* (median cosine 0.47, Table 13) and is shorter, at 0.43 of $\|e^*\|_M$ against 0.90 for ACT's. Direction matters more than length. The executed chunk lies at a median Gram distance of 0.943 from the demonstration, 0.964 with the learned correction rescaled to the demonstrated norm and 0.616 with the demonstrated direction at the learned norm, against 0.450 for e^* itself. The selector's yaw error matches ACT's (median 6.4° against 6.5°), so the difference lies in the correction. These offline errors do not order the selectors by closed-loop success. The frozen-encoder selector has the largest offline error (0.980) but scores above the linear heads in closed loop.

Order of the terms. The hybrid L_{exec} executes the movement prototype nearest to the reference's chunk, placed by the selector's yaw head and corrected by its residual head under the same bound. Table 14 gives the success of every executed policy, Table 3 the terms in both orders and Table 15 their bootstrap intervals and the order-averaged selection term. Averaged over both orders, execution changes success by -16.8 ± 0.9, -12.3 ± 1.7, -1.6 ± 1.2 and -9.7 ± 0.3 for the staged selector and the three selectors of Table 3.

Other selector designs. The co-adapted selectors train the movement prototypes jointly with the selector under the IMC curriculum (Blessing et al., 2023), which enforces a target fraction of movement prototypes in use (0.95 and 0.70). The mixture selector instead executes a trained sparse mixture of its top-2 movement prototypes. All three start from the seed-0 w_g=1, K=256 library, whose library reference is within noise of the default (Table 16). They score 40.7 [34.0, 47.4], 29.4 [23.6, 35.6] and 28.9 [23.1, 34.9], against 61.6 for the staged selector at seed 0. All three saturate the bound on every chunk, with a median residual of 5 to 8 times the bound.

Table 14: Success of the executed policies behind Table 3, of the staged selector and of Diffusion Policy's full-width selector (LIBERO-90, K=256, β=0.5). Mean ± sd over three seeds.

		staged	frozen encoder			
	executed policy heads	ACT regression	ACT regression	flow DiT flow	Diff. Policy diffusion	Diff. Policy full width
$\mathrm{SR}(\pi)$	reference, bare	78.8 ± 2.3	81.9 ± 1.5		91.7 ± 0.6	
$\mathrm{SR}_{\mathrm{ref}}$	reference through the library	80.2 ± 1.0	83.4 ± 1.4		92.9 ± 1.2	
$\mathrm{SR}(L_{\mathrm{idx}})$	selector's choice, reference's correction	78.1 ± 0.6	79.3 ± 2.0	83.0 ± 2.0	92.4 ± 0.1	92.0 ± 0.8
$\mathrm{SR}(L_{\mathrm{exec}})$	reference's choice, selector's correction	61.2 ± 0.9	67.7 ± 3.1	80.3 ± 1.2	83.8 ± 1.0	87.6 ± 1.2
$\mathrm{SR}(L)$	selector's choice and correction	63.6 ± 1.7	67.1 ± 3.5	83.0 ± 0.0	82.0 ± 2.0	85.8 ± 2.1

Table 15: Terms of Table 3, of the staged selector and of Diffusion Policy's full-width selector, with paired 95% percentile bootstrap intervals over the 90 tasks (B=10,000, per-task differences averaged over the three seeds). Intervals that also resample seeds are up to 2.6 points wider. They still exclude zero for every execution term of the ACT selectors, but not for the staged selector's single-order selection terms.

	execution		selection		
selector	choice first	correction first	choice first	correction first	order average
ACT staged	-14.6 $[-17.7, -11.4]$	-19.0 $[-21.7, -16.3]$	-2.1 $[-3.9, -0.2]$	$+2.3$ $[+0.1, +4.6]$	$+0.1$ $[-1.4, +1.8]$
ACT frozen	-12.1 $[-14.7, -9.7]$	-12.4 $[-15.1, -9.9]$	-0.9 $[-2.3, +0.5]$	-0.6 $[-2.8, +1.5]$	-0.8 $[-2.1, +0.6]$
flow DiT	$+0.0$ $[-1.9, +1.8]$	-3.1 $[-4.6, -1.6]$	-0.4 $[-2.0, +1.2]$	$+2.7$ $[+0.9, +4.5]$	$+1.2$ $[-0.1, +2.5]$
Diff. Policy	-10.3 $[-12.4, -8.4]$	-9.0 $[-11.3, -6.9]$	-0.5 $[-1.7, +0.7]$	-1.8 $[-3.6, -0.0]$	-1.1 $[-2.2, -0.1]$
Diff. Policy, full width	-6.3 $[-8.0, -4.7]$	-5.2 $[-7.1, -3.4]$	-0.8 $[-2.0, +0.3]$	-1.9 $[-3.4, -0.3]$	-1.3 $[-2.4, -0.3]$

Language conditioning. Conditioned on CLIP embeddings of the instruction instead of the task id, bare ACT scores 72.1 ± 0.8 and wrapped ACT 71.6 ± 0.8, a paired cost of 0.6 ± 0.1 (Table 6). A language-conditioned staged selector, on a K=64 library (yaw and amplitude canonicalization, w_g=1), scores 45.3 [39.7, 51.1], 27 points below bare, against 15 under task-id conditioning. Loosening its bound to β=2, where almost no chunk is clipped, gives 47.7 [42.2, 53.2], and removing its residual (β=0) gives 32.1. A learned residual freed of the bound thus does not recover the loss.

F LIBRARY REFERENCE OF A LEARNED CODEBOOK

The library reference applies to a latent codebook only in a confounded form. QueST's codec consumes 32-step windows, so a reference's 20-step chunks must first be resampled. The resampling alone lowers ACT from 79.9 to 28.2 and the Diffusion Policy from 91.7 to 59.6. On identical initial states, the codec round trip then changes the resampled ACT by $+1.7$ $[-2.4, +5.8]$ and the resampled Diffusion Policy by -10.3 $[-14.3, -6.3]$. Because the decoder and its fixed time support cannot be separated from the quantization, Eq. 1 is ill-posed for this codebook. A tokenizer without these confounds behaves like the library. Round-tripping every Diffusion Policy chunk through FAST (Pertsch et al., 2025), which encodes each chunk at its native length, gives 92.8 [90.3, 95.0] against 92.2 bare.

G LIBRARY DESIGN, SIZE, TRANSFER AND EDITING

G.1 DESIGN SWEEP

Table 16 gives the full design sweep, and Table 4 is its $K{=}256$, $w_g{=}5$ column. Yaw canonicalization adds 10 points at $K{=}64$ and 3 to 4 at $K{=}256$, so it matters most where the library is too small to absorb the variety of orientations. The Gram metric adds 3 to 4 points over Euclidean distance at $K{=}64$ and is within noise at $K{=}256$. The two gripper weights are within noise of each other at $K{=}256$. Unprojected centroids score 79.8 [75.2, 84.0] against 79.4 for projected ones, and an IMC-style curriculum mixture (Blessing et al., 2023; Li et al., 2023b) scores 77.8 [72.4, 82.7]. The number of control points has little effect, with $B{=}6$, 10 and 14 giving 93.0, 92.1 and 93.4 for the Diffusion Policy.

Table 16: Closed-loop library reference of every library design. ACT (seed 0, bare 79.9) executed through each library at $\beta{=}0.5$, $q{=}0.75$, one 900-rollout run per cell (noise floor about 3 points). Rows form a cumulative ladder. Euclidean: clustering and assignment by Euclidean distance on control points. Gram: by the metric of Eq. 2, without canonicalization. w_g: gripper-channel weight.

	$K{=}64$		$K{=}256$	
design	$w_g{=}1$	$w_g{=}5$	$w_g{=}1$	$w_g{=}5$
Euclidean	65.6	62.0	73.2	74.6
Gram	69.1	66.0	76.4	76.7
+ yaw (ours)	79.3	76.0	80.3	79.4
+ amplitude	73.9	73.0	78.4	74.4

G.2 LIBRARY SIZE ON ROBOCASA

Doubling the in-suite RoboCasa library to $K{=}512$ changes no policy by more than a point against bare. ACT changes by $+0.4 \pm 2.7$ and the Diffusion Policy by $+0.6 \pm 1.9$ (three seeds), and the flow-matching policy by -0.6 $[-5.1, +4.1]$ (seed 0). Over $K \in \{64, 128, 256, 512, 1024\}$, the Diffusion Policy scores 40.0, 40.9, 42.1, 42.7 and 44.8 (bare 43.5), with task-level intervals of about ± 11.

G.3 TRANSFER

The LIBERO-90 library explains 89.0% of the variance of the 402,344 RoboCasa arm segments, against 92.0% for RoboCasa's own library. On LIBERO-10 it executes ACT at 64.5 ± 2.9, at parity with bare ACT and 5.8 ± 2.6 below LIBERO-10's own library. The occupancy counts of §6.2 are effective numbers of movement prototypes, the exponential of the entropy of a corpus's assignment frequencies.

G.4 EDITING

Table 17 reports edits of the LIBERO-90 library toward RoboCasa. Re-clustering the union of the two corpora at $K{=}256$ brings RoboCasa within 1.2 points of its own library and costs LIBERO-90 0.4 ± 0.6. At $K{=}512$, neither column moves by more than a point. Appending 64 RoboCasa movement prototypes leaves LIBERO-90 unchanged and recovers part of the RoboCasa shortfall (-3.3 against -4.8). Choosing the 64 appended movement prototypes differently, as RoboCasa's own $K{=}64$ library or at random, makes no measurable difference (seed 0, five-point intervals over 17 tasks). For the Diffusion Policy (seed 0), the re-clustered library gives $+1.2$ on RoboCasa and $+1.0$ on LIBERO-90. Each edit is a two-minute clustering or a concatenation, and the policies remain untouched.

Deleting fast movement prototypes. We delete from the $K{=}256$ library the 64 movement prototypes whose largest commanded per-step translation reaches 0.843 in normalized action units, the library's 75th percentile. Table 18 reports wrapped ACT with the full and the edited library, together with the share of executed chunks whose commanded speed reaches this threshold. With the full

Table 17: Library editing toward RoboCasa (ACT, β=0.5, q=0.75). Paired difference against each suite's own K=256 library. Mean $\pm$ sd over three seeds where given, otherwise seed 0 with 95% bootstrap CIs over tasks. Appended: the 64 most-populated of RoboCasa's 256 movement prototypes.

		Δ against own library	
edit	K	RoboCasa policy	LIBERO-90 policy
none (LIBERO-90 library)	256	-4.8 [-10.2, $+0.9$]	—
re-cluster LIBERO-90 $\cup$ RoboCasa	256	-1.2 [-5.5, $+2.7$]	-0.4 ± 0.6
	512	-2.0 [-6.7, $+2.9$]	-0.6 [-3.4, $+2.4$]
append 64 RoboCasa prototypes	320	-3.3 [-6.8, $+0.1$]	-0.2 ± 0.7

library, tightening the bound barely lowers this share, because the library's own fast elements remain. The threshold is a quantile of the library, not a physical limit, and speeds are commanded, not measured.

Table 18: Deleting the fastest quarter of the library (ACT, LIBERO-90, K=256). Success: mean $\pm$ sd over three seeds. Effect of deletion: paired against the full library at the same β, with 95% bootstrap CIs over tasks. Fast: executed chunks with commanded speed ≥ 0.843. All cells were evaluated together, so bare and full-library success differ from Table 1 within repeat noise.

	success			fast [%]	
β	full	deleted	effect of deletion	full	deleted
bare ACT	79.0 $\pm$ 0.6			28.4	
0.5	79.2 $\pm$ 1.8	79.3 $\pm$ 1.2	$+0.0$ [-1.6, $+1.6$]	19.6	11.9
0.25	71.9 $\pm$ 1.0	69.1 $\pm$ 0.6	-2.7 [-5.0, -0.4]	16.0	4.5
0.1	64.6 $\pm$ 0.4	61.0 $\pm$ 0.3	-3.5 [-7.1, -0.2]	14.4	0.8
0	58.5 $\pm$ 1.7	53.7 $\pm$ 2.1	-4.9 [-8.0, -1.8]	13.8	0.0

H Extended related work

Discrete skill representations. BeT (Shafiullah et al., 2022) is the closest structural relative, since its bin plus offset resembles a movement prototype plus residual. Its bins, however, are means of raw single-step actions rather than canonicalized chunks under a trajectory metric. Its selector is a trained classifier rather than a lookup on the policy's own output, and its offsets are unbounded. Methods that learn continuous skill latents (Pertsch et al., 2020; Ajay et al., 2021; Nasiriany et al., 2022) place the bottleneck inside the policy but have no quantization term. Concurrent work pairs a learned selector with an unbounded residual (Lei et al., 2026; Wang et al., 2026). In our experiments, an unbounded regression residual does not recover what a selector's learned correction loses, while a generative head recovers part of it within the bound (§5, Appendix E).

Trajectory primitives. Dynamical movement primitives encode motions as attractor systems (Ijspeert et al., 2013), and probabilistic movement primitives encode them as distributions over trajectories (Paraschos et al., 2013; Li et al., 2023a). Trajectory libraries turn stored trajectories into a policy by executing the one whose states are nearest to the current state (Stolle & Atkeson, 2006). Maneuver automata compose a finite set of maneuvers, each defined up to the symmetry group of the system, so that one stored maneuver serves every translated or rotated instance (Frazzoli et al., 2005). The yaw canonicalization of §3.3 applies the same idea to demonstration segments, restricted to rotations about gravity.

Prototypes, retrieval and runtime projection. ProtoPNet (Chen et al., 2019) classifies inputs by their similarity to prototypes projected onto training examples, and it has been extended to sequences (Ming et al., 2019). We adopt its projection step, and the selector of §5 is a classifier over the library's elements. Retrieval of sub-trajectories for training (Memmel et al., 2025; Nasiriany et al., 2022) leaves the executed action unconstrained, as retrieval over observations does. Bidirectional decoding (Liu et al., 2025) samples several chunks and keeps the one most consistent with past and

likely future actions, so the executed chunk is still one that the policy produced. Residual policy learning (Silver et al., 2018; Johannink et al., 2019) trains a correction on top of a fixed controller. MPQ inverts these roles, since the demonstrated segment is fixed and the policy's prediction supplies a bounded correction. Safety shields and control-barrier-function filters (Alshiekh et al., 2018; Ames et al., 2019) project a policy's action onto an implicit safe set rather than onto a finite library, and runtime monitors (Ahmed & Begum, 2026) detect unsafe executions without changing the action.

I PROTOTYPE CARDS

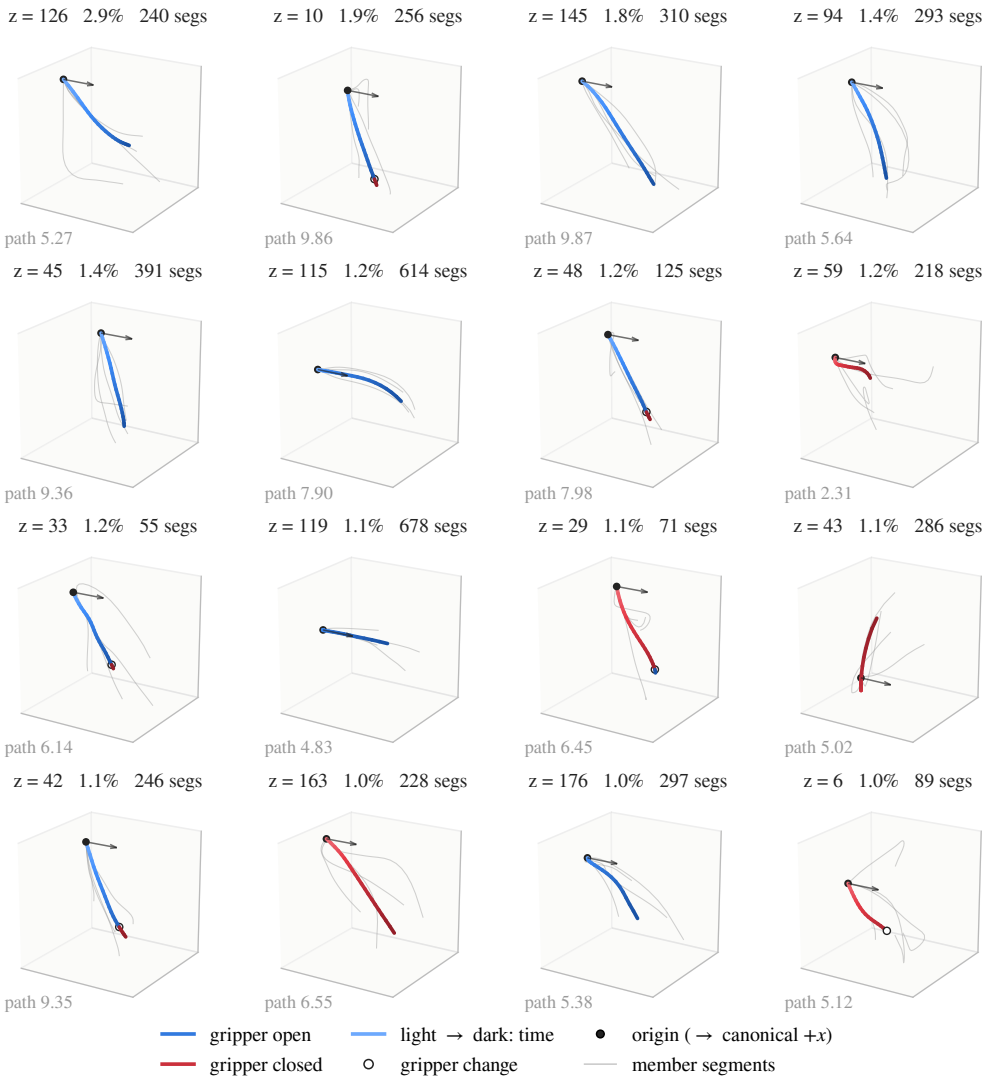


Figure 7: The sixteen most-used movement prototypes of wrapped ACT on LIBERO-90, ranked by closed-loop usage over the rollouts of §4.2, excluding the near-idle z=149 (Fig. 2, Appendix D.4). Each card shows the decoded end-effector path in the canonical frame from a shared camera, scaled to its own extent. Headers: index z, usage share and number of member segments. Path: path length in action units, which restores absolute scale.