ACTIONCODEC2: A STREAMING ACTION TOKENIZER

Anonymous authors
Paper under double-blind review

ABSTRACT

In autoregressive vision-language-action (AR-VLA) models, action tokenizers map continuous actions to discrete tokens and back to executable actions. Most existing tokenizers are chunkwise, allowing the robot to react only at chunk boundaries. AR models can instead support streaming tokenization, where each action is decoded as soon as its tokens are available. Existing streaming designs, however, either require many tokens or lack distribution-independent reconstruction guarantees, while small errors in incremental actions can accumulate into trajectory drift. We therefore seek the minimum token rate subject to bounded reconstruction error and a user-specified trajectory tolerance. Under the group structure of the action space, this becomes a covering problem. We prove fixed motion primitives optimal among one-label-per-step decoders, with an exact closed-form solution for translation and constant-factor-tight bounds for orientation. We further compress the primitive stream losslessly with Set-BPE, a byte-pair encoding over unordered sets. The resulting **ActionCodec2** requires no gradient training. Fitted on 20k hours from 24 embodiments, it encodes one second of 15 Hz motion in 8.8–10.8 tokens on average. Using the same VLA backbone, ActionCodec2 outperforms the compared tokenizers on LIBERO (98.2%), LIBERO-Plus (79.5%), and SimplerEnv (93.8%), and reaches 73.9% task progress across four zero-shot Franka tasks. More results are available at `https://actioncodec2-anon.pages.dev`.

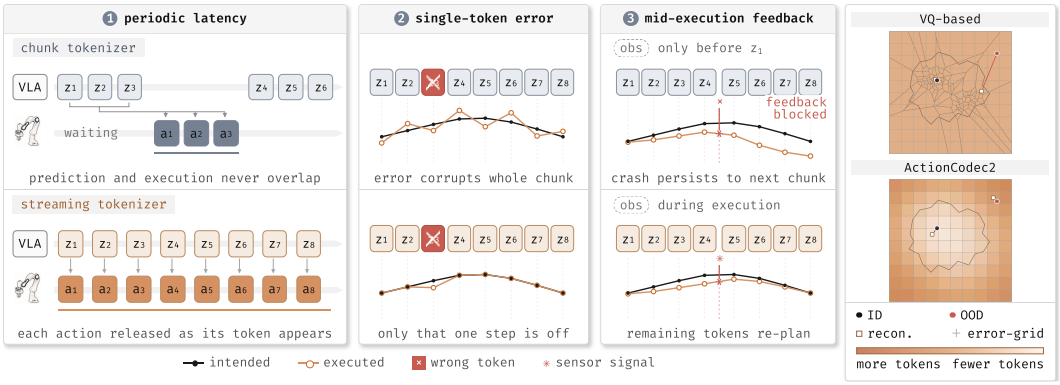


Figure 1: **Why stream actions and cover their reachable range.** (1) Chunkwise decoding waits for the whole chunk, while streaming releases each action as its token arrives. (2) A wrong token corrupts the rest of a chunk, but only one streaming step. (3) Streaming can use feedback during execution and re-plan before the next chunk. At right, a fixed VQ codebook concentrates codes in distribution and incurs large out-of-distribution error; ActionCodec2 covers the reachable range and spends more tokens when needed.

1 INTRODUCTION

In an autoregressive (AR) VLA, the action tokenizer is the interface between the model and the robot [51, 15, 29, 9]. It encodes continuous actions into discrete tokens for training and decodes the predicted tokens back into executable actions. Following the chunked-action formulation widely used in diffusion policies [8, 47, 3], most existing action tokenizers encode one to two seconds of motion jointly into a token sequence that is decoded as a whole (Figure 1, left). FAST [29] compresses the DCT coefficients of a chunk with BPE, and VQ-VAE-based tokenizers [25, 38, 20, 13, 9] learn to

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

quantize a chunk into a few latent codes. Because the chunk is encoded jointly, none of its actions can be decoded until all tokens have been generated. The robot must therefore wait for each chunk before acting and can respond to new observations only at chunk boundaries. Moreover, an erroneous token can perturb multiple actions within the chunk rather than only a single step. Real-time chunking [6] hides the waiting time by decoding the next chunk during execution, but still does not allow the policy to react within a chunk, as required by control loops driven by high-frequency tactile feedback. Chunkwise prediction, however, is not required by an AR VLA, which can instead decode actions in a streaming manner, executing each action as soon as the tokens representing it are available.

The most direct way to obtain a streaming tokenizer is the per-dimension binning of OpenVLA [15], which discretizes each action dimension independently into 256 bins. Under this scheme, one second of 7-dimensional actions at 15Hz requires 105 tokens, increasing sequence length during training and decoding cost at inference. A second option is to shorten a learned chunkwise tokenizer to one or two steps. Most of such designs use VQ-VAE-style quantization [25, 38, 24, 20]. Such a tokenizer represents every action with the same number of codes from a fixed codebook, so the information available to each action is bounded. The learned codebook allocates this finite capacity according to the training distribution. As a result, reconstruction resolution is highest in regions well covered by the training data, while no comparable accuracy guarantee exists for motions outside those regions (Figure 1, right). OpenVLA incurs a high token rate, whereas fixed-rate learned tokenizers provide no distribution-independent reconstruction guarantee.

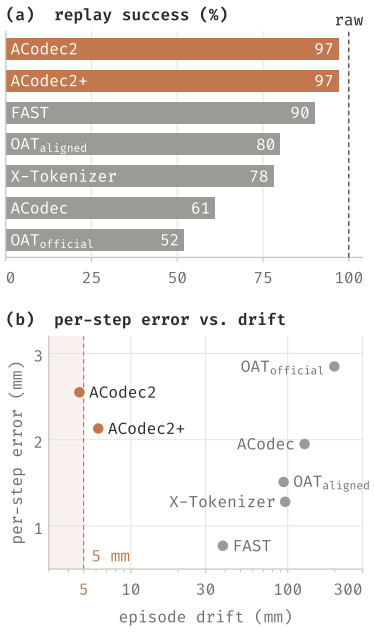


Figure 2: **Round-trip fidelity of action tokenizers on LIBERO demonstrations** (first-order variants). (a) Demonstrations are tokenized, decoded, and replayed open loop; raw demonstrations score 100%. (b) Per-step error versus mean episode drift. Current tokenizers achieve high per-step accuracy but leave the episode drift unbounded; ActionCodec2 keeps the tracking error within its declared bound and attains the highest replay success.

We require a streaming action tokenizer to satisfy two properties. First, reconstruction accuracy should be bounded independently of the training distribution. An unfamiliar motion may require more tokens, but not a larger reconstruction error. Second, the decoded trajectory should remain within a user-specified tracking tolerance, such as $5\,\mathrm{mm}$, at every step. For incremental action representations, bounding each action independently is insufficient because small per-step errors accumulate along the trajectory. In a replay test on LIBERO (Figure 2), FAST reconstructs individual actions to within $1\,\mathrm{mm}$ but accumulates about $40\,\mathrm{mm}$ of drift per episode and replays only 90% of the demonstrations successfully, whereas ActionCodec2 keeps the drift below $5\,\mathrm{mm}$ and succeeds on 97%. Among tokenizers satisfying these two requirements, we seek to minimize the number of emitted tokens. For the action spaces considered here, the update operation admits a group structure. Under a per-step tracking constraint, tokenizer design can then be formulated as a covering problem: how many symbols are required to keep the reconstructed trajectory within the prescribed error bound? We show that a finite codebook suffices and derive the exact minimum for simple update rules such as first-order differencing. On our pretraining corpus, about ten bins per action dimension suffice to keep the translational error below $5\,\mathrm{mm}$. We then reduce the token count with a lossless packing stage based on BPE [29, 33]. We introduce *Set-BPE*, which operates on unordered sets of per-dimension symbols and can therefore merge co-occurring symbols across arbitrary action dimensions rather than only adjacent symbols in a fixed serialization.

These components form **ActionCodec2**, a training-free[1] streaming action tokenizer. We fit ActionCodec2 on about 20k hours of data from 24 embodiments. On this corpus, one second of $15\,\mathrm{Hz}$ motion requires 8.8 tokens on average for end-effector control and 10.8 for joint control. We evaluate ActionCodec2 against four baselines, FAST [29], OAT [20], X-Tokenizer [13], and ActionCodec [9], using the same G0.5 VLA backbone [23]. ActionCodec2 reaches 98.2% on LIBERO, 93.8% on SimplerEnv, and 79.5% on LIBERO-Plus, outperforming the four baselines on all three benchmarks. On

[1]By *training-free* we mean free of gradient-based optimization of neural-network parameters; the covering tables and the Set-BPE merges are still data-driven, fitted on the pretraining corpus.

a physical Franka arm, we deploy the resulting VLA zero-shot on four tasks: color recognition, snack serving, nested-cup arrangement, and towel unfolding. The streaming policy executes continuously without the periodic pauses of chunkwise baselines, and ActionCodec2 achieves 73.9% task progress, the highest among the compared tokenizers. Our contributions are summarized as follows:

- We formulate streaming action tokenization under a per-step tracking constraint as a covering problem, and characterize the minimum codebook size for composition decoders: exactly for translation and up to order-tight bounds for orientation.
- We introduce *Set-BPE*, a lossless packing scheme that does not require a fixed ordering of action dimensions and reduces token count by 8.0–11.9% relative to standard BPE on held-out data.
- We fit ActionCodec2 on 20k hours of data from 24 embodiments and evaluate it in the same G0.5 VLA, obtaining 98.2% on LIBERO, 93.8% on SimplerEnv, 79.5% on LIBERO-Plus, and 73.9% task progress across four zero-shot Franka tasks.

2 PRELIMINARIES

We consider action sequences $q_{0:T} = (q_0, \dots, q_T)$ taking values in a group G: end-effector translation lives in $(\mathbb{R}^3, +)$, joint angles in $(\mathbb{R}^n, +)$, orientation in $\mathrm{SO}(3)$. An action tokenizer maps such a sequence to discrete labels $z_0, z_1, \dots$ from a vocabulary of size K, and a decoder maps the labels back to an executable trajectory $\hat{q}_{0:T}$. Encoder and decoder share the initial configuration at no cost in labels, so $\hat{q}_0 = q_0$. Let $\nu(t)$ be the number of labels the decoder consumes before it releases step t. We call a tokenizer *streaming* if $\nu(t) > \nu(t-1)$ for every $t \geq 1$: every further action costs at least one further label. We call it *chunkwise with length* τ if ν is constant on each block $\{m\tau + 1, \dots, (m+1)\tau\}$ and increases only between blocks: the robot idles until an entire chunk has been decoded. In this paper, we focus on streaming tokenizers. At every configuration the actuators realize a motion from $\mathcal{A} \subseteq G$, so every action sequence satisfies $q_{t+1} = q_t \delta_t$ with $\delta_t \in \mathcal{A}$. Let $\xi_t := \hat{q}_t^{-1} q_t$ be the error. We require one property of the tokenizer throughout, *bounded error*: given a tolerance set $\mathcal{E} \subseteq G$ containing the identity, $\xi_t \in \mathcal{E}$ for every $t \geq 0$, on every action sequence of any length. Bounded error is a per-step worst-case guarantee rather than an average reconstruction error. Our goal is to make K as small as possible while bounded error holds.

3 THE SMALLEST STREAMING LABEL SET

We consider decoders that read one label per step and update the reconstruction according to that:

$$\hat{q}_{t+1} = y_{z_t}(\hat{q}_t), \qquad z_t \in \{1, \dots, K\}, \qquad \hat{q}_0 = q_0, \tag{1}$$

with $y_1, \dots, y_K : G \to G$ arbitrary. This form also covers decoders with memory, since a decoder that reads back n steps is again (1) over a stacked configuration. The bounded-error guarantee must hold on trajectories of every length, but it reduces to a condition on a single transition. Write $\mathcal{S} = \{(q, \hat{q}) : \hat{q}^{-1} q \in \mathcal{E}\}$ for pairs the tolerance accepts and $\mathcal{F}(q)$ for the feasible successors of q, and ask that, for every accepted pair and successor in $\mathcal{F}(q)$, some label keeps the new pair accepted:

$$\forall (q, \hat{q}) \in \mathcal{S}, \quad \forall q^+ \in \mathcal{F}(q), \quad \exists k \leq K : \ \big(q^+, y_k(\hat{q})\big) \in \mathcal{S}. \tag{2}$$

Call this the *full tolerance requirement*, and let $K^\star$ be the smallest K that meets it.

Lemma 1 (One step suffices). *If the full tolerance requirement holds, there is a causal encoding policy $z_t = \pi_t(q_{0:t+1})$ under which the decoder keeps $\xi_t \in \mathcal{E}$ for every t, from every exact start and on every feasible trajectory of any length.*

The requirement depends on the tolerance and the feasible successors alone, which is what makes the optimum computable; the second-order variant, whose feasible successors depend on state, relaxes it to the pairs a decoder can reach (Appendix F.1). The maps y_k are still arbitrary: scalar binning [15] and a learned per-codeword decoder [36] are both instances of (1). Let each label instead apply a fixed relative motion by *composition*, $y_k(\hat{q}) = \hat{q} u_k$, for a list $\mathcal{U} = \{u_1, \dots, u_K\} \subseteq G$. In one step the error can grow anywhere in $\mathcal{E}\mathcal{A} := \{\xi\delta : \xi \in \mathcal{E},\ \delta \in \mathcal{A}\}$. Applying u returns to the tolerance exactly the part of $\mathcal{E}\mathcal{A}$ lying in $u\mathcal{E} := \{u\eta : \eta \in \mathcal{E}\}$, so leaving nothing uncovered means $\mathcal{E}\mathcal{A} \subseteq \bigcup_{u \in \mathcal{U}} u\mathcal{E}$.

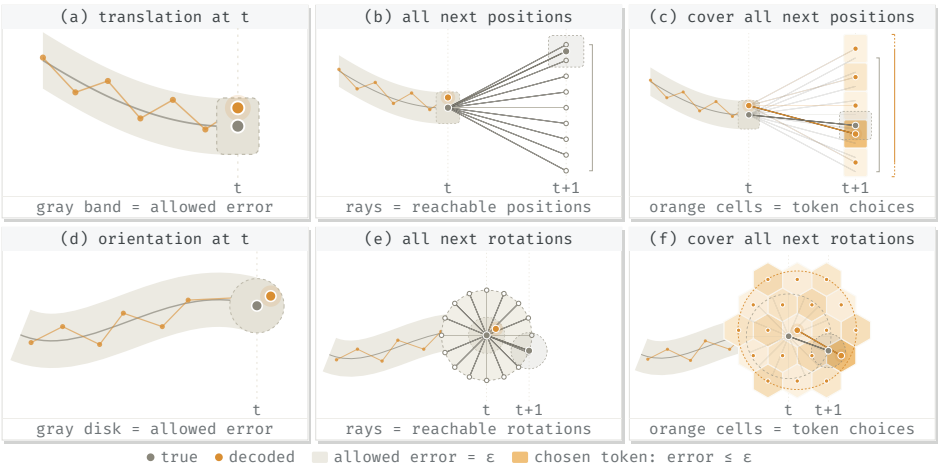


Figure 3: **Covering primitives for translation and orientation.** The decoded state starts within the allowed error of the true state. The next true state may land anywhere in the reachable set, so candidate tokens cover that set; the chosen token keeps the tracking error within ϵ.

Theorem 2 (Optimality of composition). *Under the full tolerance requirement, (1) the composition decoder with list $\mathcal{U}$ meets it if and only if $\mathcal{EA} \subseteq \bigcup_{u\in\mathcal{U}} u\mathcal{E}$; (2) every decoder of the general form* (1) *that meets it uses at least $\min\{|\mathcal{U}| : \mathcal{U} \subseteq G \text{ finite}, \ \mathcal{EA} \subseteq \bigcup_{u\in\mathcal{U}} u\mathcal{E}\}$ labels, the minimum over an empty family read as $+\infty$; (3) hence $K^\star$ equals that minimum, and composition attains it whenever it is finite. (4) All three survive a fixed distortion of the error between steps: if $q_{t+1} = \varphi(q_t)\delta_t$ for an automorphism φ of G and the decoder composes after it, $y_k(\hat{q}) = \varphi(\hat{q})u_k$, then $\xi_{t+1} = u^{-1}\varphi(\xi_t)\delta_t$ and (1)–(3) hold with $\mathcal{EA}$ replaced by $\varphi(\mathcal{E})\mathcal{A}$.*

The lower bound assumes nothing about the encoder, so under the same requirement it also binds an encoder that sees the whole trajectory in advance, while the matching construction needs only a causal one: the offline lookahead used to tokenize a dataset buys nothing a causal decoder cannot recover. The qualifier matters: the relaxed criterion of Proposition 6 uses causality and does not cover a non-causal encoder. Part (4) is the form the higher-order rungs use: Appendix F is the case where φ depends on the state, where the closed form gives way to a fixed-point kernel.

4 Practical cases: translation and orientation

We instantiate Theorem 2 in the two settings a robot codec actually meets: translation and orientation (Figure 3). Translation covers end-effector position and joint control; orientation covers end-effector rotation. Substituting each setting into the theorem gives the primitive list $\mathcal{U}$ and its size $K^\star$.

Corollary 3 (Translation). *Let $G = (\mathbb{R}^d, +)$ with box motion set $\mathcal{A} = \prod_{j=1}^d [\underline{a}_j, \overline{a}_j]$ and tolerance $\mathcal{E} = \prod_{j=1}^d [-\epsilon_j, \epsilon_j]$, writing $\Delta a_j := \overline{a}_j - \underline{a}_j$ with $\epsilon_j > 0$. Here $\mathcal{EA} \subseteq \bigcup_{u\in\mathcal{U}} u\mathcal{E}$ reads $\mathcal{E} + \mathcal{A} \subseteq \bigcup_{u\in\mathcal{U}}(u + \mathcal{E})$, and the smallest alphabet meeting the full tolerance requirement has size*

$$K^\star_{\text{trans}} = \prod_{j=1}^{d} N_j, \qquad N_j = \left\lceil 1 + \frac{\Delta a_j}{2\epsilon_j} \right\rceil, \tag{3}$$

attained by axiswise uniform $\mathcal{U} = \prod_j \mathcal{U}_j$ of Algorithm 1, with $\mathcal{U}_j = \{\underline{a}_j\}$ if $\Delta a_j = 0$ and otherwise

$$\mathcal{U}_j = \left\{ \underline{a}_j + \frac{\Delta a_j (i-1)}{N_j - 1} : i = 1, \ldots, N_j \right\}. \tag{4}$$

If real trajectories stay in a workspace, clipping each reconstructed successor to it preserves the error guarantee, so the same alphabet remains sufficient; the workspace may admit a smaller optimum.

Corollary 4 (Orientation). *Let $G = \mathrm{SO}(3)$ with body-frame increments, $R_{t+1} = R_t\Delta_t$ and $\hat{R}_{t+1} = \hat{R}_t U$, error measured by the intrinsic angle $\theta(Q) = \arccos\big((\mathrm{tr}(Q) - 1)/2\big) \in [0, \pi]$. Let $\mathcal{B}_R(r) = \{Q : \theta(Q) \le r\}$, with tolerance $\mathcal{E} = \mathcal{B}_R(\epsilon_R)$ and motions $\mathcal{A}_R = \mathcal{B}_R(b_R)$ of any axis,*

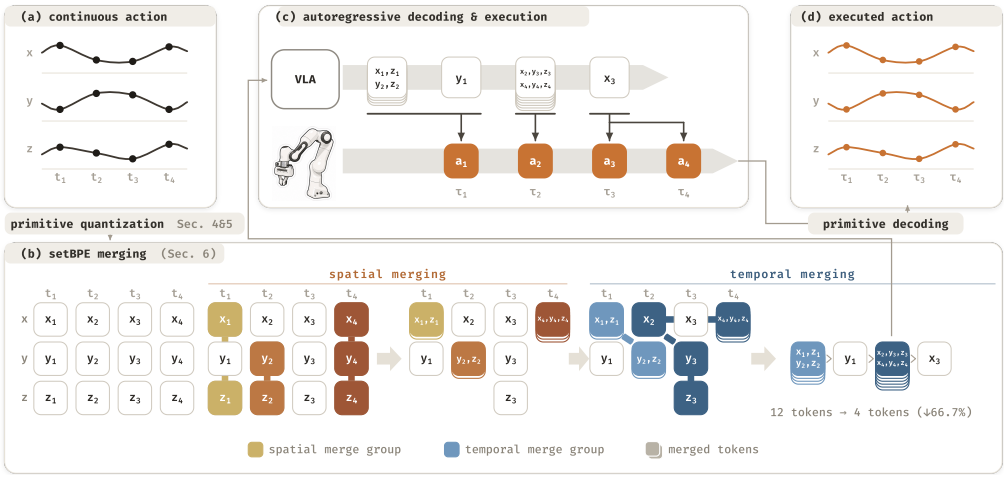


Figure 4: **ActionCodec2.** (a) Each step of a continuous action is quantized into primitive atoms, one per coordinate. (b) Set-BPE merges atoms that co-occur within a step and then across nearby steps, independent of coordinate order. (c) The VLA emits the tokens autoregressively and releases each action once its step is complete. (d) The decoded primitives track the trajectory within ϵ.

$0 < \epsilon_R \le \pi$ *and* $0 \le b_R \le \pi$. *Here* $\mathcal{EA} \subseteq \bigcup_{u\in\mathcal{U}} u\mathcal{E}$ *reads* $\mathcal{B}_R(\epsilon_R)\mathcal{A}_R \subseteq \bigcup_{k=1}^{K} U_k \mathcal{B}_R(\epsilon_R)$, *and with* $\rho_R = \min\{\pi, \epsilon_R + b_R\}$, $\beta_R = b_R/\epsilon_R$, *and* $V_R(r) = (r - \sin r)/\pi$ *the normalized Haar volume of* $\mathcal{B}_R(r)$, *the smallest such alphabet obeys*

$$\left\lceil \frac{V_R(\rho_R)}{V_R(\epsilon_R)} \right\rceil \le K^{\star}_{\mathrm{rot}} \le \left\lfloor \frac{V_R(\min\{\pi, \rho_R + \epsilon_R/2\})}{V_R(\epsilon_R/2)} \right\rfloor, \tag{5}$$

so that $K^{\star}_{\mathrm{rot}} = \Theta\big((1+\beta_R)^3\big)$ *whenever* $\rho_R + \frac{\epsilon_R}{2} \le r_0 < \pi$, *with constants depending on* r_0 *alone. Algorithm 1, with step* $0 < h < \frac{2\epsilon_R}{\sqrt{3}}$, *returns a list satisfying the inclusion, not claimed to be minimal.*

Corollary 4 assumes body-frame increments; a controller in the world frame reaches the same $K^{\star}$ whenever its motion set is closed under inversion, a geodesic ball in particular.

Proposition 5 (Frame equivalence). *Let increments act on the left,* $R_{t+1} = \Delta_t^w R_t$ *and* $\hat{R}_{t+1} = U\hat{R}_t$, *so that the error is* $\Xi_t^w = R_t\hat{R}_t^{\top}$ *and the covering condition is* $\mathcal{A}_R^w \mathcal{B}_R(\epsilon_R) \subseteq \bigcup_k \mathcal{B}_R(\epsilon_R)U_k$. *Its minimum equals the body-frame* $K^{\star}_{\mathrm{rot}}$ *for the motion set* $(\mathcal{A}_R^w)^{\top}$. *The two coincide whenever* $\mathcal{A}_R^w$ *is closed under inversion, a geodesic ball in particular.*

Algorithm 1 gives pseudocode for both lists and the nearest-primitive encoder that picks a label at run time; the translation list is the closed-form grid, the orientation list a greedy cover with a local refinement. In a concrete task the inputs are the embodiment's per-step motion bounds and the declared tolerance: fixing $(\underline{a}_j, \overline{a}_j)$ and ϵ_j for translation, or (ϵ_R, b_R) for orientation, determines $\mathcal{U}$ and $K^{\star}$ outright. The result is the smallest streaming vocabulary Theorem 2 permits, with the per-step error guarantee and no training.

5 Set-BPE: Order-free lossless packing

Section 4 fixes which primitive is chosen at each step; what remains is how the index sequence $Z_{1:T}$ is written for an AR model, and a lossless merge vocabulary trained on the data [33] is the standard tool—it cannot touch the guarantee. But BPE fuses only what a serialization places next to each other, so adjacency reflects the chosen coordinate order, not the data. In end-effector coordinates $(x, y, z, r_x, r_y, r_z, g)$, advancing along z while turning about it couples z with r_z, three positions apart, and no merge reaches the pair; every ordering strands some such pair. Across embodiments the problem compounds: coordinate sets differ, serialized positions shift with their companions, and merges learned under one layout do not transfer. Set-BPE instead merges *sets* of coordinate assignments, so co-occurring quantities fuse regardless of layout (Figure 4). Write the joint choice at step t as $S_t = \{(j, z_{t,j})\}_{j=1}^{D}$: one of $K_{\mathrm{joint}} = \prod_{j=1}^{D} N_j$ values, built from only $\sum_j N_j$ distinct *atoms*, each naming one coordinate and one bin; an atom is a token to begin with. A *spatial* token

Table 1: **Success rate (%) averaged within each simulation benchmark.** Every column is the same G0.5 [23] trained with the corresponding tokenizer; "+" marks the pretrained universal tokenizer and "II" the second order, and OAT is run at its official and at an ACodec2-matched token budget. LIBERO: 2,000 episodes; LIBERO-Plus: 10,030 perturbed tasks; SimplerEnv: 480 episodes. **Bold** is the best in a row, underline the second best.

Benchmark	FAST	$\text{OAT}_{\text{official}}$	$\text{OAT}_{\text{aligned}}$	ACodec	X-Tokenizer	ACodec2	ACodec2+	$\text{ACodec2}^{\text{II}}$	$\text{ACodec2+}^{\text{II}}$
LIBERO	92.8	91.0	96.4	95.1	88.8	96.9	96.7	<u>97.3</u>	**98.2**
LIBERO-Plus	69.7	64.3	73.2	69.4	52.9	71.8	74.8	<u>77.7</u>	**79.5**
SimplerEnv	86.0	12.9	19.0	64.4	83.3	90.8	91.9	<u>93.1</u>	**93.8**

expands to atoms on distinct coordinates, and the tokens present at a step partition S_t; a *spatial merge* replaces two co-occurring tokens with disjoint coordinates by one token for their union. A *temporal* token crosses steps: spatial tokens at offsets $0, \ldots, \tau - 1$ with exactly one at offset zero, its *root*; a *temporal merge* fuses two instances $\delta < \tau$ steps apart whose combined span fits in τ, the child rooted where the left parent was. The stream carries one canonical order, the fixed target an autoregressive policy trains against: instances ordered by start step, ties by the smallest coordinate of the root, no timestamps. The decoder recovers every step boundary from the stream alone—the next root always sits at the lowest unassigned coordinate of the earliest incomplete step—and every prefix admits a legal continuation, because a primitive atom always exists for the next uncovered coordinate. Writing Y for the atom instances, the merges compose into a bijection with the token stream: $H(Y) = H(Z_{1:T})$, the trajectory unchanged.

A merge's gain is the number of instance pairs it fuses, so greedy training needs each round's exact argmax—and the candidates are symbol pairs with an offset, quadratic in the vocabulary, each exact score apparently requiring a corpus pass; at the scale of a robot pretraining corpus, enumeration is infeasible. Three mechanisms and one special case make it affordable. Counting is aggregate: identical steps collapse into one weighted record, and a candidate's exact score is the weighted intersection of two occurrence records, so counters exist only for candidates that occur. Updates are local: a merge invalidates only the counts incident to the instances it replaces. The perturbation is one-sided: every count a merge creates involves the new symbol, so surviving counts only shrink and a stale priority remains an upper bound; a lazy priority queue inspected in decreasing bound returns the exact argmax. The special case is a symbol merged with itself δ steps apart: one instance cannot serve as both endpoints, and the true gain is a maximum matching, exact under left-to-right greedy selection on the time-ordered paths the offset induces. One vectorized pass builds the counters; each round costs work proportional to the instances the committed merge touches, not to the corpus or the vocabulary squared. Algorithm 2 gives the trainer and coder, Appendix H proves losslessness, Appendix C details the machinery, and Section 7.6 isolates the order-free merges against order-dependent BPE.

6 ACTIONCODEC2

ActionCodec2 turns the preceding sections into a complete tokenizer. Fitting starts from a data audit: the bound of every action dimension is the $q_{99.9}$ quantile of the corpus, which covers household and industrial motion. The tolerance is 5 mm in translation, 0.02 rad in rotation, and 4–20 mrad across joint axes. The primitives follow from these limits in closed form. Two costs of the first-order design motivate a *second-order* variant. Acceleration limits and workspace boundaries make the feasible increments depend on state, and folding every truncation into one enlarged motion set covers maneuvers the robot never performs, inflating the count. Smooth motion also changes its increment gradually, so primitives that name bounded changes of increment, accelerations, span a narrower range than increments themselves. The second-order decoder therefore tracks a velocity state beside its position, applies primitives as bounded changes of increment, and keeps the tolerance on the trajectory error alone; Appendix F develops the scheme and supplies the relaxed grids the fit uses. Every primitive axis contains an exact zero, so stillness is encoded exactly. Encoding proceeds step by step: each step selects the primitive that best corrects the current reconstruction error, and Set-BPE (Algorithm 2) learns the vocabulary over the resulting index streams. Nothing is trained by gradient descent; a full fit is minutes of counting and merging on one CPU node. Two profiles suffice: end-effector pose plus a binary gripper, and joint positions plus a binary gripper. Dual arms pass through the same processor as single arms, so the vocabulary does not depend on the embodiment.

The universal tokenizers are fitted once, for both profiles and both orders, on roughly 20k hours of data from 24 embodiments [27, 14, 1]: 105M end-effector actions over 146k trajectories and 204M joint actions over 233k. The experiments of Section 7 evaluate them as the pretrained variants



Figure 5: **Example rollouts of ActionCodec2 on the self-designed DROID-Franka tasks.**

ActionCodec2+ and **ActionCodec2+**^{II}, while plain ActionCodec2 names the corpus-fit instances refit on each benchmark. The second order needs fewer atoms than the first: 86 versus 176 (end-effector) and 273 versus 353 (joint). Both profiles end with 4,096 tokens and cost 8.8 and 10.8 tokens per second of action (first order), respectively. On held-out trajectories the worst errors stay well within the declared bounds, and every stream round-trips bit-exactly.

7 Experiments

We evaluate **ActionCodec2** as a drop-in action tokenizer and organize the study around four questions: (i) does a streaming tokenizer match chunkwise tokenizers on simulation benchmarks, both in distribution and out; (ii) does it help when the policy is improved by reinforcement learning; (iii) does the advantage survive deployment on a real robot in the wild; and (iv) how much of the closed-loop gain is attributable to streaming decoding under a real-time control loop and to the order-free Set-BPE packing. Every tokenizer is integrated into a pretrained AR VLA, G0.5 [23]. None is G0.5's pre-training tokenizer, so all are fine-tuned under an identical recipe. We also report a separate case study of direct VLM agent control through ActionCodec2 tokens.

7.1 Setup

Tokenizers. We abbreviate ActionCodec2 as **ACodec2** and the chunkwise tokenizer ActionCodec [9] as **ACodec**; a "+" suffix marks the universal pretrained instance of Section 6, and **ACodec2**^{II} denotes the second-order variant. We compare against advanced action tokenizers: **FAST** [29] is a rule-based universal tokenizer in the frequency domain and the most widely adopted baseline; **X-Tokenizer** [13] is a VQ-based action tokenizer released by XSquare and pretrained on 2.0B action frames; **OAT** [20] and **ACodec** are VQ-based action tokenizers built on Perceiver-style architectures; OAT is evaluated at its official token budget ($\mathrm{OAT}_{\mathrm{official}}$) and at one matched to ACodec2's average compression rate ($\mathrm{OAT}_{\mathrm{aligned}}$). See Appendix D.1 for more details.

Benchmarks. Our experiments cover four benchmarks: LIBERO, LIBERO-Plus, SimplerEnv, and DROID-Franka. LIBERO [19] probes goal reaching, spatial reasoning, object recognition, and long-horizon execution across four task suites. LIBERO-Plus [10] adds seven categories of perturbation, under which policies usually suffer drastic drops. SimplerEnv [18] is a real-to-sim task suite: policies are trained on BridgeData [37] and evaluated zero-shot on four WidowX simulation tasks. For DROID-Franka we train on DROID [14] and then evaluate zero-shot, in the wild, on four groups of test tasks of increasing difficulty: color-discriminated pick-and-place, snack storing and serving, nested-cup arrangement, and towel unfolding. All evaluations except LIBERO are zero-shot generalization tests. See Appendix D.2 for more details.

7.2 Main results on simulation benchmarks

The ACodec2 family attains the best average score on LIBERO, LIBERO-Plus and SimplerEnv (Table 1), on a par with the strongest VLAs and WAMs [23] (full breakdowns in Appendix D.2). The pretrained ACodec2+ leads the corpus-fit ACodec2 on all OOD suites, plausibly because pretraining keeps its commands conservative under distribution shift, limiting drift during rollouts: on LIBERO-Plus, rotations $\leq 1^{\circ}$ account for 42% of ACodec2+'s commands versus 16% for ACodec2; on SimplerEnv, displacements $\leq 1\,\mathrm{mm}$ account for 18% versus 0%. The second-order variants gain most under robot-initial-state perturbations on LIBERO-Plus (Table 4).

7.3 ActionCodec2 under reinforcement learning

We run PPO [32], the most widely used RL algorithm for AR VLAs [11, 41, 21], on a hardened LIBERO-Object suite where every episode re-places all objects and the target basket at random anywhere. All policies are cold-started on standard LIBERO until their initial success rate on the

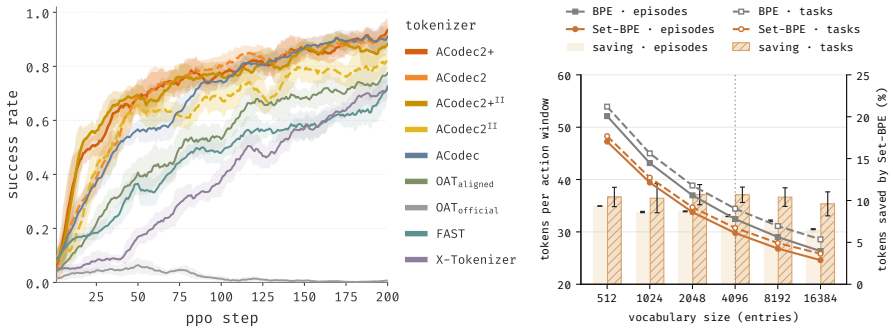


Figure 6: **PPO learning curves (left) and packing compression (right).** Left: hardened LIBERO-Object, with objects and basket re-placed at random; policies are cold-started to about 5% success, then trained for 200 PPO steps of 320 rollouts. Solid curves show the rollout success rate as a rolling mean, shaded bands ± one rolling standard deviation. Right: lines show mean tokens per action window, bars the token saving over BPE; *unseen episodes* and *unseen tasks* are held out from fitting; the dotted line marks the deployed 4,096-entry vocabulary.

modified suite reaches about 5%, then trained for 200 PPO steps of 320 rollouts each. As Figure 6 (left) shows, apart from ACodec2^{II}, which suffers training instability and settles near 83%, the ACodec2 variants converge to about 93%; among the baselines only ACodec catches up by the end, $\text{OAT}_{\text{aligned}}$, FAST, and X-Tokenizer stay near 70%, and $\text{OAT}_{\text{official}}$ never improves. The gap is behavioral as much as numerical: the target basket on standard LIBERO always sits on the left, and every initial policy acquires the move-left habit regardless of the actual position; the ACodec2 family sheds it early, whereas the others correct it slowly, if at all. This is likely because an ACodec2 token names an executable primitive, so a token-distribution shift changes the motion at specific moments; for chunk tokens the same shift changes a whole chunk, which slows credit assignment.

7.4 Real-robot experiments in the wild

We evaluate four long-horizon tasks with unseen scenes, goals, and camera placements. The comparison of interest is against FAST and X-Tokenizer, two baselines that also claim to be pretrained general-purpose action tokenizers: every tokenizer is used off the shelf except X-Tokenizer, additionally fine-tuned on DROID, and the corpus-fit ACodec2. Figure 8 reports task progress, the fraction of prescribed milestones attained, and Figure 10 (Appendix D.2) unpacks each suite at the milestone level: the ACodec2 family markedly outperforms FAST (50.8%) and X-Tokenizer (52.8%), and the pretrained ACodec2+ leads with 73.9% task progress. Qualitatively, streaming removes the periodic latency of chunked decoding, and the robot moves fluently and completes the tasks faster; beyond this deployment difference, the ACodec2 policies are more decisive, whereas the baselines hesitate near the goal, adjusting back and forth before the grasp or the release. These results support ACodec2+ as an off-the-shelf action tokenizer for a variety of robots. Figure 5 illustrates one rollout per suite, from the initial state through three milestones to the final state.

7.5 ActionCodec2 as an agent control interface

We also expose ActionCodec2 tokens directly to a vision–language agent (Claude Opus 5.5 [2]) controlling a simulated YAM bimanual robot [12]. In *Multi-Drawer Search*, it must search two cabinets for a measuring cup and tongs, then place each in a bin, cup first. Across 20 episodes with a fixed seed, operator-curated reflections and a growing library of named token-sequence macros are passed to the next attempt. The agent develops reusable drawer-opening, grasping, transport, and handover routines; episode 20 is the first environment-confirmed full success (Figure 7). The plotted intermediate values are maximum shaped rewards, not success rates, and some early positive values were false triggers. This evolving-agent demonstration shows how the token vocabulary can support long-horizon planning, while the transferred knowledge and changing decision budget preclude attributing the improvement to the tokenizer alone; a controlled RMBench study (Appendix E) finds that tokens cost the agent fewer decisions and less thinking than absolute pose commands.

7.6 Ablations

We ablate the two components that distinguish ACodec2 from a chunk tokenizer. (*a*) **Real-time LIBERO.** Every evaluation above pauses the environment while the policy decodes; here (Table 2, left) the client instead runs at a fixed 20 Hz wall clock, pulling actions from an asynchronous FIFO

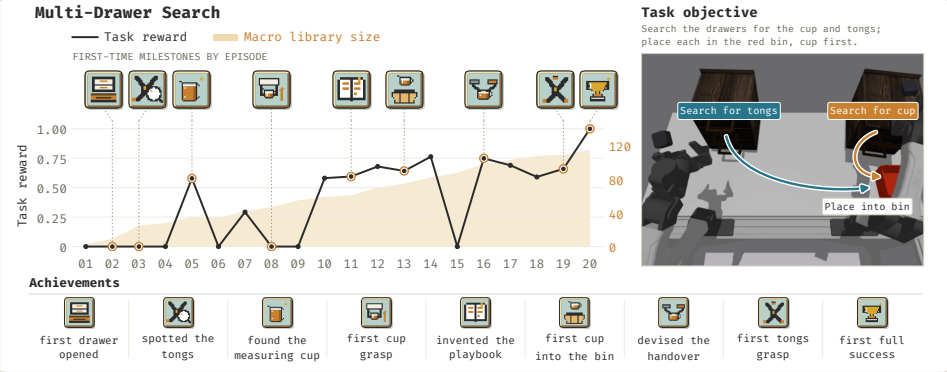


Figure 7: **ActionCodec2 as a direct agent control interface.** Claude Opus 5.5 drives a simulated YAM robot through ACodec2 tokens to search drawers for a cup and tongs and place them in a bin. The timeline shows the maximum shaped task reward per episode (not a success rate), the size of the reusable macro library, and selected first-time achievements. The first full task success occurs in episode 20.

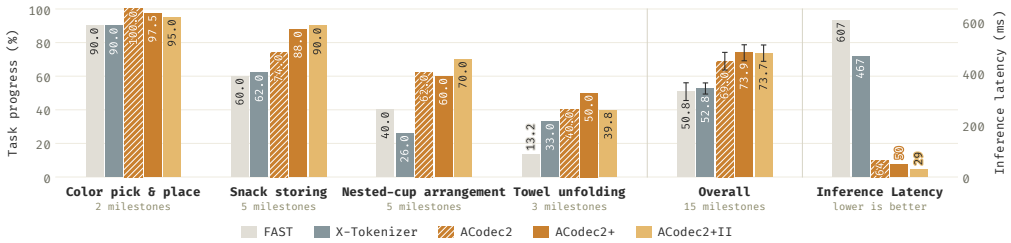


Figure 8: **DROID-Franka real-robot task progress (%).** Five policies on the four self-designed task suites, 50 rollouts per policy, scored by task progress (Appendix D.2). Policies are trained on DROID and evaluated zero-shot in unseen scenes; error bars on Overall are 90% stratified bootstrap CIs. The right panel reports per-policy inference latency (ms, right axis).

and sending zero motion whenever it runs dry. The physics engine keeps moving through each stall, so the drift left by every periodic stall of chunk execution accumulates until the policy derails; the same ACodec2+ checkpoint, released as a stream rather than after its whole chunk is decoded, degrades the least, keeping 91.0% as a stream against 73.1% when it releases whole chunks. (*b*) **Set-BPE versus BPE.** Set-BPE saves 8.0–11.9% of tokens over BPE on both unseen episodes and unseen tasks, and the margin stays nearly flat from 512 to 16,384 vocabulary entries (Figure 6, right); swapping Set-BPE for BPE on the serialized primitive stream costs 2.6 points of LIBERO success (Table 2).

Table 2: **Ablations (%): LIBERO averages.** Left: client runs on a 20 Hz wall clock and sends zero motion whenever its action queue is empty; Right: the same ACodec2 primitive stream packed by BPE or Set-BPE. Per-suite scores in Table 6. **bold** is the best within a group, underline the second best.

	Real-time control at 20 Hz						*Packing*	
	FAST *chunk*	**X-Tokenizer** *chunk*	**ACodec** *chunk*	**OAT$_{\text{aligned}}$** *chunk*	**ACodec2+** *chunk*	**ACodec2+** *streaming*	**ACodec2** *BPE*	**ACodec2** *Set-BPE*
Avg.	35.9	60.6	84.5	<u>87.2</u>	73.1	**91.0**	94.3	**96.9**(+2.6)

8 Conclusion, Limitations, and Future Work

We presented ActionCodec2, a streaming action tokenizer that casts tokenization as a covering problem on the action group: composing fixed motion primitives is optimal among decoders that read one label per step, and Set-BPE packs the primitive stream losslessly. Fitted once on 20k hours of data from 24 embodiments, the pretrained first- and second-order tokenizers set the state of the art on four benchmarks. Two limitations remain. The motion bounds of the pretrained tokenizers cap end-effector speed at about 1.2 m/s, enough for household and industrial tasks but not for high-speed motion; and on some highly dynamic sequences the compression still trails that of chunkwise tokenizers, which exploit a longer window. Finally, VLM agents that merge ActionCodec2 tokens into macros in practice build libraries that are both semantic and useful, and training such macros jointly with the VLA is a promising direction.

AI USE STATEMENT

In this work, we used generative AI tools to assist in the writing of proofs, deriving intermediate steps in the proofs of our theorems and checking those proofs for correctness, and to assist in implementing the method, writing code for designs the authors specified. We have not used generative AI tools to propose research ideas or hypotheses. Additionally, we used generative AI tools to edit the paper for readability and to create and refine scientific figures. We have reviewed all AI-assisted work: the authors verified every AI-assisted derivation step by step, reviewed and tested all AI-assisted code, and checked every figure against the underlying data. Separately, the VLM agents in Section 7.5 and Appendix E are subjects of our experiments, not research aids. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

[1] AgiBot-World-Contributors, Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xuan Hu, Xu Huang, Shu Jiang, Yuxin Jiang, Cheng Jing, Hongyang Li, Jialu Li, Chiming Liu, Yi Liu, Yuxiang Lu, Jianlan Luo, Ping Luo, Yao Mu, Yuehan Niu, Yixuan Pan, Jiangmiao Pang, Yu Qiao, Guanghui Ren, Cheng Ruan, Jiaqi Shan, Yongjian Shen, Chengshi Shi, Mingkang Shi, Modi Shi, Chonghao Sima, Jianheng Song, Huijie Wang, Wenhao Wang, Dafeng Wei, Chengen Xie, Guo Xu, Junchi Yan, Cunbiao Yang, Lei Yang, Shukai Yang, Maoqing Yao, Jia Zeng, Chi Zhang, Qinglin Zhang, Bin Zhao, Chengyue Zhao, Jiaqi Zhao, and Jianchao Zhu. AgiBot World Colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, 2025.

[2] Anthropic. System card: Claude Opus 5.5. `https://www.anthropic.com`, 2026.

[3] Homanga Bharadhwaj, Jay Vakil, Mohit Sharma, Abhinav Gupta, Shubham Tulsiani, and Vikash Kumar. RoboAgent: Generalization and efficiency in robot manipulation via semantic augmentations and action chunking. In *IEEE International Conference on Robotics and Automation, ICRA*, 2024.

[4] Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, Hongyan Zhao, Hanyu Liu, Zhizhong Su, Lei Ma, Hang Su, and Jun Zhu. Motus: A unified latent action world model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, 2026.

[5] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control. In *Robotics: Science and Systems, RSS*, 2025.

[6] Kevin Black, Manuel Y Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. In *Advances in Neural Information Processing Systems, NeurIPS*, 2025.

[7] Tianxing Chen, Yuran Wang, Mingleyang Li, Yan Qin, Hao Shi, Zixuan Li, Yifan Hu, Yingsheng Zhang, Kaixuan Wang, Yue Chen, Hongcheng Wang, Junjie Wang, Tianhang Yang, Renjing Xu, Ruihai Wu, Yao Mu, Yaodong Yang, Hao Dong, and Ping Luo. RMBench: Memory-dependent robotic manipulation benchmark with insights into policy design. *arXiv preprint arXiv:2603.01229*, 2026.

[8] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 2024.

[9] Zibin Dong, Yicheng Liu, Shiduo Zhang, Baijun Ye, Yifu Yuan, Fei Ni, Jingjing Gong, Xipeng Qiu, Hang Zhao, Yinchuan Li, and Jianye Hao. ActionCodec: What makes for good action tokenizers. In *Conference on Robot Learning, CoRL*, 2026.

[10] Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. LIBERO-Plus: A progressive robustness benchmark for visual-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, 2026.

[11] Yanjiang Guo, Jianke Zhang, Xiaoyu Chen, Xiang Ji, Yen-Jen Wang, Yucheng Hu, and Jianyu Chen. Improving vision-language-action model with online reinforcement learning. In *IEEE International Conference on Robotics and Automation, ICRA*, 2025.

[12] I2RT Robotics. YAM: Yet another manipulator. `https://doc.i2rt.com/products/yam`, 2025.

[13] Miracle Kang, Lights Shi, Lucy Liang, Roy Gan, Dongxiu Liu, Pushi Zhang, Sylas Chen, Shawn Qin, Yinan Zheng, Jinliang Zheng, Hao Wang, Xianyuan Zhan, and Hang Su. X-Tokenizer: A multimodal action tokenizer for vision-language-action pretraining. *arXiv preprint arXiv:2606.14752*, 2026.

[14] 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, RSS*, 2024.

[15] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An open-source vision-language-action model. In *Conference on Robot Learning, CoRL*, 2024.

[16] Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, Grace Lam, Percy Liang, Shuran Song, Ming-Yu Liu, Chelsea Finn, and Jinwei Gu. Cosmos policy: Fine-tuning video models for visuomotor control and planning. In *International Conference on Learning Representations, ICLR*, 2026.

[17] 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, ICML*, 2024.

[18] Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation. In *Conference on Robot Learning, CoRL*, 2024.

[19] 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, NeurIPS*, 2023.

[20] Chaoqi Liu, Xiaoshen Han, Jiawei Gao, Yue Zhao, Haonan Chen, and Yilun Du. OAT: Ordered action tokenization. In *Robotics: Science and Systems, RSS*, 2026.

[21] Jijia Liu, Feng Gao, Bingwen Wei, Xinlei Chen, Qingmin Liao, Yi Wu, Chao Yu, and Yu Wang. What can RL bring to VLA generalization? An empirical study. In *Advances in Neural Information Processing Systems, NeurIPS*, 2025.

[22] Songming Liu, Bangguo Li, Kai Ma, Lingxuan Wu, Hengkai Tan, Xiao Ouyang, Hang Su, and Jun Zhu. RDT2: Exploring the scaling limit of UMI data towards zero-shot cross-embodiment generalization. *arXiv preprint arXiv:2602.03310*, 2026.

[23] Yicheng Liu, Zibin Dong, Baijun Ye, Tianyuan Yuan, Tao Jiang, Anqi Yang, Shicheng Cao, Haonan Liu, Yue Sun, Zihan Guo, Xiao Liu, Dong Ke, Changxun Pan, Chenru Wu, Tailai Cheng, Xiaoshu Ren, Xinlei Zhang, Jianning Cui, Zijie Zhao, Haoyu Zhang, Kaiming Xu, Haodong Yang, Bowen Zhang, Jiahui Niu, Shaoting Zhu, Shiduo Zhang, and Hang Zhao. G0.5: One autoregressive stream for robot reasoning and action. *arXiv preprint arXiv:2608.11739*, 2026.

[24] Yicheng Liu, Shiduo Zhang, Zibin Dong, Baijun Ye, Tianyuan Yuan, Xiaopeng Yu, Linqi Yin, Chenhao Lu, Junhao Shi, Luca Jiang-Tao Yu, Liangtao Zheng, Tao Jiang, Jingjing Gong, Xipeng Qiu, and Hang Zhao. FASTer: Toward powerful and efficient autoregressive vision–language–action models with learnable action tokenizer and block-wise decoding. In *International Conference on Learning Representations, ICLR*, 2026.

[25] 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, NeurIPS*, 2024.

[26] NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. GR00T N1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

[27] Open X-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, et al. Open X-Embodiment: Robotic learning datasets and RT-X models. In *IEEE International Conference on Robotics and Automation, ICRA*, 2024.

[28] OpenAI. GPT-6 Astra system card. `https://deploymentsafety.openai.com/gpt-6-astra`, 2026.

[29] 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, RSS*, 2025.

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

[31] Physical Intelligence, Ali Amin, Raichelle Aniceto, Ashwin Balakrishna, Kevin Black, Ken Conley, Grace Connors, James Darpinian, Karan Dhabalia, Jared DiCarlo, Danny Driess, Michael Equi, Adnan Esmail, Yunhao Fang, Chelsea Finn, Catherine Glossop, Thomas Godden, Ivan Goryachev, Lachy Groom, Hunter Hancock, Karol Hausman, Gashon Hussein, Brian Ichter, Szymon Jakubczak, Rowan Jen, Tim Jones, Ben Katz, Liyiming Ke, Chandra Kuchi, Marinda Lamb, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Yao Lu, Vishnu Mano, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Charvi Sharma, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, Will Stoeckle, Alex Swerdlow, James Tanner, Marcel Torne, Quan Vuong, Anna Walling, Haohuan Wang, Blake Williams, Sukwon Yoo, Lili Yu, Ury Zhilinsky, and Zhiyuan Zhou. $\pi^*_{0.6}$: A VLA that learns from experience. In *Robotics: Science and Systems, RSS*, 2026.

[32] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[33] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, ACL*, 2016.

[34] Modi Shi, Li Chen, Jin Chen, Yuxiang Lu, Chiming Liu, Guanghui Ren, Ping Luo, Di Huang, Maoqing Yao, and Hongyang Li. Is diversity all you need for scalable robotic manipulation? *IEEE Transactions on Robotics*, 2026.

[35] Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei A. Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International Conference on Machine Learning, ICML*, 2020.

[36] Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems, NeurIPS*, 2017.

[37] Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, Vivek Myers, Kuan Fang, Chelsea Finn, and Sergey Levine. BridgeData V2: A dataset for robot learning at scale. In *Conference on Robot Learning, CoRL*, 2023.

[38] Yating Wang, Haoyi Zhu, Mingyu Liu, Jiange Yang, Hao-Shu Fang, and Tong He. VQ-VLA: Improving vision-language-action models via scaling vector-quantized action tokenizers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision, ICCV*, 2025.

[39] Yuran Wang, Siqiao Huang, Mingleyang Li, Chenhao Zhang, Jiaqi Liang, Weiyang Jin, Yue Chen, Xuemin Chi, Donghao Zhou, Qize Yu, Yu-Kai Wang, Yuhan Rui, Shenzhe Yao, Zhen Yuan, Zhenhao Shen, Kefei Zhu, Zijie Zhu, Ning Gao, Xiaowei Chi, Guanqi He, Shanghang Zhang, Hao Dong, Lin Shao, and Hang Zhao. OpenWAM: An open, modular exploration towards systematic world-action model pretraining. *arXiv preprint arXiv:2609.07398*, 2026.

[40] Xiaomi Robotics Team, Jun Guo, Piaopiao Jin, Jason Li, Peiyan Li, Yingyan Li, Futeng Liu, Wanli Peng, Optimus Qin, Yifei Su, Nan Sun, Qiao Sun, Runze Suo, Heyun Wang, Yunhong Wang, Rujie Wu, Caoyu Xia, Lina Zhang, Jack Zhao, Guoliang Chen, Wenlong Chen, Xinze He, Bin Li, Qing Li, Zhuorong Li, Heng Qu, Wenxuan Song, Diyun Xiang, Yifan Xie, Peiran Xu, Hangjun Ye, Wen Ye, Han Zhao, and Quanyun Zhou. Xiaomi-Robotics-1: Scaling vision-language-action models with over 100K hours of real-world trajectories. *arXiv preprint arXiv:2607.15330*, 2026.

[41] Charles Xu, Qiyang Li, Jianlan Luo, and Sergey Levine. Robotic generalist policy distillation via reinforcement learning. In *Robotics: Science and Systems, RSS*, 2025.

[42] Chao Yu, Yuanqing Wang, Zhen Guo, Hao Lin, Si Xu, Hongzhi Zang, Quanlu Zhang, Yongji Wu, Chunyang Zhu, Junhao Hu, Zixiao Huang, Mingjie Wei, Yuqing Xie, Ke Yang, Bo Dai, Zhexuan Xu, Jiakun Du, Xiangyuan Wang, Xu Fu, Letong Shi, Zhihao Liu, Kang Chen, Weilin Liu, Gang Liu, Boxun Li, Jianlei Yang, Zhi Yang, Guohao Dai, and Yu Wang. RLinf: Flexible and efficient large-scale reinforcement learning via macro-to-micro flow transformation. In *USENIX Symposium on Operating Systems Design and Implementation, OSDI*, 2026.

[43] Ryan Yu, Pushi Zhang, Starrick Liu, Brae Liu, Miracle Kang, Shalfun Li, Lights Shi, Ellie Ma, Ping Yang, Chris Pan, Jerry Chen, Dongxiu Liu, Rain Sun, Miles Guo, Byron Zhang, Hugo Zhou, Zach Xu, Vincent Chen, Harrison Huang, James Wang, Dance Kuzi, Andy Zhai, Hang Su, Roy Gan, Lucy Liang, Hao Wang, and Qian Wang. Wall-OSS-0.5 technical report. *arXiv preprint arXiv:2605.30877*, 2026.

[44] Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-WAM: Do world action models need test-time future imagination? In *Advances in Neural Information Processing Systems, NeurIPS*, 2026.

[45] Andy Zhai, Brae Liu, Bruno Fang, Chalse Cai, Ellie Ma, Ethan Yin, Hao Wang, Hugo Zhou, James Wang, Lights Shi, Lucy Liang, Make Wang, Qian Wang, Roy Gan, Ryan Yu, Shalfun Li, Starrick Liu, Sylas Chen, Vincent Chen, and Zach Xu. Igniting VLMs toward the embodied space. *arXiv preprint arXiv:2509.11766*, 2025.

[46] Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, Ankur Handa, Ming-Yu Liu, Donglai Xiang, Gordon Wetzstein, and Tsung-Yi Lin. CoT-VLA: Visual chain-of-thought reasoning for vision-language-action models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, 2025.

[47] Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Robotics: Science and Systems, RSS*, 2023.

[48] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems, NeurIPS*, 2024.

[49] Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, Zheyan Luo, Zhangchi Feng, and Yongqiang Ma. LlamaFactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics, ACL*, 2024.

[50] Hongyi Zhou, Weiran Liao, Xi Huang, Yucheng Tang, Fabian Otto, Xiaogang Jia, Xinkai Jiang, Simon Hilber, Ge Li, Qian Wang, Ömer Erdinç Yağmurlu, Nils Blank, Moritz Reuss, and Rudolf Lioutikov. BEAST: Efficient tokenization of B-Splines encoded action sequences for imitation learning. In *Advances in Neural Information Processing Systems, NeurIPS*, 2025.

[51] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. RT-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning, CoRL*, 2023.

A Related Work

Robot foundation models. Two families of large models are used for general-purpose robot control. Vision–language–action (VLA) models are built on vision–language models [51, 15, 5, 30, 26], and world–action models (WAMs) are built on video generation models [4, 44, 16]. Recent studies find that the training data affect performance more than the choice between the two [39, 40, 34]. VLAs further differ in how they output actions. Some attach a separate diffusion or flow-matching action expert to the vision–language model [5, 30, 26]. Others, which we call discrete VLAs, convert actions into tokens and predict them with the same autoregressive head that produces text [51, 15, 29, 22, 45, 43, 23]. Because a discrete VLA is an ordinary language model, it can be trained and served with existing LLM infrastructure [49, 42, 48], trains faster than a VLA with a flow-matching expert [24, 31], and supports chain-of-thought reasoning without modification [46]. How well a discrete VLA trains and acts depends largely on its action tokenizer [29, 9].

Action tokenizers. Among existing action tokenizers, only the per-dimension binning used by RT-2 and OpenVLA [51, 15] is streaming, since each action is decoded from its own tokens. It spends one token on every action dimension, however, so the token sequences are long and training is inefficient [29]. All other tokenizers encode an action chunk of one to two seconds jointly. FAST [29] applies a discrete cosine transform to the chunk and compresses the quantized coefficients with BPE. BEAST [50] fits the chunk with B-splines and quantizes the control points. VQ-VAE-based tokenizers [17, 25, 38, 24, 20] learn an encoder that maps the chunk to a few discrete codes, and ActionCodec [9] and X-Tokenizer [13] add training objectives to this design that make the tokens easier for a VLA to predict. Because these tokenizers decode a chunk only after all of its tokens are generated, the policy can react to new observations only at chunk boundaries. Real-time chunking [6] decodes the next chunk while the current one is executed, which removes the waiting time but not this restriction. Moreover, none of these tokenizers bounds the error that accumulates along the decoded trajectory. ActionCodec2 differs in both respects. It is streaming, and it keeps the trajectory error within a user-specified tolerance at every step while using few tokens.

B Proofs

Each statement is restated here before its proof; the theorem number in the body links to the restatement, and the number in the restatement links back.

Lemma 1 (One step suffices), restated. *If the full tolerance requirement* (2) *holds, there is a causal encoding policy $z_t = \pi_t(q_{0:t+1})$ under which the decoder keeps $\xi_t \in \mathcal{E}$ for every t, from every exact start and on every feasible trajectory of any length.*

Proof. Set $\hat{q}_0 = q_0$, so $(q_0, \hat{q}_0) \in \mathcal{S}$ because $\mathcal{E}$ contains the identity. If $(q_t, \hat{q}_t) \in \mathcal{S}$ and the observed successor is $q_{t+1} \in \mathcal{F}(q_t)$, then (2) supplies a label z_t with $\big(q_{t+1}, y_{z_t}(\hat{q}_t)\big) \in \mathcal{S}$, which by (1) is $(q_{t+1}, \hat{q}_{t+1}) \in \mathcal{S}$. The choice uses only $(q_t, \hat{q}_t)$ and q_{t+1}, and $\hat{q}_t$ is itself a function of $q_{0:t}$, so the policy has the causal form $z_t = \pi_t(q_{0:t+1})$. Induction gives $\xi_t \in \mathcal{E}$ for every t. □

Theorem 2 (Optimality of composition), restated. *Under the full tolerance requirement, (1) the composition decoder with list $\mathcal{U}$ meets it if and only if $\mathcal{EA} \subseteq \bigcup_{u\in\mathcal{U}} u\mathcal{E}$; (2) every decoder of the general form* (1) *that meets it uses at least $\min\{|\mathcal{U}| : \mathcal{U} \subseteq G \text{ finite},\ \mathcal{EA} \subseteq \bigcup_{u\in\mathcal{U}} u\mathcal{E}\}$ labels, the minimum over an empty family read as $+\infty$; (3) hence $K^\star$ equals that minimum, and composition attains it whenever it is finite. (4) All three survive a fixed distortion of the error between steps: if $q_{t+1} = \varphi(q_t)\delta_t$ for an automorphism φ of G and the decoder composes after it, $y_k(\hat{q}) = \varphi(\hat{q})u_k$, then $\xi_{t+1} = u^{-1}\varphi(\xi_t)\delta_t$ and (1)–(3) hold with $\mathcal{EA}$ replaced by $\varphi(\mathcal{E})\mathcal{A}$.*

Proof. **Part 1.** Under composition the error recursion closes on itself: with $\xi_t = \hat{q}_t^{-1} q_t$ and $q_{t+1} = q_t\delta_t$,

$$\xi_{t+1} = (\hat{q}_t u_{z_t})^{-1}(q_t\delta_t) = u_{z_t}^{-1}\xi_t\delta_t, \tag{6}$$

in which the reconstructed configuration has cancelled, so the next error depends on the current error and the realized motion alone. Hence u answers the pair (ξ, δ) iff $u^{-1}\xi\delta \in \mathcal{E}$, i.e. iff $\xi\delta \in u\mathcal{E}$.

A pair in $\mathcal{S}$ is $q = \hat{q}\xi$ with $\xi \in \mathcal{E}$, and by the motion model its successors are $q^+ = \hat{q}\xi\delta$ with $\delta \in \mathcal{A}$; by the previous paragraph the pair reached after executing u lies in $\mathcal{S}$ iff $\xi\delta \in u\mathcal{E}$. Quantifying over all $\xi \in \mathcal{E}$ and all $\delta \in \mathcal{A}$, the full tolerance requirement (2) is therefore exactly $\mathcal{E}\mathcal{A} \subseteq \bigcup_{u\in\mathcal{U}} u\mathcal{E}$, and Lemma 1 turns it into tracking over an infinite horizon. Conversely, if that inclusion fails there are $\xi \in \mathcal{E}$ and $\delta \in \mathcal{A}$ that no primitive corrects, so (2) fails at every pair carrying error ξ.

Part 2. Fix any reconstruction $\hat{q}$ and set $c_k := \hat{q}^{-1} y_k(\hat{q})$, so that $y_k(\hat{q}) = \hat{q}c_k$. This change of coordinates is available at every $\hat{q}$ because G is a group, and it assumes nothing about the maps: at a single configuration, every decoder (1) *is* a composition decoder, with a list $c_1, \ldots, c_K$ that may depend on $\hat{q}$. The content of Part 2 is that the dependence is worthless.

Take any $\xi \in \mathcal{E}$, so the real configuration is $q = \hat{q}\xi$, and any $\delta \in \mathcal{A}$, so $q^+ = \hat{q}\xi\delta$. If label k is chosen, the next error is $y_k(\hat{q})^{-1} q^+ = (\hat{q}c_k)^{-1}(\hat{q}\xi\delta) = c_k^{-1}\xi\delta$, in which $\hat{q}$ cancels as in (6), and it is acceptable precisely when $\xi\delta \in c_k\mathcal{E}$. Because the tolerance is complete, every $\xi \in \mathcal{E}$ and every $\delta \in \mathcal{A}$ must be handled at this same $\hat{q}$, so

$$\mathcal{E}\mathcal{A} \subseteq \bigcup_{k=1}^{K} c_k \mathcal{E}. \tag{7}$$

The left-hand side does not depend on $\hat{q}$: every reconstruction faces one and the same covering problem, which is why adapting the list to $\hat{q}$ gains nothing. The set $\{c_1, \ldots, c_K\}$ satisfies $\mathcal{E}\mathcal{A} \subseteq \bigcup_{k=1}^{K} c_k\mathcal{E}$, so K is at least the stated minimum.

Part 3. Take $\mathcal{U}$ of minimum size with $\mathcal{E}\mathcal{A} \subseteq \bigcup_{u\in\mathcal{U}} u\mathcal{E}$. Part 1 shows the decoder it builds meets the full tolerance requirement, Part 2 shows no such decoder is smaller.

Part 4. Only two steps above used $\varphi = \mathrm{id}$, and both survive because φ is a homomorphism. In Part 1, $\xi_{t+1} = (\varphi(\hat{q}_t)u_{z_t})^{-1}\varphi(q_t)\delta_t = u_{z_t}^{-1}\varphi(\hat{q}_t^{-1}q_t)\delta_t = u_{z_t}^{-1}\varphi(\xi_t)\delta_t$, so $\hat{q}_t$ still cancels and u answers (ξ, δ) iff $\varphi(\xi)\delta \in u\mathcal{E}$; quantifying over $\mathcal{E}$ and $\mathcal{A}$ replaces $\mathcal{E}\mathcal{A}$ by $\varphi(\mathcal{E})\mathcal{A}$. In Part 2, set $c_k := \varphi(\hat{q})^{-1} y_k(\hat{q})$ instead, which is again available at every $\hat{q}$, and the same cancellation gives $\varphi(\mathcal{E})\mathcal{A} \subseteq \bigcup_k c_k\mathcal{E}$ at every reconstruction. Part 3 is unchanged. The set being covered still does not depend on $\hat{q}$, which is the whole content of the lower bound. □

Corollary 3 (Translation), restated. *Let $G = (\mathbb{R}^d, +)$ with $\underline{a}_j \le \overline{a}_j$, box motion set $\mathcal{A} = \prod_j [\underline{a}_j, \overline{a}_j]$ with $\Delta a_j := \overline{a}_j - \underline{a}_j$, and tolerance $\mathcal{E} = \prod_j [-\epsilon_j, \epsilon_j]$ with $\epsilon_j > 0$. Then $\mathcal{E}\mathcal{A} \subseteq \bigcup_{u\in\mathcal{U}} u\mathcal{E}$ reads $\mathcal{E} + \mathcal{A} \subseteq \bigcup_{u\in\mathcal{U}} (u + \mathcal{E})$, and the smallest alphabet meeting the full tolerance requirement has size $K^\star_{\mathrm{trans}} = \prod_j N_j$ with $N_j = \lceil 1 + \Delta a_j / 2\epsilon_j \rceil$, attained by the axiswise uniform grid* (4) *of Algorithm 1. If the real trajectory stays in a workspace box, clipping each reconstructed successor to that box preserves the error guarantee.*

Proof. The claim is about the decoder class (1), not about composition alone, so start from Theorem 2: by Parts 2 and 3, every decoder meeting the full tolerance requirement supplies a finite $\{c_1, \ldots, c_K\} \subseteq \mathbb{R}^d$ satisfying $\mathcal{E} + \mathcal{A} \subseteq \bigcup_k (c_k + \mathcal{E})$, and $K^\star_{\mathrm{trans}}$ is the least cardinality of such a set. It is therefore enough to bound arbitrary translate covers. Here $\xi_{t+1} = \xi_t + \delta_t - u$, so the set to be covered is the box $\prod_j [\underline{a}_j - \epsilon_j, \overline{a}_j + \epsilon_j]$, of side $L_j = \Delta a_j + 2\epsilon_j$, and $N_j = \lceil L_j / 2\epsilon_j \rceil$. Take one point in the jth side if $N_j = 1$ and N_j equally spaced points spanning it otherwise; these are more than $2\epsilon_j$ apart, since $N_j - 1 < L_j / 2\epsilon_j$, and a single translate $u + \mathcal{E}$ spans at most $2\epsilon_j$ per axis, so it contains at most one of the $\prod_j N_j$ product points: no cover, product or not, is smaller. Conversely the points of (4) are at most $2\epsilon_j$ apart and start and end at $\underline{a}_j$ and $\overline{a}_j$, so their ϵ_j-intervals tile the jth side without a gap. Clipping toward a target inside the workspace cannot increase any coordinate error, so the cover survives it, and the count stays necessary as long as every axis of the workspace is wider than $\Delta a_j + 2\epsilon_j$, so that the separated points above still fit inside it. □

Corollary 4 (Orientation), restated. *Let $G = \mathrm{SO}(3)$ with body-frame increments, error measured by the intrinsic angle $\theta(Q) = \arccos\big((\mathrm{tr}(Q) - 1)/2\big)$, and $\mathcal{B}_R(r) = \{Q : \theta(Q) \le r\}$, with tolerance $\mathcal{E} = \mathcal{B}_R(\epsilon_R)$ and motions $\mathcal{A}_R = \mathcal{B}_R(b_R)$ of any axis, $0 < \epsilon_R \le \pi$ and $0 \le b_R \le \pi$. Then $\mathcal{E}\mathcal{A} \subseteq \bigcup_{u\in\mathcal{U}} u\mathcal{E}$ reads $\mathcal{B}_R(\epsilon_R)\mathcal{A}_R \subseteq \bigcup_{k=1}^{K} U_k \mathcal{B}_R(\epsilon_R)$, and with $\rho_R = \min\{\pi, \epsilon_R + b_R\}$, $\beta_R = b_R/\epsilon_R$ and $V_R(r) = (r - \sin r)/\pi$, the smallest such alphabet obeys* (5)*, so that $K^\star_{\mathrm{rot}} = \Theta((1 + \beta_R)^3)$*

whenever $\rho_R + \epsilon_R/2 \le r_0 < \pi$. Algorithm 1, run with grid step $0 < h < 2\epsilon_R/\sqrt{3}$, returns a list satisfying that inclusion, not claimed to be minimal.

Proof. The bi-invariant angle metric gives $\mathcal{B}_R(\epsilon_R)\mathcal{B}_R(b_R) = \mathcal{B}_R(\rho_R)$: the triangle inequality one way, and splitting a rotation of angle at most ρ_R about its own axis the other. So $\mathcal{B}_R(\rho_R)$ is the set to be covered by K translates of volume $V_R(\epsilon_R)$, which gives the lower bound. For the upper bound take a maximal ϵ_R-separated set in $\mathcal{B}_R(\rho_R)$; maximality makes its ϵ_R-balls a cover, and its disjoint $\epsilon_R/2$-balls all lie in $\mathcal{B}_R(\min\{\pi, \rho_R + \epsilon_R/2\})$. Finally $V_R(r) = \int_0^r \sin^2(\vartheta/2)\, d\vartheta \big/ \int_0^\pi \sin^2(\vartheta/2)\, d\vartheta$ is bounded above and below by constants times r^3 on $[0, r_0]$, and $\rho_R/\epsilon_R = 1 + \beta_R$. For the last claim, $\theta(\exp v) = \|v\|$ and $\mathrm{SO}(3)$ carries nonnegative curvature, so exp is 1-Lipschitz, $d(\exp u, \exp v) \le \|u - v\|$; a Euclidean covering of $\{\|v\| \le \rho_R\}$ by radius-ϵ_R balls therefore transfers to the body-frame cover $\mathcal{B}_R(\epsilon_R)\mathcal{A}_R \subseteq \bigcup_k U_k \mathcal{B}_R(\epsilon_R)$ with no slack, and the certificate in Algorithm 1 is exact because a grid of spacing h leaves every point of the ball within $h\sqrt{3}/2$ of a witness. A list larger than $K^\star_{\mathrm{rot}}$ still meets the full tolerance requirement by Theorem 2, so the excess is spent in bits and not in error. □

Proposition 5 (Frame equivalence), restated. *Let increments act on the left, $R_{t+1} = \Delta_t^w R_t$ and $\hat{R}_{t+1} = U\hat{R}_t$, so that the error is $\Xi_t^w = R_t\hat{R}_t^\top$ and the covering condition is $\mathcal{A}_R^w \mathcal{B}_R(\epsilon_R) \subseteq \bigcup_k \mathcal{B}_R(\epsilon_R) U_k$. Its minimum equals the body-frame $K^\star_{\mathrm{rot}}$ for the motion set $(\mathcal{A}_R^w)^\top$. The two coincide whenever $\mathcal{A}_R^w$ is closed under inversion, a geodesic ball in particular.*

Proof. Transposition is a bijection of $\mathrm{SO}(3)$ that reverses products and preserves the intrinsic angle, so $\mathcal{B}_R(\epsilon_R)^\top = \mathcal{B}_R(\epsilon_R)$; applying it termwise turns the world-frame inclusion into the body-frame cover for $(\mathcal{A}_R^w)^\top$ with primitives $U_k^\top$, a size-preserving correspondence of the two feasible sets. □

C Algorithms

Algorithm 1 builds the primitive lists of Section 4: a closed-form axiswise grid for translation, and for orientation a greedy cover of the reachable increment ball refined by a local move that recenters each ball on the region it alone covers. Encode selects a label at run time by nearest reconstruction, using $d(x, y) = \max_j |x_j - y_j|/\epsilon_j$ for translation and $d(R, S) = \theta(S^\top R)/\epsilon_R$ for orientation. Rotation uses $[v]_\times x = v \times x$; its output may be stored as unit quaternions $(\cos(\alpha/2), \sin(\alpha/2)v/\alpha)$, where $\alpha = \|v\|$, using $(1, 0, 0, 0)$ at $v = 0$.

Algorithm 2 is the Set-BPE fitter and streaming coder of Section 5: Fit runs once over the steps for the spatial vocabulary and once over their token instances for the temporal vocabulary, and the coder applies the resulting merges in creation order.

Exact greedy training without a dense candidate table. Algorithm 2 states the greedy rule over all legal candidates; the lazy loop certifies the same winner without enumerating them, with the aggregate counting, local updates, one-sided perturbation, and self-rule matching of Section 5. A few details make the certificate exact in practice. A spatial pair's exact score is the weighted intersection of the occurrence records of its two symbols, (25), and a temporal candidate (P, R, δ) occurs only where a start of P is followed by a start of R at offset δ, so intersecting shifted start sets yields exactly its instance pairs. Every unexamined candidate carries a stored bound $U \ge F$ on its current score, and version numbers keep a bound computed before an earlier merge from being reused after the records supporting it change; a candidate wins once its exact score matches or exceeds every remaining bound, with the tie-breaking of exhaustive enumeration. The reference implementation and an optimized native kernel are kept merge-for-merge identical: both maintain exact counters over observed events only, break ties by count and then by the smallest symbol pair, and apply merges in creation order at encode time, so the trained vocabulary is reproducible and independent of the backend. Legality of a temporal candidate is decided from per-symbol time-shifted support masks in time proportional to the span, so no table over the vocabulary is materialized—sparse and lazy enumeration bound memory, not time.

Algorithm 1 Primitive lists for $(\mathbb{R}^d, +)$ and $\mathrm{SO}(3)$: uniform binning for translation, greedy ball cover for rotation

procedure TRANSLATION($\underline{a}, \overline{a}, \epsilon$)
Require: $\underline{a}_j \le \overline{a}_j, \epsilon_j > 0$ for all j
  for $j = 1, \dots, d$ **do**
    $\Delta a_j \leftarrow \overline{a}_j - \underline{a}_j$ # one-step motion range, e.g. EEF translation per step
    $N_j \leftarrow \lceil 1 + \Delta a_j / (2\epsilon_j) \rceil$ # fewest bins of half-width ϵ_j
    $\mathcal{U}_j \leftarrow \{\underline{a}_j\}$
    if $N_j > 1$ **then**
      $\mathcal{U}_j \leftarrow \{\underline{a}_j + \frac{\Delta a_j (i-1)}{N_j - 1}\}_{i=1}^{N_j}$ # uniform binning
  return $\mathcal{U}_1 \times \cdots \times \mathcal{U}_d$
procedure ROTATION(ϵ_R, b_R, h)
Require: $0 < \epsilon_R \le \pi, 0 \le b_R \le \pi, 0 < h < 2\epsilon_R/\sqrt{3}$
  Work in axis-angle coordinates v, where tolerance and motion are balls.
  $\rho_R \leftarrow \min\{\pi, \epsilon_R + b_R\}$ # farthest the error can drift in one step
  $r \leftarrow \epsilon_R - h\sqrt{3}/2$
  $W \leftarrow \{v \in h\mathbb{Z}^3 : \|v\| \le \rho_R + h\sqrt{3}/2\}$ # the space to cover
  $C \leftarrow \{v \in \frac{h}{2}\mathbb{Z}^3 : \|v\| \le \rho_R + \epsilon_R\}$
  $B(v) := \{w \in W : \|w - v\| \le r\}$
  $\mathcal{V} \leftarrow \emptyset,\ U \leftarrow W$
  while $U \neq \emptyset$ **do** # greedy set cover
    $c \leftarrow \arg\max_{c \in C} |U \cap B(c)|$
    $\mathcal{V} \leftarrow \mathcal{V} \cup \{c\},\ U \leftarrow U \setminus B(c)$
  repeat # polish: re-center each ball on the region it alone covers
    $n \leftarrow |\mathcal{V}|$
    for all v in a snapshot of $\mathcal{V}$ **do**
      $W_v \leftarrow B(v) \setminus \bigcup_{c \in \mathcal{V} \setminus \{v\}} B(c)$
      if $W_v = \emptyset$ **then**
        $\mathcal{V} \leftarrow \mathcal{V} \setminus \{v\}$
      else
        $c \leftarrow$ center of the smallest ball enclosing W_v
        $\mathcal{V} \leftarrow (\mathcal{V} \setminus \{v\}) \cup \{c\}$
  until $|\mathcal{V}| = n$ # no ball dropped in this pass
  assert $W \subseteq \bigcup_{v \in \mathcal{V}} B(v)$
  return $\{\exp([v]_\times) : v \in \mathcal{V}\}$
procedure ENCODE($\hat{q}_t, q_{t+1}, \mathcal{U}$)
  return $\arg\min_k\ d\big(q_{t+1}, \hat{q}_t u_k\big)$

Table 3: **Qualitative comparison of the evaluated action tokenizers.** ● = attribute applies; – = not applicable; ✓ = yes; × = no. *Lossless* refers to the compression stage only: quantization always loses. *Accuracy guarantee*: a worst-case tracking-error bound on every action sequence. *Pretrained*: once on a large multi-embodiment corpus. *Held-out universal*: zero-shot on unseen corpora or embodiments.

Tokenizer	Neural	Rule-based	Lossless	Acc. guar.	Streaming	Pretrained	Held-out univ.
FAST [29]	–	●	✓	✓	×	✓	✓
X-Tokenizer [13]	●	–	×	×	×	✓	×
OAT [20]	●	–	×	×	×	×	×
ACodec [9]	●	–	×	×	×	×	×
ACodec2 (ours, corpus-fit)	–	●	✓	✓	✓	×	×
ACodec2+ (ours, universal)	–	●	✓	✓	✓	✓	✓

D EXPERIMENTAL SETUP

D.1 TOKENIZERS

Table 3 contrasts the compared tokenizers along the design axes that Sections 2–5 argue matter; the paragraphs below give per-method details.

Algorithm 2 Set-BPE: exact greedy fitting and streaming coding. Exp expands a spatial token to its atoms and Exp_T a temporal token to its (offset, spatial token) pairs; a is the start step of an instance in ENCODE and the earliest incomplete step in DECODE, and C_0 is the root.

procedure FIT(weighted steps $\{(r, w)\}$, span τ, budget V)
  counters $\leftarrow$ exact scores of observed candidates # one vectorized pass; spatial first, then temporal
  heap $\leftarrow$ observed candidates keyed by score U
  while the vocabulary is below budget **do**
    $(c, U) \leftarrow$ POPMAX(heap)
    $f \leftarrow$ exact score of c # occurrence-record intersection; greedy matching for a self-rule
    if $f < U$ **then**
      push (c, f); **continue** # stale bound: requeue
    commit c: replace the instances it fuses, update the incident counters
procedure ENCODE($S_{1:T}$)
  apply the spatial then the temporal merges in training order
  return the instances sorted by $\big(a, \min \mathrm{supp}\, C_0\big)$
procedure DECODE(token stream)
  $a \leftarrow 1$; $\hat{S}_t \leftarrow \emptyset$ for every t
  for all Q in stream order **do**
    while $|\hat{S}_a| = D$ **do**
      emit q_a; $a \leftarrow a + 1$
    $\hat{S}_{a+\delta} \leftarrow \hat{S}_{a+\delta} \uplus \mathrm{Exp}(C)$ for $(\delta, C) \in \mathrm{Exp}_T(Q)$
  emit the remaining completed steps in order

FAST. We use the official FAST release[2] [29] and follow its recommended best practices in all experiments: encoded action chunks span 1–2 s, and every action dimension is normalized to $[-1, 1]$ by per-dataset q_{01}/q_{99} quantiles. One property of the decoding path deserves note. FAST requires the predicted action tokens, after BPE inversion, to form an integer array of exactly shape $T \times D$; if the length differs, the reshape fails, decoding aborts, and the policy emits a stationary rest action for the chunk. This places an excessive demand on the predictive precision of the VLA: a single superfluous or missing token silently discards an entire chunk. We therefore wrap the decoder with a *soft decode*: an inverted sequence shorter than $T \times D$ is zero-padded to length, and an overlong one is truncated. The repair is natural for a DCT-based transform: its quantized coefficient array concentrates mass at low frequencies and decays toward zero, so completing a short array with zeros or trimming a long tail stays close to the intended signal. Soft decoding improves FAST's downstream performance substantially in our experiments, and every reported FAST number uses it; we regard this as a responsible strengthening of the baseline. It changes only how a malformed decode is salvaged, never how actions are tokenized.

X-Tokenizer. We use the officially released code and pretrained weights[3] [13], aligning our action layouts with the official action-dimension slots and with physical units. We use the released weights as they are for every robot type with an official configuration, including WidowX and Franka. Zero-shot use of these weights, however, incurs reconstruction error far beyond what a discrete VLA can absorb, and the resulting policies are correspondingly weak. Having contacted the authors, we learned that X-Tokenizer is designed to accelerate policy learning inside the $\pi_{0.5}$ architecture [30] rather than to serve as a standalone tokenizer for discrete VLAs, and their recommendation was to fine-tune the pretrained weights on each target dataset. Every X-Tokenizer number in this paper therefore uses the official pretrained weights further fine-tuned on the task dataset at hand.

OAT. OAT [20] is a strong recent baseline and a Best/Outstanding Paper finalist at RSS; we use the official release[4]. No pretrained version of OAT exists; the tokenizer is learned from scratch on every dataset. Its stated goal is an action code with an ordered structure that attains high compression and total decodability at once: each successive token refines the decoded action. We note that the official training configuration deliberately sets a very small token budget, eight tokens per action chunk, which delivers extreme compression but sharply degrades closed-loop VLA performance. We

[2] https://huggingface.co/physical-intelligence/fast
[3] https://github.com/X-Square-Robot/X-Tokenizer
[4] https://github.com/Chaoqi-LIU/oat

respect the official default and report it as $\text{OAT}_{\text{official}}$ throughout; we additionally evaluate an aligned variant, $\text{OAT}_{\text{aligned}}$, whose token budget matches the average compression rate of ACodec2, so that the comparison isolates the tokenizer design from the bit rate.

ActionCodec. ACodec [9] is another VQ-based action tokenizer. Its motivation is to add auxiliary regularization that aligns the representation space with visual and language information, improving how well a VLA trains on top of the code. We use the officially released checkpoints; because they are already pretrained on LIBERO, BridgeData, and DROID, we apply them without further fine-tuning. The G0.5 backbone [23] used in our experiments states in its technical report that its internal action tokenizer is a modified ACodec, differing in network architecture, pretraining corpus, and engineering choices. The released ACodec we evaluate therefore inherits no advantage from G0.5's action-token prior, and the comparison stays fair.

ACodec2 variants. A first-order ACodec2 is fit on each benchmark corpus in minutes (Section 6). **ACodec2+** is the universal instance pretrained once on the roughly 20k-hour, 24-embodiment corpus of Section 6, applied to every benchmark by resampling alone with no per-corpus fitting. The stateful variants $\textbf{ACodec2}^{\text{II}}$ and $\textbf{ACodec2+}^{\text{II}}$ additionally track a velocity state under the second-order feasibility analysis of Appendix F.

Baselines and training details. Every tokenizer is plugged into the same released G0.5 VLA [23][5]. All experiments share AdamW with weight decay 0.01, gradient clipping at 1.0, and a cosine schedule decaying to 10% of the peak after a short linear warmup. On LIBERO [19], every tokenizer is trained for 10k steps at learning rate 2.5×10^{-5}, global batch 128, and action-chunk horizon 20 (of which 10 are executed). On SimplerEnv, policies are trained on BridgeData [37] for 50k steps at learning rate 5×10^{-5}, global batch 64, and horizon 10. On DROID-Franka, policies are trained on DROID [14] for 50k steps at learning rate 5×10^{-5}, global batch $1{,}344$, and horizon 15.

D.2 Evaluation benchmarks

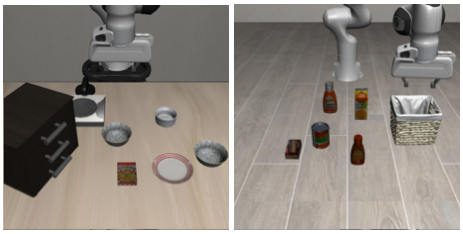
LIBERO, LIBERO-Plus, and SimplerEnv-WidowX score every rollout by binary task success from fixed initial states; DROID-Franka scores continuous *task progress*. Figure 9 shows example task observations.

LIBERO. LIBERO [19] is a MuJoCo benchmark built around a Franka Emika Panda arm observed through front and wrist cameras with proprioceptive state and a natural-language instruction. We use its 40 tasks in four suites that each isolate one type of distribution shift: *LIBERO-Spatial* (10 tasks) varies the spatial arrangement of a fixed object set, *LIBERO-Object* (10 tasks) replaces the object categories, *LIBERO-Goal* (10 tasks) keeps the scene fixed while changing the goal, and *LIBERO-Long* (10 tasks) composes these shifts into long-horizon tasks. Each task ships with 50 teleoperated demonstrations.

LIBERO-Plus. LIBERO-Plus [10] automatically generates 10,030 perturbed variants of the LIBERO tasks along seven controlled axes: object layout with distractors, camera viewpoint, robot initial state, language instruction, light condition, background texture, and sensor noise, with severity progressing from mild to extreme within each axis. It is a drop-in replacement for LIBERO, so the same checkpoints are re-evaluated without modification; following the official protocol, each variant is rolled out from its prescribed initial state. Table 4 reports the per-category results behind the LIBERO-Plus row of Table 1.

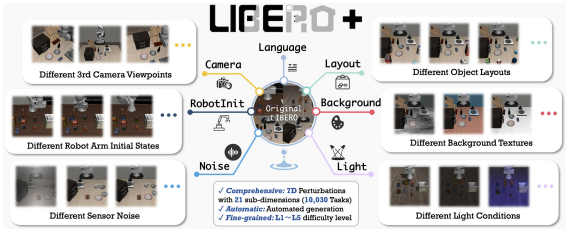
SimplerEnv-WidowX. SimplerEnv [18] reconstructs the BridgeData V2 [37] tabletop setup, based on a WidowX 250 arm, in SAPIEN, with lighting, textures, camera poses, and initial-state distributions tuned by system identification to match the real rig (*visual matching*). We adopt its four standard BridgeData tasks: *put spoon on towel*, *put carrot on plate*, *stack blocks*, and *put eggplant in the yellow basket*. Because zero-shot outcomes are high-variance, we expand the official 24 initial states (seeds) to 120 per task. Table 5 reports the full per-suite and per-task results behind the LIBERO and SimplerEnv rows of Table 1.

[5] https://huggingface.co/OpenGalaxea/G05/tree/main/g05-base



LIBERO-Spatial LIBERO-Object

(a) LIBERO



(b) LIBERO-Plus (seven perturbation axes)



spoon on towel carrot on plate stack blocks eggplant in basket

(c) SimplerEnv-WidowX (visual matching)



dining room laboratory laundry room office

(d) DROID-Franka (in-the-wild scenes of the DROID training corpus)



color pick-and-place snack storing & serving nested-cup arrangement towel unfolding

(e) DROID-Franka (initial states of the four self-designed evaluation suites)

Figure 9: **Example task observations of the evaluation benchmarks.** (a) LIBERO [19]: MuJoCo Franka Panda tasks observed through the front camera; shown are one *Spatial* and one *Object* task out of the four suites. (b) LIBERO-Plus [10]: 10,030 variants of the LIBERO tasks generated along seven perturbation axes (camera, robot init, language, layout, background, light, noise). (c) SimplerEnv-WidowX [18]: the four BridgeData tasks [37] rebuilt in SAPIEN under visual matching. (d) DROID-Franka [14]: in-the-wild scenes of the DROID corpus on which the policies are fine-tuned. (e) DROID-Franka: the initial states of the four self-designed suites (color-discriminated pick-and-place, snack storing and serving, nested-cup arrangement, towel unfolding) on which the fine-tuned policies are evaluated zero-shot; task progress is reported in Figure 8. Images in (a)–(c) are reproduced from the official benchmark repositories.

DROID-Franka. In-the-wild rollouts are long-horizon and rarely end in one binary event, so each rollout is scored by *task progress*: the fraction of prescribed milestones attained. A task group's score averages its rollouts (and the subtasks of the color group), and the overall score weights the four groups equally. *Color-discriminated pick-and-place* confronts the robot with candies and plates of two colors in four instructed combinations (5 rollouts each): a rollout earns 0.5 for grasping the candy of the instructed color and the remaining 0.5 for placing it on the plate of the instructed color. In the *yellow→green* and *blue→pink* combinations, a plate of the wrong color covers half of the instructed-color plate as a distractor, raising the difficulty. *Snack storing* asks the robot to store five items, each worth 0.2: one blue candy, two yellow candies, and two cola bottles. Two purple

Table 4: **LIBERO-Plus robustness (%): per-category breakdown of Table 1.** Task-level success over all 10,030 perturbed tasks; each column pools the perturbations in that category. **Bold** is the best in a column, underline the second best.

Tokenizer	Camera	Robot	Lang.	Light	Backgr.	Noise	Layout	Avg.
FAST [29]	55.2	64.5	73.8	86.9	87.5	51.4	80.1	69.7
$\text{OAT}_{\text{official}}$ [20]	48.0	53.8	73.7	84.6	83.7	44.1	74.8	64.3
$\text{OAT}_{\text{aligned}}$ [20]	62.5	61.5	**78.7**	94.5	89.9	53.8	83.5	73.2
ACodec [9]	59.7	57.8	75.3	90.8	87.0	48.5	79.0	69.4
X-Tokenizer [13]	42.7	52.3	56.9	67.8	65.3	34.5	59.8	52.9
ACodec2	58.2	60.9	78.5	92.7	90.4	52.3	82.1	71.8
ACodec2+	64.0	65.7	76.7	94.9	92.7	57.8	83.3	74.8
ACodec2^{II}	65.4	81.9	78.1	**95.4**	93.5	56.1	84.1	77.7
ACodec2+^{II}	**66.4**	**85.3**	77.5	90.6	**97.1**	**63.9**	**84.8**	**79.5**

Table 5: **Full LIBERO and SimplerEnv results (%): per-suite and per-task breakdown of Table 1.** Every row is the same G0.5 VLA [23] trained with the corresponding tokenizer. LIBERO: evaluated on the four LIBERO suites with 50 official initial states per task (2,000 episodes). SimplerEnv: four WidowX tasks, 120 initial states each (480 episodes). **Bold** is the best in a column, underline the second best.

	LIBERO					SimplerEnv				
Tokenizer	Spatial	Object	Goal	Long	Avg.	Spoon	Carrot	Stack	Eggplant	Avg.
FAST [29]	95.4	99.2	85.4	91.2	92.8	91.7	64.2	88.3	**100.0**	86.0
$\text{OAT}_{\text{official}}$ [20]	93.8	95.8	88.4	85.8	91.0	9.2	16.7	2.5	23.3	12.9
$\text{OAT}_{\text{aligned}}$ [20]	98.2	99.8	95.6	91.8	96.4	9.2	7.5	15.0	44.2	19.0
ACodec [9]	95.8	99.6	94.8	90.2	95.1	78.3	55.0	47.5	76.7	64.4
X-Tokenizer [13]	88.6	96.0	87.6	82.8	88.8	90.8	65.8	81.7	95.0	83.3
ACodec2	98.6	99.2	97.8	91.8	96.9	95.8	83.3	90.0	94.2	90.8
ACodec2+	97.2	99.6	95.4	94.4	96.7	97.5	85.0	89.2	95.8	91.9
ACodec2^{II}	98.2	**100.0**	95.8	95.2	97.3	95.8	95.0	**94.2**	87.5	93.1
ACodec2+^{II}	**98.8**	99.4	**98.2**	**96.4**	**98.2**	**98.3**	**96.7**	89.2	90.8	**93.8**

distractor candies must be left out at a penalty of -0.2 each, so the maximum rollout score of 1.0 requires storing every correct item and no distractor (10 rollouts). *Nested-cup arrangement* stacks three cups and places the stack on a plate, with five milestones of 0.2 each: blue on pink, blue into pink, the pair on yellow, the pair into yellow, the stack onto the green plate (10 rollouts). *Towel unfolding* carries three stages, intent, grasp, and unfold, worth 0.33, 0.67, and 1.0 cumulatively (10 rollouts). The real-robot study compares five policies, FAST, X-Tokenizer, ACodec2, ACodec2+, and ACodec2+^{II}, over 50 rollouts each; Figure 8 reports the task progress of every policy on each suite, and Figure 10 breaks each suite down to the milestone level.

E ACTIONCODEC2 AS AN AGENT CONTROL INTERFACE

The case study of Section 7.5 let a VLM agent drive the simulated YAM robot through ACodec2 tokens and watched macros accumulate over long episodes. This section reports a controlled companion study on RMBench [7] that isolates what the control representation itself contributes: one agent and one task, and three interfaces that differ only in how the agent speaks actions.

E.1 RMBENCH BLOCKS-TRY

Setup. An episode of *blocks-try* asks the agent to find, by trial, the correct ranking of three blocks: six orderings are possible, nothing in the scene reveals the right one, so the agent can only keep trying—it must remember which orderings it has already attempted and reason about which possibilities remain. We compare three control interfaces, illustrated in Figure 11(*a*): (*i*) **Language Command**, where the model writes the absolute target TCP pose for the next $0.5\,\text{s}$ window; (*ii*) **ActionCodec2 tokens**, where the model emits ActionCodec2 tokens that steer the trajectory over the same window; and (*iii*) **ActionCodec2 macros**, which extends (*ii*) with a library of named token-sequence macros (Table 7 shows the one the agent induced)—the agent may call macros, is encouraged to merge

Table 6: **Full ablation results (%): per-suite breakdown of Table 2.** Protocols and controlled variables as in Section 7.6; both packers are lossless, and gray values give the gain of Set-BPE. **Bold** is the best within a panel, underline the second best.

Tokenizer / setting	Spatial	Object	Goal	Long	Avg.
Deployment: real-time control at 20 Hz					
FAST [29] / chunk	35.0	40.0	47.2	21.4	35.9
X-Tokenizer [13] / chunk	58.8	72.8	65.6	45.0	60.6
ACodec [9] / chunk	86.0	94.6	83.8	73.4	84.5
$\text{OAT}_{\text{aligned}}$ [20] / chunk	92.0	94.0	86.4	76.4	87.2
ACodec2+ / chunk	82.2	83.0	72.8	54.4	73.1
ACodec2+ / streaming	**94.0**	**96.8**	**88.2**	**85.0**	**91.0**
Packing: Set-BPE vs. order-dependent BPE					
ACodec2 / order-dependent BPE	96.0	98.8	94.8	87.4	94.3
ACodec2 / Set-BPE (Table 5)	**98.6**(+2.6)	**99.2**(+0.4)	**97.8**(+3.0)	**91.8**(+4.4)	**96.9**(+2.6)

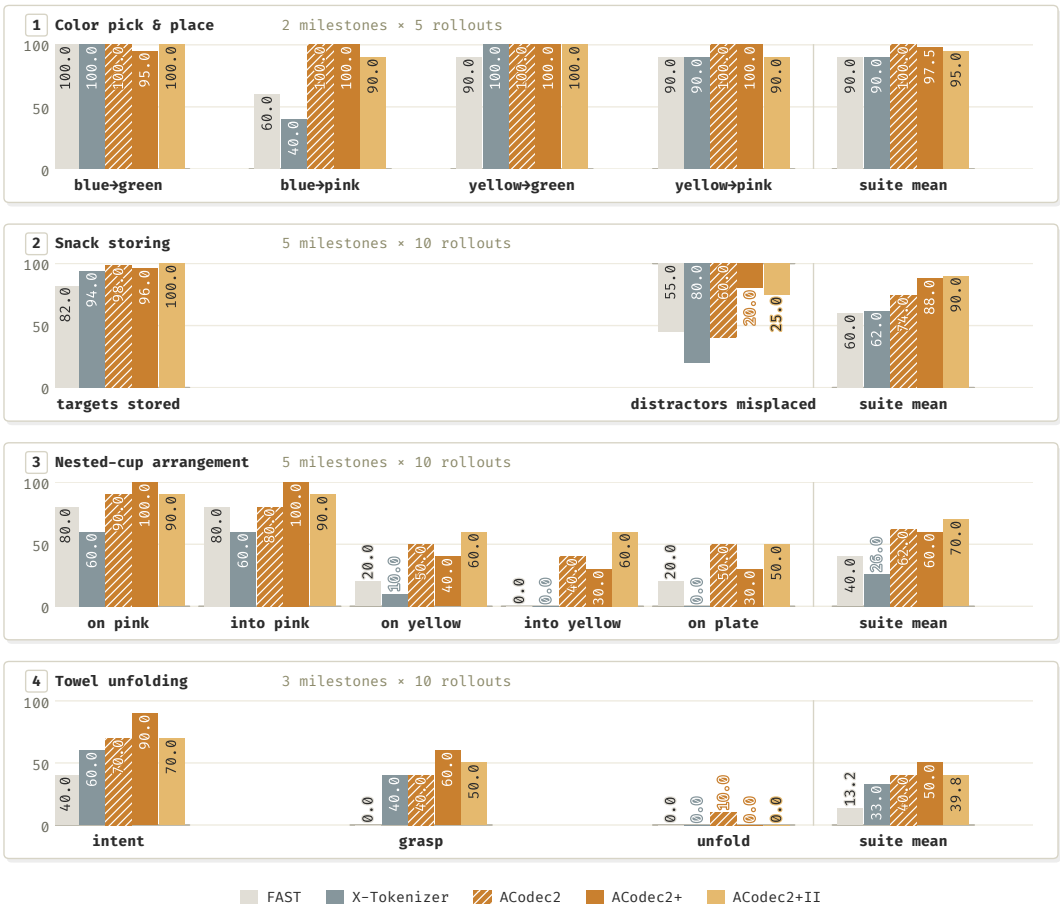


Figure 10: **DROID-Franka per-suite breakdown (%).** The milestone-level process behind Figure 8: task progress per instructed color pair, target items stored vs. distractor items misplaced (drawn from the inverted top axis; fewer is better), per-milestone nested-cup success, and towel-unfold attainment. Each row closes with the suite mean behind a divider; colors and legend match Figure 8.

verified, reusable token combinations into new ones, and at the episode's end a reflection step may create or merge macros further, whose curated library is inherited by the next episode. All three interfaces share the same episodic memory: end-of-episode reflections are injected into the next episode's system prompt. We use GPT-6 Astra [28] at low reasoning effort.

Findings. ActionCodec2 tokens beat absolute pose commands on both costs the agent controls: fewer decisions per action window and less thinking per submission, and the gap widens as macros

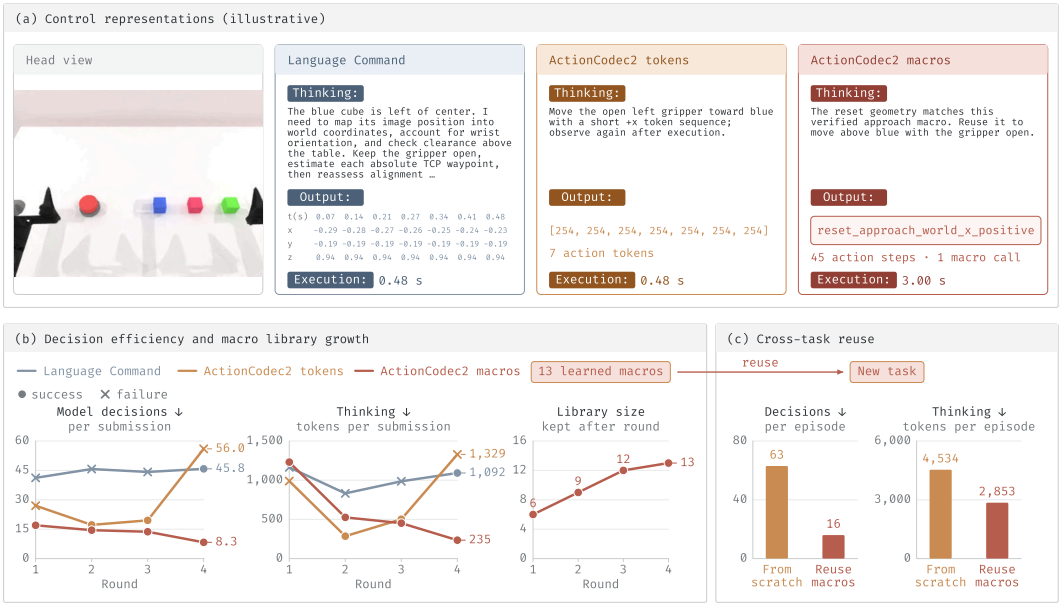


Figure 11: **Three control interfaces for one VLM agent on RMBench blocks-try.** (*a*) One decision under each representation (illustrative). (*b*) Four chained rounds per interface (GPT-6 Astra, low effort): decisions and thinking tokens per action window, and macro-library size. (*c*) Transfer to a held-out rearrange task (GPT-6 Astra, low).

absorb the routine geometry. The macro library is also what makes the token interface stable—reset after every round, the agent keeps re-deriving its routines at erratic cost; inherited across rounds, the library grows, macro calls take over more of the executed steps, and decisions and thinking fall in lock-step until every round succeeds. To probe whether the distilled library generalizes, we ran a further contrast on a new block-rearrange task, both arms using the ActionCodec2-macros interface: one from scratch, the other initialized with the 13 training macros; decisions and thinking tokens both drop markedly, so the macros transfer at least across similar tasks. The induced macros (Table 7) explain the gain: task-specific routines with stated preconditions and effects, named and worded by the VLM itself rather than by us—a light form of test-time training (TTT) [35] on the agent's own action vocabulary, run entirely inside its context; baking macros into the weights, in a genuine TTT sense, is natural future work.

E.2 Milestones of the Multi-Drawer case study

The badges of Figure 7 mark nine first-time achievements across the twenty episodes of the Multi-Drawer case study (Section 7.5); Table 8 quotes the model's own words at each, with the macros born or executed there. This subsection walks the arc and states what each milestone changed for later episodes.

Brute force (episodes 1–5). The founding discovery is the $+90°$ wrist roll: horizontal handle bars can only be pinched by vertically opening fingers, and every later drawer interaction is a variant of this grip. The same era fixed the gripper-reading semantics that later episodes treat as law—a full close near 0.004 means the fingers closed on nothing, 0.2–0.4 is a real bite—and delivered the first positive score, which the post-episode review exposed as a false trigger: the bin had been shoved under an open drawer, so the scorer saw the cup as delivered. The lasting outputs are rules, not motion—never move the bin; trust the instruction line, not the image.

Primitive accumulation (episodes 6–10). Reliable motions sediment into macros: slow 1 cm-step pulls (fast pulls slip off), the handle standoff, and in episode 8 the first genuine cup grasp—whose transport immediately snagged on the drawer handle, producing the "rise before carry" rule. Protocol pitfalls surfaced alongside: one arm's rejected IK voids the entire reply, a hold token issued to an open gripper silently drops the object, and forty decisions were spent grasping a peeler the camera

Table 7: **Macros induced by the VLM on RMBench** (the library carried into Figure 11c). Names and semantics are the model's own words, not post-hoc human summaries.

Macro	Semantics (VLM-authored)	Uses	Duration (s)
grasp_from_row_hover	From an aligned hover 10 cm above grasp height, descend open, close, and test-lift.	26	1.40
park_cube_in_front	With a verified cube lifted 4 cm, buffer 15 cm nearward, release, and withdraw.	20	2.40
set_down_and_clear	With a held cube 4 cm above placement height, lower, release, and withdraw 10 cm.	33	1.40
buffer_to_row_left_hover	From a matching front buffer hover, approach the neighbouring row slot 9 cm left and 15 cm forward.	2	1.60
row_to_front_buffer_hover	From a row-slot hover, move open 15 cm nearward to its front buffer.	2	1.00
buffer_to_row_left_and_place	With a verified buffer cube lifted 4 cm, add clearance, transfer 9 cm left and 15 cm forward, place, and withdraw.	2	3.80
adjacent_cube_right_and_place	With a verified cube lifted 4 cm, transfer 9 cm right into an empty slot, place, and withdraw.	3	2.00
descend_open_to_cube	With downward fingers above a cube, descend open 7 cm for alignment inspection.	2	0.47
orient_down	Pitch about -88° open; only from a verified compatible orientation with clearance.	7	2.27
carry_slot_left	Move base $-x$ 10 cm closed; require verified grasp, spacing and clearance.	15	0.67
carry_slot_right	Move base $+x$ 10 cm closed; require verified grasp, spacing and clearance.	13	0.67
button_press	Close, command 8 cm descent and hold; require cap alignment, flat fingertips and both grippers previously opened.	12	1.33
carry_forward_12cm	Move base $+y$ 12 cm closed; require verified grasp and clear route.	14	0.80

had called a cup. Each fix became a standing discipline, and the operator addendum that closes this era hands the agent the layout ground truth—seeing is confirming, not discovering.

Proceduralization (episodes 11–14). Episode 11 is the hinge of the arc. Its reflection diagnosed the failure as epistemic—"lessons were facts, not procedures"; every decision re-derived heights, drawer level, and escape orders—and answered with a phase *playbook* that compiles the cup chain into named phases, phase macros, and a reading-to-action table. From episode 12 on, the chain executes on the playbook, and episode 13 completes the first genuine cup delivery; when the left arm then hit its reach limit with the tongs, it dropped them on the table and the right arm pinched them vertically—an unplanned first left-to-right handover.

Refinement and the final mile (episodes 15–20). The closing episodes run on negative results. Episode 15, the series' worst, misread both targets and cemented layout-as-truth plus the mirrored -90° right-arm clamp. Episode 16's level-view-first discipline ended the empty-close plague, and its mapping of the left arm's carry wall produced the terminal strategy: deposit the object near the table center early and let the bin-side arm fetch it. Episode 19, with the decision budget doubled, verified the first in-drawer tongs grasp; episode 20 cleared the interference directly—pushing the overhanging drawer shut as an offensive move—which made the steep center pinch on the tongs deterministic, and the deliberate release over the bin center closed the chain: the series' first environment-confirmed success.

Findings. First, experience compounds only when compiled: the same facts that failed as prose became competence as procedures (the playbook) and as named token sequences (macros); the library

Table 8: **The nine first-time milestones of Figure 7, in the model's own words.** Thinking is quoted (abridged) from the decision logs and end-of-episode reflections; macros are those born at, or executed during, the milestone (the library is append-only).

Ep.	Milestone	*Model's thinking*	`Macros`
2	First drawer opened	*"Three closes have missed. The wrist view shows the fingers open left–right, while the handle bar also runs left–right, parallel to the finger axis, so the fingers can never pinch it. I'll roll the tool about 90° so the fingers open vertically and can straddle the horizontal bar."*	`roll_plus_90_open`, `rolled_drawer_grip_pull`
3	Spotted the tongs	*"The left wrist view looks into the open top drawer of the left cabinet and shows only silver tongs, not the measuring cup."*	`open_drawer_from_handle`, `tool_axis_advance_pitched`
5	Found the measuring cup	*"The head view now shows the measuring cup in the open right-cabinet drawer: a white round bowl with a handle."*	`slow_pull_drawer_10cm`, `pull_drawer_by_side_wall`
8	First cup grasp	*"The close stalled at 0.113, which suggests the thin tab is pinching again. … this time I'm holding closed for 16 rows to build full grip force, then lifting very slowly."*	`pitched_yaw_center_pinch`, `close_lift_slow_2`
11	Invented the playbook	*"Lessons were facts, not procedures. Each decision re-derived heights, drawer level, and the escape order. … I kept guessing." (reflection)*	`drawer_open_rolled_full`, `grasp_close_verify_lift`, `carry_pitchup_back_10`, `release_open_rise`
13	First cup into the bin	*"The cup is in the bin: the right wrist view shows it resting inside, and the instruction has advanced to the tongs."*	`carry_pitchup_back_10`, `release_open_rise`, `table_pinch_vertical`
16	Devised the handover	*"When the target's drawer is on the far side from the bin, set the object down near the table centre early and let the bin-side arm fetch it." (hypothesis)*	`left_carry_minus_y_low`, `pitched_hook_handle_extend`
19	First tongs grasp	*"The lift worked: the wrist moved back 3 cm and up 2.6 cm, and the reading holds at 0.232. The wrist view shows the tongs still clamped between the fingers."*	`grasp_close_verify_lift`, `pitch_up_13_closed`
20	First full success	*"The close stalled at 0.204 and the 4.6 cm lift completed, so the tongs are grasped."*	`steep_center_pinch`, `bin_lower_release_rise`

is append-only, so nothing learned is unlearned. Second, most permanent rules govern interpretation, not motion—what a gripper reading means, what an image shows, what voids a reply—so the binding constraint is perception and protocol, not actuation. Third, the terminal handover strategy was found by the model mapping its own kinematic limits and executed with primitives the first episodes already knew; the finish line required no new motion, only better organization of old ones.

F THE SECOND-ORDER SCHEME ON A GROUP

At first order the feasible successors $\mathcal{F}(q) = q\mathcal{A}$ carry the same relative motions to every configuration, and Theorem 2 covers them in closed form. Acceleration limits and workspace boundaries break this: the feasible next increment depends on the current one and on the distance to the boundary, state that the first-order error recursion cancels. Folding every truncation into one enlarged $\mathcal{A}$ covers increments that are never realized, such as a one-step reversal from full speed, and inflates the count. The second-order variant of Section 6 instead lets the decoder track the increment, so that a primitive names a bounded change of increment, a smaller set to cover. This appendix supplies its theory in three steps.

1. Section F.1 defines the decoder, whose reconstruction obeys its own copy of the real limits widened by a margin, and replaces the full tolerance requirement, which no primitive set meets here, by an exact test and the optimum $K_2^\star$ it defines.

2. Section F.2 shows why the margin is needed: a strict decoder cannot close a lag, while a margin of a few ϵ admits an explicit primitive grid on translation and on $\mathrm{SO}(3)$. These are the grids the fit uses.
3. Section F.3 computes $K_2^\star$ exactly on translation and explains why the search does not transfer to $\mathrm{SO}(3)$.

F.1 The decoder and the exact criterion

The configuration becomes the pair $(q, \delta) \in G \times G$ of position and the increment that produced it, $\delta_t = q_{t-1}^{-1} q_t$, so that a step first updates the increment by the change $\gamma_t = \delta_t^{-1} \delta_{t+1}$ and then applies it,

$$(q, \delta) \mapsto (q\,\delta\,\gamma,\ \delta\,\gamma). \tag{8}$$

An instance fixes closed sets $\mathcal{W}, \mathcal{D}, \mathcal{C} \subseteq G$, and a trajectory is admissible while $q_t \in \mathcal{W}$, $\delta_t \in \mathcal{D}$, $\gamma_t \in \mathcal{C}$. A state can obey all three and still have no infinite admissible continuation, so we use the viable states $\mathcal{V}_2 = \{(q, \delta) :\ q\delta^{-1} \in \mathcal{W}$ and an infinite admissible trajectory starts at $(q, \delta)\}$, the first clause asking that the step producing q also started in the workspace, and their feasible changes $\mathcal{F}(q, \delta) = \{\gamma \in \mathcal{C} :\ (q\delta\gamma, \delta\gamma) \in \mathcal{V}_2\}$, nonempty at every viable state.

At first order nothing asks the reconstruction to be a configuration the robot could reach. At second order its increment $\hat{\delta}$ is the rate the decoded stream commands, so we must say which limits it obeys. We let it run its own instance, nested around the real one: closed $\mathcal{W} \subseteq \mathcal{W}^\mu$, $\mathcal{D} \subseteq \mathcal{D}^\mu$, $\mathcal{C} \subseteq \mathcal{C}^\mu$, with viable states $\mathcal{V}_2^\mu$ and feasible changes $\mathcal{F}^\mu$. The *margin* $\mu \geq 0$ has one component per relaxed bound, and the *strict* case $\mu = 0$ holds the reconstruction to the robot's own limits. The tolerance $\mathcal{E}$ is never enlarged, so a margin changes what the reconstruction may command and not how accurate it must be.

A primitive $u \in \mathcal{U} \subseteq G$ is a fixed request for the next change. It is made without regard to the reconstructed state, so it is clipped to the nearest feasible value $\bar{u} = \Pi_{\mathcal{F}^\mu(\hat{q}, \hat{\delta})}(u)$ and executed as

$$\hat{\delta}_{t+1} = \hat{\delta}_t \bar{u}_t, \qquad \hat{q}_{t+1} = \hat{q}_t \hat{\delta}_{t+1}. \tag{9}$$

We assume $\mathcal{F}^\mu$ is compact and nonempty on $\mathcal{V}_2^\mu$ and fix one nearest-point selection; both instances below satisfy this. Since $\bar{u}_t \in \mathcal{F}^\mu(\hat{q}_t, \hat{\delta}_t)$, the reconstruction never leaves $\mathcal{V}_2^\mu$. With the errors $\xi_t = \hat{q}_t^{-1} q_t$ and $\psi_t = \hat{\delta}_t^{-1} \delta_t$, one step gives

$$\psi_{t+1} = \bar{u}_t^{-1}\, \psi_t\, \gamma_t, \qquad \xi_{t+1} = \bar{u}_t^{-1}\, c_{\hat{\delta}_t}(\xi_t)\, \psi_t\, \gamma_t, \qquad c_h(\xi) = h^{-1} \xi h, \tag{10}$$

the first-order step $\xi_{t+1} = u^{-1} \xi_t \delta_t$ conjugated into the frame of the current increment. The tolerance applies to ξ alone, and the accepted pairs are $\mathcal{S}_2 = \{(q, \delta; \hat{q}, \hat{\delta}) :\ (q, \delta) \in \mathcal{V}_2,\ (\hat{q}, \hat{\delta}) \in \mathcal{V}_2^\mu,\ \hat{q}^{-1} q \in \mathcal{E}\}$.

Translation takes $G = (\mathbb{R}, +)$ with $\mathcal{W} = [\ell, h]$, $\mathcal{D} = [-b, b]$, $\mathcal{C} = [-c, c]$, $\mathcal{E} = [-\epsilon, \epsilon]$. Writing the pair as (x, v) and the change as g_t, (10) is plain accumulation,

$$\psi_{t+1} = \psi_t + g_t - \bar{u}_t, \qquad \xi_{t+1} = \xi_t + \psi_{t+1}, \tag{11}$$

so quantizing each change separately does not control the position. For $c > 0$ the extra distance needed to bring a positive displacement v to rest under $|g| \leq c$ is $B(v) = \sum_{i=1}^{\lceil v/c \rceil - 1} (v - ic)$, with $B(v) = 0$ for $v \leq 0$, and

$$\mathcal{V}_2 = \big\{(x, v) :\ |v| \leq b, \quad x - v \in [\ell, h], \quad x + B(v) \leq h, \quad x - B(-v) \geq \ell\big\}. \tag{12}$$

The margin has three components, replacing $[\ell, h]$, b, c by $[\ell - \mu_x, h + \mu_x]$, $b + \mu_v$, $c + \mu_g$. Orientation takes $G = \mathrm{SO}(3)$ with $\mathcal{W} = \mathrm{SO}(3)$, $\mathcal{D} = \mathcal{B}_R(b_R)$, $\mathcal{C} = \mathcal{B}_R(c_R)$, $\mathcal{E} = \mathcal{B}_R(\epsilon_R)$ in the notation of Corollary 4. With no boundary nothing has to brake, since repeating $\gamma = I_3$ continues forever, so $\mathcal{V}_2 = \mathrm{SO}(3) \times \mathcal{B}_R(b_R)$ and $\mathcal{F}(q, \delta) = \{\gamma \in \mathcal{B}_R(c_R) :\ \theta(\delta\gamma) \leq b_R\}$. The margin has two components, one on the increment angle and one on the change angle.

At first order the optimum is defined by the full tolerance requirement (2), a label for every accepted pair. On $\mathcal{S}_2$ this asks too much. $\mathcal{S}_2$ tests ξ alone, so it contains pairs whose ψ is as large as the increment sets allow, while one label moves $\hat{\delta}$ by a bounded amount. On translation a pair with $\xi = \epsilon$

and $\psi = 2b$ leaves $\mathcal{E}$ whatever label is sent once $2b > 2c + \mu_g$, by (11), so the requirement rejects every primitive set, even those that never reach such a pair from an exact start. The guarantee needs only a set of accepted pairs that contains the exact starts and that the decoder can always remain inside. Proposition 6 shows that this condition is exact and obtains the largest such set, the viability kernel of the accepted pairs, as the limit of a decreasing recursion. It holds for the general decoder (1) on any G, and the full tolerance requirement is its condition (3) with $\mathcal{R} = \mathcal{S}$.

Proposition 6 (Invariant-set criterion). *Assume every configuration has an infinite feasible continuation.*

(a) The criterion. *There exists a causal encoding policy $z_t = \pi_t(q_{0:t+1})$ under which the decoder* (1) *keeps $(q_t, \hat{q}_t) \in \mathcal{S}$ on every infinite feasible trajectory, from every exact start, if and only if there is a set $\mathcal{R}$ of pairs with (1) $\mathcal{R} \subseteq \mathcal{S}$; (2) $(q, q) \in \mathcal{R}$ for every q; (3) $\forall (q, \hat{q}) \in \mathcal{R},\ \forall q^+ \in \mathcal{F}(q),\ \exists k \le K :$ $\big(q^+, y_k(\hat{q})\big) \in \mathcal{R}$. A policy that succeeds may then be taken stationary.*

(b) The kernel. *Put $\mathcal{Q}_0 = \mathcal{S}$ and*

$$\mathcal{Q}_{n+1} = \big\{(q, \hat{q}) \in \mathcal{Q}_n :\ \forall q^+ \in \mathcal{F}(q)\ \exists k \le K \text{ with } \big(q^+, y_k(\hat{q})\big) \in \mathcal{Q}_n\big\}, \qquad \mathcal{Q}_\infty = \bigcap_n \mathcal{Q}_n. \tag{13}$$

Then $\mathcal{Q}_n$ is exactly the set of pairs from which some causal policy survives n more steps; every $\mathcal{R}$ obeying (1) and (3) lies in $\mathcal{Q}_\infty$; and when K is finite $\mathcal{Q}_\infty$ itself obeys (1) and (3). A largest admissible $\mathcal{R}$ therefore exists and equals $\mathcal{Q}_\infty$, and the criterion of (a) is $\{(q, q)\} \subseteq \mathcal{Q}_\infty$.

Proof. **(a) Sufficiency.** Repeat the proof of Lemma 1 with $\mathcal{R}$ in place of $\mathcal{S}$: requirement 2 starts the induction at (q_0, q_0), requirement 3 continues it, and requirement 1 returns the conclusion to $\mathcal{S}$. The label chosen there depends on $(q, \hat{q})$ and q^+ and not on t, which is what makes the policy stationary.

(a) Necessity. Suppose the decoder succeeds from every exact start, and let $\mathcal{R}$ be the set of pairs $(q_t, \hat{q}_t)$ it reaches over all exact starts and all infinite feasible trajectories. Requirement 1 holds because success means every reached pair lies in $\mathcal{S}$, and requirement 2 because the exact start at q reaches (q, q) at $t = 0$. For requirement 3, take $(q, \hat{q}) \in \mathcal{R}$, reached at some time t along a feasible history $q_{0:t}$ with $q_t = q$, and take any $q^+ \in \mathcal{F}(q)$. Causality is what makes q^+ free to choose here: π_s reads only $q_{0:s+1}$, so $z_{0:t-1}$ is determined by $q_{0:t}$ and hence so is $\hat{q}_t = \hat{q}$. Extend $q_{0:t}q^+$ by an infinite feasible continuation, which exists by assumption. The result is an infinite feasible trajectory agreeing with the history on $q_{0:t}$, so the encoder reaches the same $\hat{q}_t = \hat{q}$ and then emits some $z_t = k$ after observing q^+; the decoder moves to $y_k(\hat{q})$. The pair $\big(q^+, y_k(\hat{q})\big)$ is therefore reached and lies in $\mathcal{R}$.

(b). That $\mathcal{Q}_n$ is the n-step survivor set is induction on n: $\mathcal{Q}_0 = \mathcal{S}$, and a pair permits $n + 1$ safe steps exactly when it is safe now and, after every feasible successor, some label reaches a pair permitting n more, which is (13), the rule observing q^+ before choosing the label. If $\mathcal{R}$ obeys (1) and (3) then $\mathcal{R} \subseteq \mathcal{Q}_0$, and $\mathcal{R} \subseteq \mathcal{Q}_n$ gives $\mathcal{R} \subseteq \mathcal{Q}_{n+1}$ by (3), so $\mathcal{R} \subseteq \mathcal{Q}_\infty$. For the converse, $\mathcal{Q}_\infty \subseteq \mathcal{Q}_0 = \mathcal{S}$ is (1); for (3) fix $(q, \hat{q}) \in \mathcal{Q}_\infty$ and $q^+ \in \mathcal{F}(q)$, so each membership $(q, \hat{q}) \in \mathcal{Q}_{n+1}$ supplies a label k_n whose successor lies in $\mathcal{Q}_n$. With K finite one label $k^\star$ recurs for infinitely many n, and the sets decrease, so its successor lies in every $\mathcal{Q}_m$. □

On $\mathcal{S}_2$ the recursion reads as follows. Write $y^+(\gamma, u)$ for the pair reached from y when the real system takes $\gamma \in \mathcal{F}(q, \delta)$ and the decoder, having observed γ, takes $u \in \mathcal{U}$. Starting from $Q_0(\mathcal{U}) = \mathcal{S}_2$,

$$Q_{n+1}(\mathcal{U}) = \big\{y \in Q_n(\mathcal{U}) :\ \forall \gamma \in \mathcal{F}(q, \delta)\ \exists u \in \mathcal{U} \text{ with } y^+(\gamma, u) \in Q_n(\mathcal{U})\big\}, \qquad Q_\infty(\mathcal{U}) = \bigcap_n Q_n(\mathcal{U}), \tag{14}$$

and exact initialization is $\mathcal{I} = \{(q, \delta;\, q, \delta) : (q, \delta) \in \mathcal{V}_2\}$.

Theorem 7 (Exact test for a fixed second-order primitive set). *Let $\mathcal{U} \subseteq G$ be finite. Then (1) $Q_n(\mathcal{U})$ is exactly the set of pairs from which some causal encoding policy $z_t = \pi_t(q_{0:t+1})$, in the sense of Lemma 1, keeps the pair in $\mathcal{S}_2$ for the next n steps; (2) every $y \in Q_\infty(\mathcal{U})$ has, for each real $\gamma \in \mathcal{F}(q, \delta)$, some $u \in \mathcal{U}$ with $y^+(\gamma, u) \in Q_\infty(\mathcal{U})$; and (3) $\mathcal{U}$ tracks every infinite admissible trajectory from exact initialization, with $\xi_t \in \mathcal{E}$ for all t, if and only if*

$$\mathcal{I} \subseteq Q_\infty(\mathcal{U}). \tag{15}$$

Proof. This is Proposition 6 on the pair state space. The real configuration is (q, δ) and the decoder (9) is (1) on $(\hat{q}, \hat{\delta})$ with the finite label set $\mathcal{U}$; $\mathcal{S}_2$ plays the role of $\mathcal{S}$, $\mathcal{F}(q, \delta)$ that of $\mathcal{F}(q)$, and (14) is then literally (13). Restricting the two state spaces to $\mathcal{V}_2$ and $\mathcal{V}_2^{\mu}$ is how that proposition's infinite-continuation assumption is met, and its exact starts are $\mathcal{I}$. Parts (1) and (2) are Proposition 6(b), the finiteness of $\mathcal{U}$ being what (2) needs; part (3) is its criterion $\mathcal{I} \subseteq Q_\infty(\mathcal{U})$ together with (a). □

The optimum is therefore

$$K_2^\star(\mu) = \min\big\{|\mathcal{U}| : \mathcal{U} \subseteq G \text{ finite}, \ \mathcal{I} \subseteq Q_\infty(\mathcal{U})\big\}, \tag{16}$$

$+\infty$ if no finite set works.

F.2 Why the decoder needs a margin, and the grids it admits

Lemma 8 (A strict decoder cannot close a lag). *Take the translational instance with $\mu = 0$ and $c > 0$. Suppose the displacement error satisfies $\psi_t \geq \rho > 0$ at some time t, and the real trajectory takes $g_s = c$ for $s = t, \ldots, t + n - 1$. Then $\psi_s \geq \rho$ for every $s = t, \ldots, t + n$, and*

$$\xi_{t+n} \ \geq \ \xi_t + n\rho. \tag{17}$$

The same holds with every sign reversed when $\psi_t \leq -\rho$ and $g_s = -c$.

Proof. At $\mu = 0$ the reconstruction's feasible set satisfies $\mathcal{F}(\hat{x}, \hat{v}) \subseteq \mathcal{C} = [-c, c]$, so every executed value obeys $\bar{u}_s \leq c$, whatever the primitive set. With $g_s = c$, the first half of (11) gives $\psi_{s+1} = \psi_s + c - \bar{u}_s \geq \psi_s$, and induction from $\psi_t \geq \rho$ gives $\psi_s \geq \rho$ for all s up to $t + n$. Summing the second half over $s = t, \ldots, t + n - 1$ gives $\xi_{t+n} = \xi_t + \sum_{s=t+1}^{t+n} \psi_s \geq \xi_t + n\rho$. □

Since $\mathcal{E}$ bounds ξ, (17) caps $n\rho$: the longer the robot can hold its sharpest maneuver, the smaller the lag a strict decoder may ever open, and its primitives must be fine enough to guarantee that in advance. In particular no box $|\xi| \leq \epsilon_\xi$, $|\psi| \leq \epsilon_\psi$ with $\epsilon_\psi > 0$ is invariant in the strict case, since its corner $\xi = \epsilon_\xi$, $\psi = \epsilon_\psi$ away from the workspace boundary gives $\xi_{t+1} \geq \epsilon_\xi + \epsilon_\psi$ under $g_t = c$. A margin removes the mechanism: with $\mu_g > 0$ an executed value may exceed c, so ψ can strictly decrease and an opened lag can close. How much margin suffices is explicit on both instances. In each, the encoder sends the primitive nearest to the request $W_t = \hat{\delta}_t^{-1} \hat{q}_t^{-1} q_{t+1}$, the change that would land the reconstruction exactly on the next configuration; on translation $W_t = \xi_t + \psi_t + g_t$.

Proposition 9 (Explicit grids under a margin). (a) Translation. *Let $\mu^\dagger = (\epsilon, 2\epsilon, 2\epsilon)$ and $\mu \geq \mu^\dagger$ componentwise. Place $K = \lceil c/\epsilon \rceil + 3$ points uniformly on $[-c - 2\epsilon, \ c + 2\epsilon]$, endpoints included. Then $|\xi_t| \leq \epsilon$ for every t from exact initialization, so $\mathcal{I} \subseteq Q_\infty(\mathcal{U})$ and $K_2^\star(\mu) \leq \lceil c/\epsilon \rceil + 3$.*

(b) Orientation. *Let the margin widen the increment angle by at least $2\epsilon_R$ and the change angle by at least $3\epsilon_R$, put $\rho = \min\{\pi, c_R + 3\epsilon_R\}$, and let $\mathcal{U} \subseteq \mathcal{B}_R(\rho)$ satisfy $\mathcal{B}_R(\rho) \subseteq \bigcup_{u \in \mathcal{U}} u\, \mathcal{B}_R(\epsilon_R)$. Then $\theta(\xi_t) \leq \epsilon_R$ for every t from exact initialization, so $\mathcal{I} \subseteq Q_\infty(\mathcal{U})$, and $K_2^\star(\mu)$ is at most the right-hand side of* (5) *with $\rho_R = \rho$.*

Proof. Both parts prove by induction the joint invariant that ξ_t is within ϵ of the identity and ψ_t within 2ϵ, which exact initialization starts, and both show that the chosen primitive is feasible, so nothing is clipped.

(a). The spacing is at most 2ϵ, since $(2c + 4\epsilon)/(\lceil c/\epsilon \rceil + 2) \leq 2\epsilon$ is $c \leq \epsilon \lceil c/\epsilon \rceil$. The invariant puts $W_t = \xi_t + \psi_t + g_t$ in $[-c - 3\epsilon, \ c + 3\epsilon]$, and the grid spans $[-c - 2\epsilon, c + 2\epsilon]$ with spacing at most 2ϵ, so the nearest point u_t leaves $\xi^\circ := W_t - u_t$ with $|\xi^\circ| \leq \epsilon$, the error the step leaves if u_t is executed unclipped. Write $\hat{x}^\circ, \hat{v}^\circ$ for the reconstructed state that step produces, so $x_{t+1} - \hat{x}^\circ = \xi^\circ$ and $v_{t+1} - \hat{v}^\circ = \xi^\circ - \xi_t$. Then $|x_{t+1} - \hat{x}^\circ| \leq \epsilon$, $|v_{t+1} - \hat{v}^\circ| \leq 2\epsilon$ and $|u_t| \leq c + 2\epsilon$, which put $\hat{x}^\circ$ in $[\ell - \epsilon, h + \epsilon]$, $|\hat{v}^\circ|$ below $b + 2\epsilon$ and $\hat{x}^\circ - \hat{v}^\circ = \hat{x}_t$ in the enlarged workspace. The remaining constraint is braking, where the three components of $\mu^\dagger$ act together: enlarging the change bound from c to $c + 2\epsilon$ and the displacement by 2ϵ shortens the stopping distance, because

$$B_{c+2\epsilon}(v + 2\epsilon) = \sum_{i \geq 1} \big(v - ic - (i-1)2\epsilon\big)_+ \ \leq \ \sum_{i \geq 1} (v - ic)_+ = B_c(v),$$

so with B nondecreasing, $\hat{x}^\circ + B_{c+2\epsilon}(\hat{v}^\circ) \le x_{t+1} + \epsilon + B_c(v_{t+1}) \le h + \epsilon$, and symmetrically at the lower end. Hence $u_t \in \mathcal{F}^{\mu^\dagger}(\hat{x}_t, \hat{v}_t)$, $\bar{u}_t = u_t$, $\xi_{t+1} = \xi^\circ$, and $\psi_{t+1} = \xi_{t+1} - \xi_t$ gives $|\psi_{t+1}| \le 2\epsilon$. A larger margin admits the same requests and executes the same trajectory.

(b). Write $\epsilon = \epsilon_R$. Since $q_{t+1} = q_t \delta_t \gamma_t$, the request is $W_t = \hat{\delta}_t^{-1} \xi_t \delta_t \gamma_t = c_{\hat{\delta}_t}(\xi_t)\, \psi_t\, \gamma_t$, and the angle is invariant under conjugation and subadditive, so $\theta(W_t) \le \epsilon + 2\epsilon + c_R$ and $W_t \in \mathcal{B}_R(\rho)$. The cover supplies u_t with $\theta(u_t^{-1} W_t) \le \epsilon$. Executed unclipped, it gives $\xi_{t+1} = u_t^{-1} W_t$, so $\theta(\xi_{t+1}) \le \epsilon$; and $\xi_{t+1} = \hat{\delta}_{t+1}^{-1} \xi_t \delta_{t+1} = c_{\hat{\delta}_{t+1}}(\xi_t)\, \psi_{t+1}$ gives $\theta(\psi_{t+1}) \le 2\epsilon$. Feasibility follows: $\theta(u_t) \le c_R + 3\epsilon$ is within the widened change bound, and $\theta(\hat{\delta}_{t+1}) \le \theta(\delta_{t+1}) + \theta(\psi_{t+1}) \le b_R + 2\epsilon$ is within the widened increment bound, which is all $\mathcal{V}_2^\mu$ asks on a group without boundary. So $\bar{u}_t = u_t$ and the induction closes. The count is the upper bound of Corollary 4, whose maximal ϵ_R-separated subset of $\mathcal{B}_R(\rho)$ is such a $\mathcal{U}$. □

These are the grids of Section 6: part (a) on each translation axis and part (b) on orientation, the latter built as in Corollary 4 from a lattice in logarithmic coordinates, radially projected onto the ball of radius ρ; the projection keeps the cover because projection onto a Euclidean ball is nonexpansive and exp is 1-Lipschitz.

F.3 Computing the optimum on translation

Theorem 7 tests a given primitive set; this section finds the smallest on translation, where every object is a polytope or an interval. We carry the pair $y = (x, v; \hat{x}, \hat{v})$ in the error coordinates (x, v, ξ, ψ) of (11). Two lemmas reduce the search over all finite subsets of $\mathbb{R}$ to a compact box.

Lemma 10 (Polytope form and finite termination). *Let $c > 0$, put $N_b = \lfloor b/c \rfloor + 1$, and for $N = 1, \dots, N_b$ let $f_N(v) = (N-1)v - cN(N-1)/2$. Then $B(v) = \max_{1 \le N \le N_b} f_N(v)$ for every $|v| \le b$, so* (12) *reads*

$$\mathcal{V}_2 = \big\{(x, v) :\ |v| \le b,\ \ x - v \in [\ell, h],\ \ x + f_N(v) \le h,\ \ x - f_N(-v) \ge \ell,\ \ N = 1, \dots, N_b\big\}. \tag{18}$$

Consequently (i) $\mathcal{V}_2$ is a polytope with at most $2N_b + 4$ faces; (ii) writing $\widetilde{\mathcal{V}}_2 = \{(x, v) : x \in [\ell, h], |v| \le b, x - v \in [\ell, h]\}$ and iterating $\mathcal{V}^{(0)} = \widetilde{\mathcal{V}}_2$, $\mathcal{V}^{(m+1)} = \{(x, v) \in \mathcal{V}^{(m)} : \exists |g| \le c,\ (x + v + g, v + g) \in \mathcal{V}^{(m)}\}$, one has $\mathcal{V}^{(m)} = \mathcal{V}_2$ for every $m \ge N_b - 1$; and (iii) $\mathcal{F}(x, v)$ is a nonempty compact interval whose endpoints are finite minima and maxima of affine functions of (x, v). Replacing $([\ell, h], b, c)$ by $([\ell - \mu_x, h + \mu_x], b + \mu_v, c + \mu_g)$ gives $\mathcal{V}_2^\mu$ and $\mathcal{F}^\mu$ in the same form.

Proof. Since $f_{N+1}(v) - f_N(v) = v - Nc$, the sequence $f_N(v)$ increases while $N < v/c$ and decreases afterwards, so its maximum over $N \ge 1$ sits at $N = \lfloor v/c \rfloor + 1$ and equals $\sum_{i=1}^{\lceil v/c \rceil - 1}(v - ic) = B(v)$ for $v > 0$; for $v \le 0$ every $f_N(v) \le f_1(v) = 0 = B(v)$. The state constraint $|v| \le b$ caps the active index at N_b, giving (18) and (i).

For (ii), write $\mathcal{V}^{\mathrm{cl}}$ for the right-hand side of (12). If (x, v) admits $N_b - 1$ admissible steps inside $\widetilde{\mathcal{V}}_2$ and $v > 0$, put $N = \lceil v/c \rceil \le N_b$; from $|g_i| \le c$ we get $v_i \ge v - ic > 0$ for $i \le N - 1$, hence $x_{N-1} = x + \sum_{i \le N-1} v_i \ge x + B(v)$, and admissibility forces $x + B(v) \le h$. The case $v < 0$ is symmetric and $v \le 0$ is trivial, so $\mathcal{V}^{(N_b - 1)} \subseteq \mathcal{V}^{\mathrm{cl}}$. Conversely $\mathcal{V}^{\mathrm{cl}}$ is forward-invariant under braking: for $v > 0$ take $g = -\min\{c, v\}$, so $v' = \max\{v - c, 0\}$ satisfies $v' + B(v') = B(v)$, whence $x' + B(v') = x + v' + B(v') = x + B(v) \le h$, while $v' \ge 0$ gives $x' - B(-v') = x' \ge x \ge \ell$ and $x' - v' = x \in [\ell, h]$. So $\mathcal{V}^{\mathrm{cl}}$ is a set of states with infinite admissible continuations, giving $\mathcal{V}^{\mathrm{cl}} \subseteq \mathcal{V}_2$; with $\mathcal{V}_2 \subseteq \mathcal{V}^{(m)}$ for all m the three sets coincide from $m = N_b - 1$ on.

For (iii), $\mathcal{F}(x, v)$ intersects $[-c, c]$ with the preimage of the polytope (18) under the affine map $g \mapsto (x + v + g, v + g)$, an intersection of finitely many half-lines in g; it is nonempty because $\mathcal{V}_2$ consists of states with an admissible continuation. □

Lemma 11 (The search domain is a compact interval). *Let $\mathcal{J}^\mu = [-c - \mu_g,\ c + \mu_g]$. Replacing any finite $\mathcal{U} \subset \mathbb{R}$ by $\Pi_{\mathcal{J}^\mu}(\mathcal{U})$ changes no executed value at any reconstructed state, hence leaves Q_∞ unchanged, and does not increase cardinality. Restricting the minimization in* (16) *to $\mathcal{U} \subseteq \mathcal{J}^\mu$ therefore loses no optimality.*

Algorithm 3 Optimal second-order primitive set for the translational instance

procedure OPTIMUM($\ell, h, b, c, \epsilon, \mu$)
Require: $c > 0$, $\epsilon > 0$, $\mu \geq 0$
  build $\mathcal{V}_2, \mathcal{V}_2^{\mu}$ and $\mathcal{F}, \mathcal{F}^{\mu}$ from (18) # Lemma 10: polytopes, $\lfloor b/c \rfloor$ steps
  $\mathcal{J}^{\mu} \leftarrow [-c - \mu_g,\ c + \mu_g]$ # Lemma 11: no loss of optimality
  $lo \leftarrow \max\{1, \lceil L/2\epsilon \rceil\}$ # (20)
  if $\mu \geq \mu^{\dagger}$ **then**
    $hi \leftarrow \lceil c/\epsilon \rceil + 3$; $\mathcal{U}^{\star} \leftarrow$ the grid of Proposition 9(a)
  else
    $hi, \mathcal{U}^{\star} \leftarrow$ any certified feasible set # Proposition 9 does not apply
  while $lo < hi$ **do**
    $K \leftarrow \lfloor (lo + hi)/2 \rfloor$
    $(status, \mathcal{U}) \leftarrow$FEASIBLE($K$) # branch and bound over $(\mathcal{J}^{\mu})^K$
    if $status =$ YES **then**
      $hi, \mathcal{U}^{\star} \leftarrow K, \mathcal{U}$
    else if $status =$ NO **then**
      $lo \leftarrow K + 1$
    else return $[lo, hi]$, $\mathcal{U}^{\star}$ # certified integer gap; no branch may be taken
  return lo, $\mathcal{U}^{\star}$, $Q_{\infty}(\mathcal{U}^{\star})$

Proof. $\mathcal{F}^{\mu}(\hat{x}, \hat{v}) \subseteq \mathcal{C}^{\mu} = \mathcal{J}^{\mu}$ at every reconstructed state. For nested closed intervals $I \subseteq J$ the projections compose, $\Pi_I = \Pi_I \circ \Pi_J$; taking $I = \mathcal{F}^{\mu}(\hat{x}, \hat{v})$ and $J = \mathcal{J}^{\mu}$ gives $\Pi_I(u) = \Pi_I(\Pi_{\mathcal{J}^{\mu}}(u))$, so u and its projection execute identically at every state. □

Testing a fixed $\mathcal{U}$ is exact. The executed value $\bar{u}_k = \Pi_{\mathcal{F}^{\mu}(\hat{x},\hat{v})}(u_k)$ does not depend on g, so every coordinate of $y^{+}(g, u_k)$ is affine in g, and the real changes answered by u_k,

$$\mathcal{F}_k(y; Q_n) := \{g \in \mathcal{F}(x, v) :\ y^{+}(g, u_k) \in Q_n\}, \tag{19}$$

form a finite union of closed intervals, computable exactly by Lemma 10. The deletion test of (14) at y is then the one-dimensional covering $\mathcal{F}(x, v) \subseteq \bigcup_{k \leq K} \mathcal{F}_k(y; Q_n)$. An implementation that retains whole cells $\mathcal{N}$ of states must keep the two quantifiers joint: it subdivides $\{(y, g) : y \in \mathcal{N},\ g \in \mathcal{F}(x, v)\}$ and certifies by interval arithmetic that each piece is answered by some label, rather than asking one label to answer a given g at every state of the cell.

Searching is a branch and bound. By Lemma 11 the candidates at budget K form the compact box $\{u \in (\mathcal{J}^{\mu})^K : u_1 \leq \cdots \leq u_K\}$. For a sub-box $X_1 \times \cdots \times X_K$, letting each use of the kth label re-select its value anywhere in X_k gives a relaxation at least as capable as any $\mathcal{U}$ in the box, so if its kernel misses $\mathcal{I}$ the whole box is pruned soundly; the relaxed vocabulary is infinite, so its kernel serves only as the outer approximation that pruning needs. A concrete $\mathcal{U}$ that passes the certified test is a witness. Feasibility at budget K is monotone in K, so bisection on K is sound, and both endpoints are explicit. For the lower one put $L = \max_{(x,v) \in \mathcal{V}_2} |\mathcal{F}(x, v)|$, attained because the graph of $\mathcal{F}$ is compact and its length is upper semicontinuous; at exact initialization $\xi^{+} = g - \bar{u}$, and each executed value keeps at most an interval of length 2ϵ of $\mathcal{F}(x, v)$ inside the tolerance, whence

$$K_2^{\star}(\mu) \ \geq\ \max\{1,\ \lceil L/2\epsilon \rceil\}. \tag{20}$$

The upper one is Proposition 9(a) when $\mu \geq \mu^{\dagger}$, and the two then bracket the answer within three labels.

Algorithm 3 assembles these pieces. When it returns a single integer, the witness $\mathcal{U}^{\star}$ with $Q_{\infty}(\mathcal{U}^{\star}) \supseteq \mathcal{I}$ and a certified prune of the whole box at budget $K^{\star} - 1$ together certify optimality, and the encoder is read off the kernel by part (2) of Theorem 7. If the optimum sits on a non-robust boundary, no finite refinement need reach either verdict, and the algorithm returns the surviving integer interval rather than reading a failed search as infeasibility.

On $\mathrm{SO}(3)$ the test of Theorem 7 and the grid of Proposition 9(b) hold, but the search does not transfer. Lemma 11 rests on projections onto nested intervals composing, which has no analogue for nearest-point projection onto nested geodesic balls, so there is no compact domain to branch over, and $K_2^{\star}$ on $\mathrm{SO}(3)$ is bounded by Proposition 9(b) rather than computed.

G COMPARISON WITH CHUNK TOKENIZERS

Every chunk tokenizer buys its compression the same way: it waits. The policy buffers τ control steps, the tokenizer encodes the whole block jointly, and the block is released as a unit; a streaming tokenizer must answer every control step the instant it arrives. The question is blunt: *what does answering immediately cost?* In the worst case, almost nothing. At $\beta_j = 4$, for example, waiting saves at most 0.32 bits per step per axis (Proposition 12). And the two deployed chunk families never collect even that: a FAST-style transform emits the same number of labels as streaming, delivered $\tau - 1$ steps later, and pays extra bits for the privilege (Theorem 13); a learned codebook holds its budget of distinct blocks fixed before training, and at realistic sizes that budget covers an astronomically small fraction of the motions a robot may legally perform ($\leq 2^{-370}$, (23)). The paragraphs below prove these statements.

Setup. Everything is stated for the box model of Section 4: increments in $\mathcal{A} = \prod_j [-b_j, b_j]$, per-step tolerance $\mathcal{E} = \prod_j [-\epsilon_j, \epsilon_j]$, and the ratio $\beta_j = b_j/\epsilon_j$—how many tolerance widths one control step can travel along axis j. Two volume ratios recur, $\mathcal{B} = \prod_j \beta_j = \mathrm{Vol}(\mathcal{A})/\mathrm{Vol}(\mathcal{E})$ and $\mathcal{B}_+ = \prod_j (1 + \beta_j)$. The streaming optimum is known exactly (Corollary 3): $K^\star(1) = \prod_j N_j$ labels, $N_j = \lceil 1 + \beta_j \rceil$, at $R(1) = \sum_j \log_2 N_j$ bits per step. For a chunk decoder releasing τ steps at a time, $K^\star(\tau)$ is the smallest dictionary of reconstructed blocks that covers within tolerance, at every step of the block, every legal path (residual in $\mathcal{E} + \mathcal{A}$, increments in $\mathcal{A}$), the final constraint returning the residual to $\mathcal{E}$ so that blocks concatenate indefinitely; its rate is $R(\tau) = \tau^{-1} \log_2 K^\star(\tau)$, and $R(1) - R(\tau)$ is exactly what a delay of $\tau - 1$ steps buys. Nothing assumes a memoryless decoder: the legal paths do not depend on the state the previous block left behind, so every decoder state faces the same covering problem and the bounds bind every architecture.

Waiting saves at most a rounding error. If waiting were powerful, long blocks would amortize the per-step cost away; they cannot, because every buffered step still costs its own $\log_2 \mathcal{B}$.

Proposition 12 (Waiting cannot amortize a step). *$K^\star(1) = \prod_j N_j$ and $K^\star(\tau + \tau') \leq K^\star(\tau) K^\star(\tau')$, so $R(\tau) \leq R(1)$ and $R(\tau) \to R_\infty = \inf_\tau R(\tau)$; moreover*

$$K^\star(\tau) \;\geq\; \mathcal{B}_+ \, \mathcal{B}^{\tau - 1}, \qquad \textit{hence} \qquad \log_2 \mathcal{B} \;\leq\; R(\tau) \;\leq\; R(1),$$
$$\textit{so} \qquad R(1) - R_\infty \;\leq\; \sum_{j=1}^{d} \log_2 \frac{\lceil 1 + \beta_j \rceil}{\beta_j} \;\leq\; \sum_{j=1}^{d} \log_2 \Big(1 + \frac{2}{\beta_j} \Big). \tag{21}$$

Proof. Concatenating an optimal τ-code with an optimal τ'-code covers the $(\tau + \tau')$ paths, so $\log_2 K^\star$ is subadditive and Fekete's lemma applies. Each reconstruction covers one translate of $\mathcal{E}^\tau$, while the legal paths shear to $(\mathcal{E} + \mathcal{A}) \times \mathcal{A}^{\tau - 1}$, so covering them forces $K^\star(\tau) \, \mathrm{Vol}(\mathcal{E})^\tau \geq \mathrm{Vol}(\mathcal{E} + \mathcal{A}) \, \mathrm{Vol}(\mathcal{A})^{\tau - 1}$; divide by $\mathrm{Vol}(\mathcal{E})^\tau$ and take $\tau^{-1} \log_2$, using $\lceil 1 + \beta \rceil \leq \beta + 2$ for the last line. □

For example, at $\beta_j = 4$ the value of waiting is 0.32 bits per step per axis, against a streaming cost of 2.32. Only when $\beta_j < 1$—one step cannot cross its own tolerance—does the integer ceiling $N_j = 2$ rather than the physics set $R(1)$ and the bound lose force; the deployed tokenizers do not operate there. Neither chunk family in fact lands inside the band at all: as the next two paragraphs show, one pays more than its top, the other covers too little to count.

FAST: the same labels, later, plus a rotation penalty. FAST [29] normalizes a block, applies a temporal DCT, rounds the coefficients, and runs BPE. The DCT is invertible, so the quantization stage emits exactly τ integers per axis per block—one per control step, the same count as streaming, delivered $\tau - 1$ steps late. The token saving FAST reports comes entirely from its BPE stage, which is lossless: that is the layer of Section 5, it applies verbatim to our stream too, so the fair comparison is at the physical layer. There the transform is a liability: rounding in the frequency basis makes the error cell a cube rotated by the DCT, and a rotated cube must shrink before it fits inside the axis-aligned tolerance box.

Theorem 13 (A FAST-style quantizer is dominated by streaming). *Let $\Lambda_\tau := \max_l \sum_k |C_{kl}|$ be the largest column ℓ_1 norm of the orthonormal DCT-II; $\sqrt{\tau/2} \le \Lambda_\tau \le \sqrt{\tau}$. If the quantizer $\hat{q} = \alpha^{-1} C^\top \operatorname{round}(\alpha C q)$, with any per-axis scales α_j, meets the per-step tolerance on every legal path, then it spends at least*

$$R_{\mathrm{DCT}}(\tau) - R(1) \;\ge\; \frac{d}{2}\log_2\frac{\tau}{2} - \sum_{j=1}^{d}\log_2\Big(1+\frac{2}{\beta_j}\Big) \tag{22}$$

bits per control step more than streaming.

Proof. The rounding error e obeys $|e_k| \le \alpha^{-1}/2$ and $\hat{q} - q = C^\top e$, so $\|\hat{q} - q\|_\infty \le \alpha^{-1}\Lambda_\tau/2$, and the extremal pattern is realized on legal paths, so the tolerance forces $\alpha_j^{-1} \le 2\epsilon_j/\Lambda_\tau$: the transform must quantize Λ_τ times finer than an axis-aligned grid. Cells of volume $\prod_j \alpha_j^{-\tau}$ must then cover the legal paths, of volume $\prod_j (2b_j + 2\epsilon_j)(2b_j)^{\tau-1}$, so at least $\Lambda_\tau^{d\tau}\mathcal{B}_+\mathcal{B}^{\tau-1}$ reconstructions are used; take $\tau^{-1}\log_2$ and subtract $R(1) \le \log_2 \mathcal{B} + \sum_j \log_2(1 + 2/\beta_j)$, using $\Lambda_\tau \ge \sqrt{\tau/2}$. □

At $\beta_j = 4$ on all seven axes, (22) is ≥ 2.9 bits per step already at $\tau = 8$ and ≥ 13.4 at $\tau = 64$; no per-axis scale removes it, because Λ_τ grows with τ.

Learned codebooks: a budget fixed before training. Learned chunk codebooks—the OAT [20], X-Tokenizer [13], and ACodec [9] baselines—encode a block into m symbols over a vocabulary of size S and decode with a learned network. The compression sounds informational, $m \ll \tau$; it is not. However the network is trained, it can emit at most $M \le S^m$ distinct blocks, each covering one translate of the tolerance box, so the fraction of legal paths reconstructible within tolerance is

$$\gamma_\theta \;\le\; \frac{M}{\mathcal{B}_+\mathcal{B}^{\tau-1}} \tag{23}$$

for every θ—fixed before training begins. At realistic sizes the shortfall is astronomical: with $S = 1024$, $m \le 8$, and a 32-step block on seven axes at $\beta_j = 4$, $\gamma_\theta \le 2^{-370}$. Training only chooses where those few covered cells sit: split the paths into a training region and its complement and the two covered fractions trade off linearly, so accuracy at home is bought with coverage everywhere else. This is why every learned-codebook baseline in our study arrives prematched to its evaluation corpus—OAT is learned from scratch on each dataset, X-Tokenizer is fine-tuned on it, and the released ACodec is pretrained on exactly LIBERO, BridgeData, and DROID (Appendix D.1)—while the pretrained streaming instance transfers zero-shot across embodiments. A worst-case guarantee, by (23), forces $M \ge K^\star(\tau)$: the streaming rate.

In plain terms. The bit cost of safe tokenization is set by physics—how far one step can move and how tight the tolerance is—not by how labels are grouped. Waiting regroups labels and saves at most the rounding error (0.32 bits per step per axis). Changing the basis cannot change physics either: FAST emits the same labels later and pays a rotation penalty (2.9–13.4 bits per step at $\beta_j = 4$). A learned codebook spends a budget fixed before training and lets the data place it: strong at home, unbounded away ($\le 2^{-370}$ of legal paths). Streaming composition pays the physics price up front—at most $\sum_j \log_2\big(\lceil 1 + \beta_j\rceil/\beta_j\big)$ bits per step above any tokenizer on any release schedule—and holds its guarantee everywhere the robot can legally go.

H LOSSLESS PACKING AFTER PHYSICAL TOKENIZATION

Everything here starts from the already fixed primitive sequence: a packer may change how that sequence is written for an AR model, but not which primitive was chosen or which state was reconstructed.

At step t the emitted token instances T_t partition the step's joint assignment,

$$\biguplus_{C \in T_t} \mathrm{Exp}(C) = S_t, \tag{24}$$

every atom covered exactly once, and the weighted count of a pair of token types over training steps is

$$F(A, B) = \sum_t w_t \, \mathbf{1}\{A \in T_t,\ B \in T_t\}. \tag{25}$$

A temporal instance starts at a step a and covers occurrences at nonnegative offsets within the horizon τ; the stream orders the final instances by start step and, at equal starts, by the smallest coordinate of the leading spatial token.

Proposition 14 (Set-BPE is lossless and entropy-preserving). *Every merge of Algorithm 2 replaces two token instances by one whose expansion is their disjoint union. The emitted instances therefore partition the episode's assignments, the stream is a strict total order that the decoder inverts without timestamps or separators, the trained packer is a bijection with $H(Y) = H(Z_{1:T})$ for every source distribution and with the reconstructed trajectory of Corollaries 3 and 4 unchanged, and each applied merge removes exactly one emitted token per affected occurrence, so the greedy scores in Algorithm 2 are the exact one-step gains.*

Proof. A spatial merge replaces two disjoint expansions by their union, so (24) keeps returning S_t and the step holds one token fewer; a temporal merge replaces two disjoint subsets of the episode's occurrences by their union, so induction from the singletons keeps the emitted instances a partition. Two final instances never share a leading occurrence, and at a fixed start the leading tokens have disjoint coordinate supports, so the serialization order is strict. An instance covers only steps from its start onward, so when the earliest incomplete step is a, the next instance in order leads there and its root sits at the lowest coordinate of a still unassigned; decoding therefore advances over completed steps with no timestamp or separator, and expansion restores every S_t. The packer is thus injective, the trajectory and its distribution are unchanged, and each merge's drop in emitted tokens is exact: a spatial rule's weighted count falls by exactly $F(A, B)$, and the largest reduction of a temporal rule (P, R, δ) is the maximum matching of its instance graph, which for a fixed offset has degree at most two—time-ordered paths on which left-to-right greedy selection is exact. □

Appendix C describes the exact sparse and lazy implementations of this loop.