LEARNING REUSABLE MOTION PRIMITIVES BY SEPARATING IDENTITY FROM EXECUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Complex skills can be viewed as organized compositions of reusable motion primitives. Learning such primitives therefore requires an explicit notion of equivalence: when do different motion trajectories instantiate the same primitive? Recent action-representation research spans action chunking, tokenization, and latent-action learning, with increasing attention to shaping latent-action semantics; yet these advances do not by themselves specify what defines reusable primitive identity. To make this equivalence explicit, we define primitive identity as the ordered structure of canonical basic motions, while execution captures how that structure is realized through local transformations, relative rhythm, and overall temporal scale. We instantiate this separation in a hierarchical model that canonicalizes local motion geometry, encodes the ordered structure of the corresponding basic motions, and estimates execution factors separately. We evaluate the resulting representation by testing cross-execution reuse, preservation of the complete identity-defining structure, and identity–execution recombination, and further test whether the model can detect and identify primitive occurrences within longer, unsegmented trajectories. On a controlled benchmark of variable-length primitives, the model recovers both primitive structure and execution from isolated recordings and recovers primitive boundaries and identities within longer trajectories composed from a known set of primitives. Two comparison settings reveal distinct gaps. A fixed-duration chunking interface is poorly aligned with the variable temporal support of primitive instances. Even with oracle primitive support, reconstruction-based models with primitive-level semantic shaping—including a factorized extension with supervised execution—do not satisfy the required representation responsibilities: cross-execution reuse, complete-structure preservation, and recombination. Together, these results show that defining primitive identity separately from execution yields a learnable representation with a directly testable criterion for recognizing the same primitive across execution variations and across occurrences within longer trajectories.

1 INTRODUCTION

A longstanding view in motor control and skill learning is that complex behavior can be constructed through hierarchical and modular organization of reusable movement elements (Bizzi et al., 2002; Diedrichsen & Kornysheva, 2015; Giszter, 2015). This perspective motivates motion primitives as a representation between low-level motion trajectories and higher-level skill organization. Yet reuse presupposes identity: before asking how primitives compose into longer behaviors, we must determine when different motion trajectories instantiate the same primitive. Reusable primitive learning therefore requires an explicit equivalence criterion—what must remain invariant across executions, and what is allowed to change?

Existing work explores multiple ways to construct action representations. Action chunking predicts multiple controls over a prescribed horizon (Zhao et al., 2023); action tokenizers map action sequences into discrete symbols or learned codes (Lee et al., 2024; Pertsch et al., 2025; Wang et al., 2025); and latent-action models can also learn action-like variables from videos without ground-truth robot actions (Bruce et al., 2024; Ye et al., 2025). For latent-action models in particular, what these variables capture can depend strongly on the data and learning signals used to obtain them (Zhang et al., 2025), while recent methods explicitly introduce consistency or action-alignment objectives

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.

to shape their semantics (Chen et al., 2026; Luo et al., 2026). These choices can yield useful action representations, but they do not by themselves define when different executions instantiate the same reusable primitive.

We make this criterion explicit by representing each primitive instance in terms of structure and execution, with identity defined by structure. This broader idea has precedents in motor-control theory, where relatively invariant movement features or abstract movement structure have been used to characterize movement classes amid parameter variation (Heuer, 1991; Shea & Wulf, 2005). We write a primitive instance as $p = (s, e)$, where s is the ordered structure of canonical basic motions and e specifies how that structure is realized through local transformations, relative rhythm, and overall temporal scale. We define an equivalence relation $\sim$ over primitive instances by $p_A \sim p_B \iff s_A = s_B$, so execution may vary without changing primitive identity as long as the ordered structure is preserved.

This definition yields concrete tests of whether reusable primitive identity can be learned. We test whether the model preserves the ordered structure that defines identity, supports cross-execution reuse, and allows identity–execution recombination. We view reusable primitive identities as candidate building blocks for modeling the organization of longer behavior. Under this view, the same primitive identities should remain recoverable when instances occur at different positions within longer trajectories rather than being presented in isolation. Accordingly, at the system level, we test whether the model can locate primitive boundaries and assign their identities within such trajectories. Together, these tests ask whether the proposed equivalence defines a primitive identity that remains reusable across execution variation and across occurrences in longer trajectories.

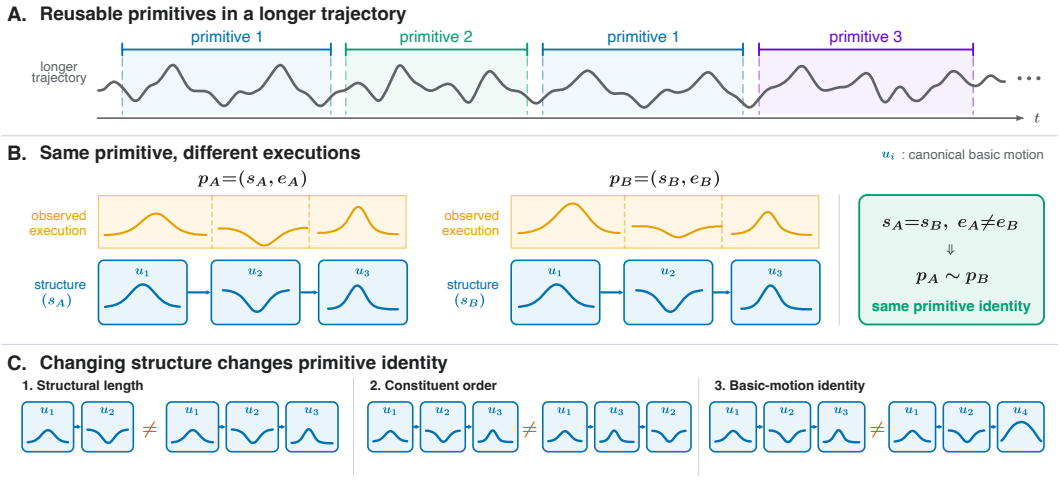

Figure 1: Reusable primitive identity separates structure from execution. **(A)** Reusable primitive identities recur within a longer trajectory in this motivating schematic; higher-level skill organization is not learned here. **(B)** Instances share primitive identity when their ordered structure agrees despite different executions. Orange curves schematically show observed realizations under different execution factors; blue glyphs show the shared canonical basic-motion structure. **(C)** Structure changes with structural length, constituent order, or constituent basic-motion identity. Each u_i comprises canonical local-motion geometry and its associated actor; waveform glyphs schematically depict the geometry.

Guided by this criterion, we build a hierarchical model with a learned structure path and a separate execution estimator. The structure path learns a representation of the ordered structure that defines primitive identity. It first canonicalizes the geometry of each constituent local motion relative to a specified restricted transformation family and represents each basic motion using its canonical geometry together with the associated actor. The resulting basic-motion representations are then aggregated in order to form the primitive-level structure representation. The execution estimator separately recovers the transformation and timing factors that specify how the structure was executed. This explicit separation avoids leaving the distinction between identity and execution to emerge implicitly within a single latent representation. The resulting structure representation is used to match

primitive identity across executions, while the explicit execution factors describe how each instance was realized and support identity–execution recombination. At the system level, we extend the model to detect and identify occurrences of these reusable primitives within longer, unsegmented trajectories while preserving the same notion of identity.

On isolated primitive recordings, the experiments show that our model recovers the intended primitive factorization. The learned structure representation preserves the ordered structure that defines primitive identity across execution variation, while the explicit execution factors support identity–execution recombination. The same model also recovers primitive boundaries and identities when multiple known primitive instances are composed into longer trajectories. Two comparison settings reveal distinct limitations. A fixed-duration chunking interface can be poorly aligned with the variable temporal support of primitive instances. Even when the correct primitive support is supplied, reconstruction-based models with primitive-level semantic shaping—including a factorized extension with supervised execution—do not satisfy the required representation responsibilities: cross-execution reuse, complete-structure preservation, and identity–execution recombination. Taken together, the paper contributes an explicit definition of reusable primitive identity, a model that learns the identity-defining structure while estimating execution separately, and controlled evidence that the learned structure representation supports identity reuse across executions while the model recovers those identities within longer trajectories.

2 Related Work

Temporal action abstractions and learned semantics. Action abstraction spans fixed-horizon chunks and vector-quantized interfaces (Zhao et al., 2023; Lee et al., 2024), as well as variable-span skills learned through sequence compression (Zheng et al., 2024). Recent work also analyzes or explicitly structures the semantics of action and skill representations through latent-content analysis, semantic-consistency objectives, and reference-grounded skill discovery (Zhang et al., 2025; Chen et al., 2026; Rho et al., 2026). Within this broader space, we ask a more specific question: what equivalence relation determines whether two primitive instances share identity? We formalize this criterion and derive representation and evaluation responsibilities for identity and execution from it.

Movement primitives, reuse, and invariance. Classical DMPs provide reusable trajectory generators with temporal and spatial modulation (Ijspeert et al., 2013), while ProMPs model trajectory distributions that support conditioning and temporal scaling (Paraschos et al., 2013). More recent work addresses structured motion variability: EMMP learns motion manifolds that are equivariant with respect to prescribed task symmetries (Lee et al., 2023), while VLBiMan preserves invariant primitives as anchors and adapts adjustable components under scene variation (Zhou & Jia, 2026). These works provide close precedents for reusable motion and for distinguishing preserved from variable aspects of behavior. Against this background, our formulation specifies the equality criterion itself: primitive identity is defined by the ordered canonical basic-motion structure, so two primitive instances are equivalent when that structure agrees; permitted local geometric and temporal variation belongs to execution.

Segmentation and primitive recovery in unsegmented behavior. Work on temporal segmentation and skill/primitive discovery asks how coherent units and their boundaries can be identified in longer, unsegmented behavior. Open-World Skill Discovery detects semantic skill boundaries without boundary annotations (Deng et al., 2025), while LAPS discovers and segments coherent action primitives from continuous video (Zhang et al., 2026). We instead ask whether the same primitive identities can be recovered in longer trajectories using representations learned from isolated recordings, given a fixed primitive bank but no primitive-recording boundaries. This yields a primitive-identity sequence that can serve as input to the next hierarchical problem of modeling how primitives are organized into skills, which we leave to future work.

3 Defining and Learning Reusable Motion Primitives

We define a primitive instance by separating identity-defining structure from execution, and use this definition to specify the corresponding representation responsibilities. We then build a hierarchical model that learns a representation of structure while separately estimating execution factors. Finally,

we apply the same primitive-identity criterion to recover occurrences of known primitives in longer, unsegmented trajectories.

3.1 Primitive Definition: Structure, Execution, and Responsibilities

Structure and primitive identity. We represent a primitive instance as $p = (s, e)$, where s denotes its structure and e its execution. Primitive identity is defined by structure: two primitive instances share the same identity if and only if their structures agree,

$$p_A \sim p_B \iff s_A = s_B. \tag{1}$$

Thus execution may vary across instances without changing primitive identity as long as the structure is preserved. The structure is an ordered, variable-length sequence of canonical basic motions,

$$s = (u_1, \ldots, u_K). \tag{2}$$

Each canonical basic motion u_i pairs an associated actor f_i with a canonical local-motion geometry. We represent this geometry by a canonical motion function ϕ_i^S, and encode the function by coordinates c_i^S in a shared motion representation, writing

$$u_i = (f_i, c_i^S). \tag{3}$$

Because structure is an ordered sequence, changing a constituent basic motion, the ordering of constituents, or the sequence length changes s.

Execution of a fixed structure. For a primitive with fixed structure, execution captures two kinds of variation: how each canonical basic motion is locally realized, and how the ordered basic motions unfold in time. For basic motion u_i, let x_i denote its observed local motion. We express this realization as

$$x_i = T_{\eta_i}[\phi_i^S], \qquad T_{\eta_i} \in \mathcal{G}, \tag{4}$$

where η_i denotes the local execution of u_i, T_{η_i} is the corresponding transformation, and $\mathcal{G}$ is a deliberately restricted family of identity-preserving transformations. Restricting $\mathcal{G}$ ensures that local execution cannot account for motion differences that should instead change structure.

The temporal organization of the primitive is defined through semantic anchors. Each canonical basic motion has a semantic anchor defined at a fixed location in its canonical local-motion geometry, and τ_i denotes the realized time of this anchor in the primitive instance, with

$$\tau_1 < \cdots < \tau_K. \tag{5}$$

We define the temporal scale L of the primitive instance as the span between its first and last realized anchors,

$$L = \tau_K - \tau_1. \tag{6}$$

Relative rhythm ρ describes how this anchor span is distributed between successive anchors:

$$\rho = (\Delta_1, \ldots, \Delta_{K-1}), \qquad \Delta_i = \frac{\tau_{i+1} - \tau_i}{L} > 0, \qquad \sum_{i=1}^{K-1} \Delta_i = 1. \tag{7}$$

Thus $\eta_{1:K}$ describes local basic-motion execution, while ρ and L describe the relative timing and temporal scale of the primitive instance. The execution of a primitive instance is therefore written as

$$e = (\eta_{1:K}, \rho, L). \tag{8}$$

Representation responsibilities. The structure–execution distinction specifies the corresponding representation responsibilities. The structure representation should preserve the ordered canonical basic-motion structure that defines primitive identity and remain stable across primitive instances that share the same structure. Execution information should remain explicit as the local execution, relative rhythm, and temporal scale of each primitive instance.

These responsibilities make the factorization directly testable. We test whether the structure representation recovers the ordered basic-motion composition despite execution variation, and whether structure and execution can be recombined across instances.

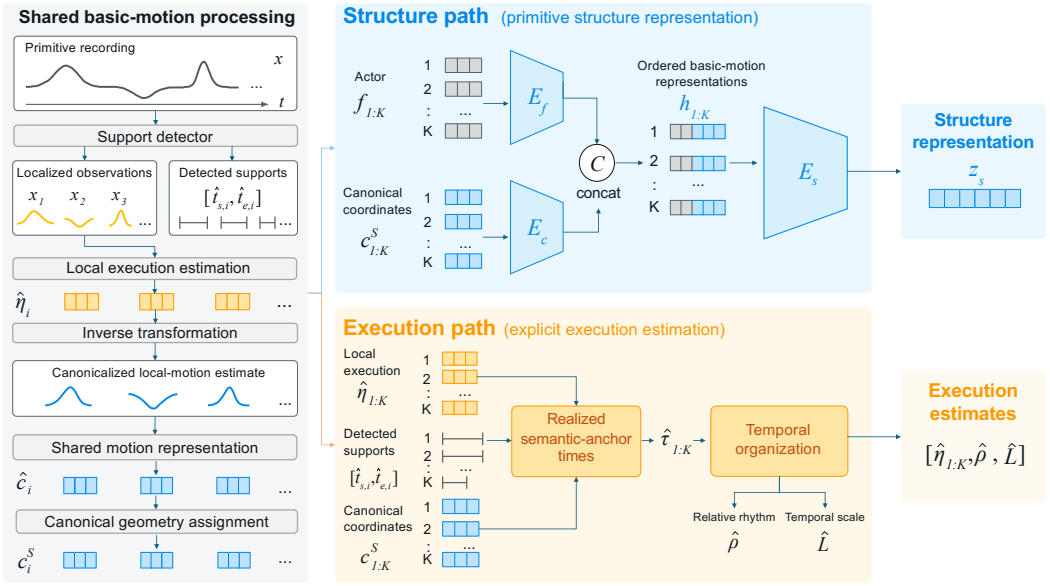

Figure 2: **Primitive-recording inference separates structure from execution.** Shared basic-motion processing detects constituent temporal supports, estimates local execution, canonicalizes the localized motions, and assigns each observation-derived coordinate $\hat{c}_i$ to a reusable canonical geometry represented by c_i^S. The structure path separately encodes actor identity and canonical coordinates and aggregates their ordered representations into z_s, while the execution path uses detected supports, local execution estimates, and canonical coordinates to obtain realized semantic-anchor times $\hat{\tau}_{1:K}$, relative rhythm $\hat{\rho}$, and temporal scale $\hat{L}$. Inference returns $[z_s, \hat{\eta}_{1:K}, \hat{\rho}, \hat{L}]$.

3.2 Learning Structure and Estimating Execution

Hierarchical model. Building on the definition above, we introduce a hierarchical model organized at the basic-motion and primitive levels (Fig. 2). The basic-motion level provides a canonical representation of each constituent motion, together with its local execution estimate and realized semantic-anchor time. At the primitive level, the structure path encodes the canonical basic motions and aggregates their ordered representations into z_s, while the execution path combines the local execution estimates with the realized anchor times to characterize primitive execution. The model therefore outputs a learned structure representation z_s together with explicit execution estimates $(\hat{\eta}_{1:K}, \hat{\rho}, \hat{L})$.

Inference from a primitive recording. Given a recording of one primitive instance, the support detector localizes the K constituent motions and returns their temporal supports $\{[\hat{t}_{s,i}, \hat{t}_{e,i}]\}_{i=1}^{K}$, which define the localized observations x_i. For each x_i, the model estimates its local execution $\hat{\eta}_i$. By inverting the estimated transformation in Eq. 4, the model obtains a canonicalized estimate of the local motion. We represent this estimate by coordinates $\hat{c}_i$ in the shared motion representation. The inferred coordinate $\hat{c}_i$ is then assigned to one of the canonical local-motion geometries estimated from the training data. We denote the coordinate of the assigned geometry by c_i^S; together with the associated actor f_i, it forms the canonical basic motion $u_i = (f_i, c_i^S)$ passed to the structure path.

Structure path. For each recovered canonical basic motion $u_i = (f_i, c_i^S)$, the associated actor and canonical-motion coordinate are encoded separately and concatenated to form $h_i = \big[E_f(f_i); E_c(c_i^S)\big]$. An order-sensitive structure encoder E_s then maps the ordered, variable-length sequence $(h_1, \ldots, h_K)$ to a fixed-dimensional primitive structure representation, $z_s = E_s(h_1, \ldots, h_K)$.

Execution path. For each recovered basic motion, its detected temporal support $[\hat{t}_{s,i}, \hat{t}_{e,i}]$, estimated local execution $\hat{\eta}_i$, and assigned canonical local-motion geometry, represented by coordinates c_i^S,

determine the realized semantic-anchor time $\hat{\tau}_i$. The ordered anchor times $\hat{\tau}_{1:K}$ are then used in Eqs. 6 and 7 to obtain the primitive-level temporal organization, represented by the relative rhythm $\hat{\rho}$ and temporal scale $\hat{L}$. Together with the retained local execution estimates $\hat{\eta}_{1:K}$, these quantities describe the inferred execution of the primitive instance.

Combining the structure and execution paths, inference from a primitive recording returns $[z_s, \hat{\eta}_{1:K}, \hat{\rho}, \hat{L}]$.

Staged learning. The components are learned or fitted in stages. We first train a shared support detector using temporal-support annotations in the training data. Applied to the training recordings, the detector provides the basic-motion supports used to obtain a coordinate estimate $\hat{c}_i$ for each motion in the shared motion representation. The training estimates are grouped into canonical local-motion geometries, and the resulting assignment rule is retained. For each group, the assigned motions are jointly used to estimate a reusable canonical coordinate c^S.

Next, we train the two basic-motion encoder branches. E_c takes $\hat{c}_i$ as input and uses the corresponding c_i^S as a canonical anchor, so that different executions of the same canonical motion are represented consistently. E_f is trained to encode actor identity, and the two outputs are concatenated into the training-time basic-motion representation $\hat{h}_i = [E_f(f_i); E_c(\hat{c}_i)]$. With E_f and E_c fixed, E_s aggregates the ordered sequence $(\hat{h}_1, \ldots, \hat{h}_K)$ into z_s. The corresponding canonical sequence, obtained by replacing $\hat{c}_i$ with c_i^S, serves as the target for a weak slot-conditioned decoder that reconstructs the ordered sequence and predicts its length. Together with an invariance objective across executions sharing the same structure, this encourages z_s to retain the canonical ordered structure rather than execution-specific variation.

3.3 Closed-Bank Primitive Recovery in Longer Trajectories

To recover primitive occurrences from longer trajectories, we apply the hierarchical model without assuming primitive boundaries. Given a trajectory and a known primitive bank, the model first localizes individual motion observations, recovers a canonical basic motion from each, and encodes them into an ordered sequence of representations $h_{1:M}$. The remaining task is to determine how this sequence should be grouped into occurrences of these known primitives.

For each candidate span in this sequence, E_s produces a structure representation z_s. We compare z_s with primitives of the same structural length in the known bank to identify the corresponding primitive. Using these matches, we recover an ordered sequence of primitive occurrences; the detailed recovery procedure is provided in Algorithm 3. For each recovered occurrence, we retain the local execution estimates of its basic motions and obtain the corresponding relative rhythm and temporal scale through the same execution path. By representing a longer trajectory as an ordered sequence of reusable primitive occurrences, the model provides a natural interface for future work on how primitives are organized into higher-level skills.

4 Experiments

We evaluate the proposed hierarchical model in three stages. We first test the definition-derived responsibilities of primitive identity and execution under controlled interventions, then evaluate integrated inference from complete primitive recordings, and finally test whether the same primitive identities remain recoverable in longer trajectories without primitive boundaries.

4.1 Experimental Setting and Comparison Design

Experimental setting. We evaluate on controlled synthetic primitive instances composed of $K \in \{2, 3, 4, 5\}$ ordered canonical basic motions, where K is the structural length. Each canonical basic motion pairs one of three observed actors with one of three canonical local-motion geometries. Primitive structure varies in structural length, canonical basic-motion identity, and constituent order, while execution varies through local amplitude and local duration, relative rhythm, and overall temporal scale. Primitive-structure training uses 10,008 representation-level instances spanning 400 unique ordered structures after train-only order augmentation.

Model and training. The hierarchical model uses a shared bidirectional GRU (Schuster & Paliwal, 1997; Cho et al., 2014) for support detection, separate actor and geometry encoders for canonical basic motions, and a Transformer (Vaswani et al., 2017) to encode primitive structure. We represent local-motion geometry with cubic B-spline coefficients (Zhou et al., 2025) and fit the reusable canonical geometries from the training data. Local execution and primitive-level temporal organization are recovered analytically. The basic-motion encoders are trained and frozen before training the primitive structure encoder. The primitive structure encoder is trained with a weak slot-conditioned readout that reconstructs the ordered canonical basic-motion representations and predicts structural length.

Comparison design. We evaluate two distinct questions: whether a temporal unit aligns with the temporal support of a primitive instance, and, once the correct primitive support is given, whether representations learned through reconstruction and primitive-level semantic shaping satisfy the required representation responsibilities. A globally calibrated fixed-width temporal partition tests how well it aligns with the varying temporal support of primitive instances. For the representation-level comparison, all comparison models receive oracle primitive support, thereby removing temporal-support mismatch. We evaluate two semantically shaped alternatives. Sem-AE augments continuous reconstruction with a primitive-identity objective that brings together different executions of the same structure while separating different structures. Sem-Factor uses the same structure-semantic objective together with a separate execution latent supervised by the execution factors. Neither semantic comparison is trained with direct supervision of the ordered constituent structure. Reconstruction-only continuous and vector-quantized controls are retained as secondary comparisons in the appendix.

Further details on benchmark construction, model architectures and training, and comparison-model implementations are provided in Appendix B; experiment-specific protocols are provided in Appendices C–F.

4.2 Definition-Derived Responsibility Tests

We next test whether the learned structure representation and execution satisfy the responsibilities defined by the primitive formulation. To avoid confounding representation quality with temporal-support mismatch, we first quantify how well a globally calibrated fixed-width partition aligns with primitive instances. It achieves an alignment rate of 0.2475 at temporal IoU ≥ 0.75 on 6,000 primitive supports disjoint from the calibration set. We therefore evaluate the comparison models with oracle primitive support, isolating representation quality from temporal-support mismatch once the primitive support is correct.

Primitive identity should remain invariant across execution changes. We perform cross-execution identity retrieval: each query must retrieve the same primitive structure under a different execution from candidates with the same K, so structural length alone cannot identify the correct match. Across 6,000 queries, our structure representation achieves R@1 of 1.000 overall and at every structural length (Fig. 3A). Sem-AE obtains overall R@1 of 0.473 and 0.314 across the two seeds, while Sem-Factor obtains 0.305 and 0.227. Reconstruction with primitive-level semantic shaping, even with the factorized extension, does not consistently recover the same primitive across executions. By contrast, our formulation defines identity through execution-invariant structure, and the learned structure representation preserves that identity across executions.

The structure representation z_s should preserve the complete primitive structure: its structural length, constituent canonical basic-motion identities, and their order. We evaluate this responsibility on primitive structures held out from primitive-structure training. For our model, a weak readout from z_s recovers the complete ordered structure exactly for all 468 instances under seen executions and all 120 instances under held-out executions. Exact recovery of the full structure, however, does not by itself isolate whether z_s preserves constituent order. We therefore use a matched order-only test that holds structural length, execution e, and the canonical basic-motion multiset fixed while changing only constituent order. Our model recovers the correct ordered structure for all 234 matched pairs (Fig. 3B). Under the same weak-readout interface, neither Sem-AE nor Sem-Factor yields an exact recovery on the held-out structures or the matched order-only pairs in either seed. Together, these results establish that reconstruction with primitive-level semantic shaping, including the factorized extension, does not yield exact recovery of the complete ordered structure in these tests; by contrast, our learned structure representation preserves the complete primitive structure. An illustrative z_s similarity analysis under selected structural changes is provided in Appendix C.2.

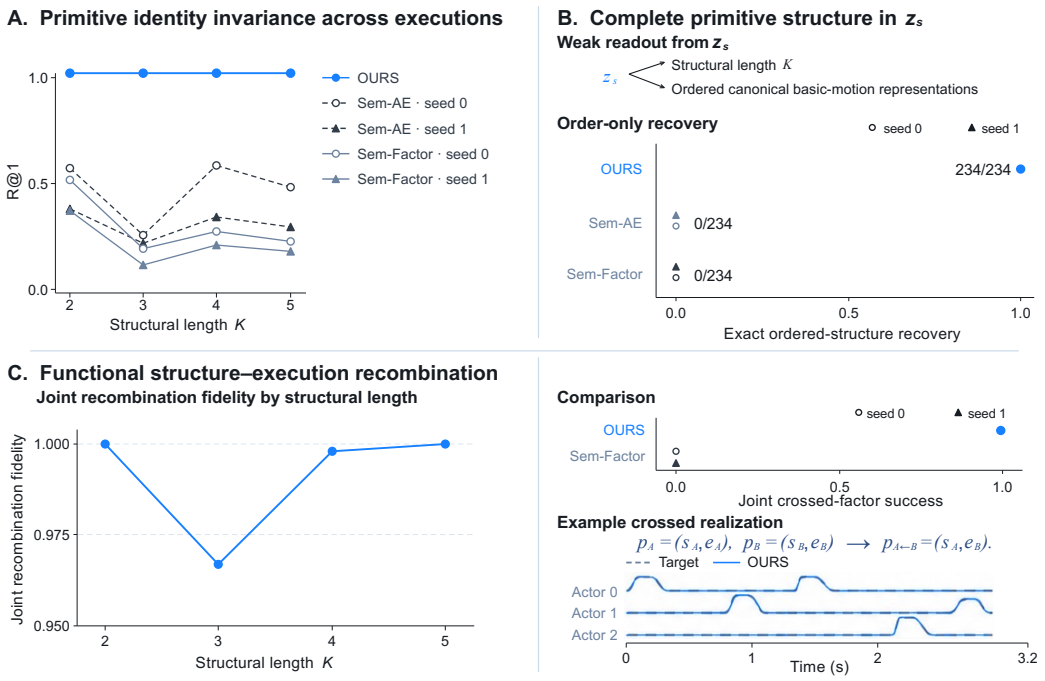

Figure 3: **Definition-derived responsibility tests for primitive structure and execution. (A)** Same-K retrieval across executions tests primitive-identity invariance; comparison models are evaluated with oracle primitive support. **(B)** A weak readout tests whether z_s preserves structural length and the ordered sequence of canonical basic-motion representations, while a matched order-only comparison isolates constituent order. **(C)** Given $p_A = (s_A, e_A)$ and $p_B = (s_B, e_B)$, recombination targets $p_{A \leftarrow B} = (s_A, e_B)$; our model achieves 2030/2048 joint successes (99.1%), with an example target-output overlay.

Finally, we test whether structure and execution remain functionally usable under recombination. Recombination provides a stronger test than identity reuse alone: structure must determine the primitive's motion structure, while execution must determine how that structure is realized. Given $p_A = (s_A, e_A)$ and $p_B = (s_B, e_B)$, we form $p_{A \leftarrow B} = (s_A, e_B)$. Across 2,048 crossed directions, our model preserves s_A in all cases and satisfies the execution criteria for e_B in 2,030, yielding 2,030/2,048 joint successes (99.1%; Fig. 3C). Sem-Factor provides the corresponding semantic comparison because it uses separate structure and execution latents and directly supervises the execution representation. Neither evaluated seed yields a joint success under the same crossed target. The tested semantic factorized model therefore does not satisfy these responsibilities under this crossing interface, whereas our learned structure representation and explicit execution remain functionally usable in recombination. Fig. 3C also shows one target–output realization.

Together, these tests establish the targeted responsibilities in this controlled setting: primitive identity remains reusable across execution changes, z_s preserves the complete primitive structure, and structure and execution remain functionally usable under recombination. Population sizes and headline results are summarized in Table 9 at the end of Appendix C.

4.3 From Primitive Instances to Longer Trajectories

Having established the definition-derived responsibilities under controlled interventions, we next evaluate integrated inference from complete primitive recordings. From each recording, the model infers the structure representation z_s together with execution $(\hat{\eta}_{1:K}, \hat{\rho}, \hat{L})$. Across 468 recordings with seen executions and 120 with held-out executions, primitive structure is recovered exactly and the execution estimates satisfy the evaluation criteria in every case. Thus, structure and execution can be inferred jointly from complete primitive recordings. We then move to longer continuous trajectories containing multiple primitive instances, with primitive boundaries no longer provided

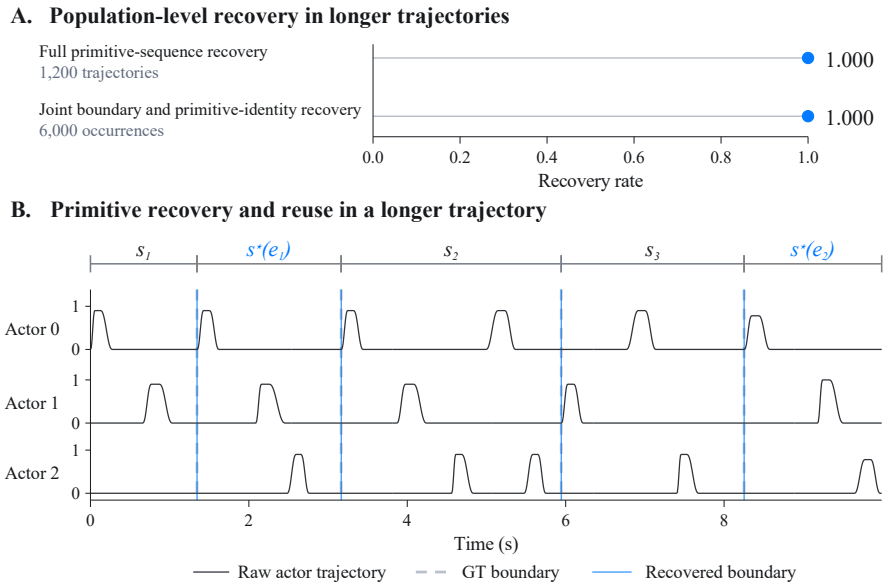

Figure 4: **Primitive recovery and reuse in longer trajectories. (A)** Closed-bank recovery without given primitive boundaries, reporting full primitive-sequence recovery and joint boundary–identity recovery. **(B)** An example trajectory shows two occurrences of the same primitive identity under different executions and neighboring contexts; ground-truth and recovered boundaries are overlaid.

at inference. Recovery is performed against a fixed bank of 168 primitive structures. The complete primitive sequence is recovered exactly on all 1,200 evaluation trajectories, and all 6,000 primitive occurrences are recovered with the correct boundary and primitive identity (Fig. 4A).

The same evaluation tests whether primitive identity remains reusable when embedded in a longer trajectory. Repeated primitive structures occur under distinct executions and neighboring contexts. Fig. 4B shows one such trajectory: the same primitive identity $s^\star$ appears as $s^\star(e_1)$ and $s^\star(e_2)$ in different surrounding contexts, and both occurrences are recovered with boundaries coincident with ground truth. These results extend the cross-execution identity reuse established in Sec. 4.2 from individual primitive instances to longer trajectories in which primitive boundaries are not provided. Appendices D and E detail integrated inference and closed-bank recovery, respectively; Appendix F details additional stress-test and evaluation-extension protocols and results.

5 Conclusion

For motion primitives to be reusable, instances that differ in execution must still be recognizable as the same primitive. We make this criterion explicit by defining primitive identity as the ordered structure of canonical basic motions, while execution captures their local realizations, relative rhythm, and overall temporal scale; a hierarchical model then learns the identity-defining structure while estimating execution separately. In controlled experiments, the learned structure representation preserves primitive identity across execution changes, preserves the complete primitive structure, and supports structure–execution recombination. The model also jointly infers structure and execution from complete primitive recordings and, against a fixed primitive bank, recovers primitive identities in longer trajectories without given boundaries, including repeated occurrences under different executions and neighboring contexts. Comparisons show that even with oracle primitive support, reconstruction-based models with primitive-level semantic shaping—including a factorized extension with supervised execution—do not satisfy the required representation responsibilities: cross-execution reuse, complete-structure preservation, and recombination. Together, these results show that reusable primitive identity can be learned and evaluated directly when identity-defining structure is made explicit and separated from execution. These findings are established in a controlled synthetic setting, while broader generalization and higher-level skill organization remain outside the present scope.

AI USE STATEMENT

We used generative AI tools to assist with the implementation of experimental code, manuscript drafting and editing, and experimental workflow management, including organizing experiment records and tracking experimental runs and results. AI-assisted code was reviewed, tested, and validated by the authors before being used in the reported experiments, and AI-assisted manuscript content and experimental records were reviewed for correctness and consistency. The authors take responsibility for the final content of this work, including its text, claims, reported results, code, and other artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Appendix B.1 documents benchmark construction, execution variation, train-only augmentation, and benchmark semantic checks. Appendix B.2 specifies the model architecture, normalization, stagewise training, checkpoint selection, and analytic execution and timing, while Appendix A provides the training, primitive-recording inference, and closed-bank recovery algorithms; Appendix B.3 gives the comparison-model and calibration procedures. Experiment-specific evaluation protocols and population results are reported in Appendices C–F for the responsibility tests, integrated primitive-recording inference, closed-bank trajectory recovery, and robustness extensions. Together, these sections provide the benchmark, model/training, and evaluation details needed to reproduce the reported experiments.

REFERENCES

Emilio Bizzi, Andrea D'Avella, Philippe Saltiel, and Matthew Tresch. Modular organization of spinal motor systems. *The Neuroscientist*, 8(5):437–442, October 2002. doi: 10.1177/107385802236969. URL https://doi.org/10.1177/107385802236969.

Jake Bruce, Michael D. Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, Yusuf Aytar, Sarah Maria Elisabeth Bechtle, Feryal Behbahani, Stephanie C. Y. Chan, Nicolas Heess, Lucy Gonzalez, Simon Osindero, Sherjil Ozair, Scott Reed, Jingwei Zhang, Konrad Zolna, Jeff Clune, Nando De Freitas, Satinder Singh, and Tim Rocktäschel. Genie: Generative interactive environments. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 4603–4623. PMLR, 2024. URL https://proceedings.mlr.press/v235/bruce24a.html.

Guangyan Chen, Qi Shao, Te Cui, Zichen Zhou, Weixin Mao, Luojie Yang, Meiling Wang, Yi Yang, Hua Chen, and Yufeng Yue. Learning a unified latent action space from videos with action-centric cycle consistency. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12871–12880, June 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Chen_Learning_a_Unified_Latent_Action_Space_from_Videos_with_Action-centric_CVPR_2026_paper.html.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1724–1734. Association for Computational Linguistics, 2014. doi: 10.3115/v1/D14-1179.

Jingwen Deng, Zihao Wang, Shaofei Cai, Anji Liu, and Yitao Liang. Open-world skill discovery from unsegmented demonstration videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10708–10718, October 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Deng_Open-World_Skill_Discovery_from_Unsegmented_Demonstration_Videos_ICCV_2025_paper.html.

Jörn Diedrichsen and Katja Kornysheva. Motor skill learning between selection and execution. *Trends in Cognitive Sciences*, 19(4):227–233, April 2015. doi: 10.1016/j.tics.2015.02.003. URL https://doi.org/10.1016/j.tics.2015.02.003.

Simon F. Giszter. Motor primitives—new data and future questions. *Current Opinion in Neurobiology*, 33:156–165, August 2015. doi: 10.1016/j.conb.2015.04.004. URL https://doi.org/10.1016/j.conb.2015.04.004.

Herbert Heuer. Invariant relative timing in motor-program theory. In *The Development of Timing Control and Temporal Organization in Coordinated Action*, volume 81 of *Advances in Psychology*, pp. 37–68. North-Holland, 1991. doi: 10.1016/S0166-4115(08)60759-1. URL https://www.sciencedirect.com/science/article/pii/S0166411508607591.

Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, 25(2): 328–373, February 2013. doi: 10.1162/NECO_a_00393. URL https://doi.org/10.1162/NECO_a_00393.

Byeongho Lee, Yonghyeon Lee, Seungyeon Kim, MinJun Son, and Frank C. Park. Equivariant motion manifold primitives. In *Proceedings of the 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 1199–1221. PMLR, 2023. URL https://proceedings.mlr.press/v229/lee23a.html.

Seungjae Lee, Yibin Wang, Haritheja Etukuru, H. Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 26991–27008. PMLR, 2024. URL https://proceedings.mlr.press/v235/lee24y.html.

Hao Luo, Ye Wang, Wanpeng Zhang, Haoqi Yuan, Yicheng Feng, Haiweng Xu, Sipeng Zheng, and Zongqing Lu. Joint-aligned latent action: Towards scalable VLA pretraining in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 35047–35058, June 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Luo_Joint-Aligned_Latent_Action_Towards_Scalable_VLA_Pretraining_in_the_Wild_CVPR_2026_paper.html.

Alexandros Paraschos, Christian Daniel, Jan R. Peters, and Gerhard Neumann. Probabilistic movement primitives. In *Advances in Neural Information Processing Systems*, volume 26, 2013. URL https://proceedings.neurips.cc/paper/2013/hash/e53a0a2978c28872a4505bdb51db06dc-Abstract.html.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. FAST: Efficient action tokenization for vision-language-action models. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.012. URL https://roboticsproceedings.org/rss21/p012.html.

Seungeun Rho, Aaron Trinh, Danfei Xu, and Sehoon Ha. Reference grounded skill discovery. In *International Conference on Learning Representations*, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/hash/98d0ad88db1e51bd0aa341a823290ece-Abstract-Conference.html.

Mike Schuster and Kuldip K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681, 1997. doi: 10.1109/78.650093.

Charles H. Shea and Gabriele Wulf. Schema theory: A critical appraisal and reevaluation. *Journal of Motor Behavior*, 37(2):85–102, 2005. doi: 10.3200/JMBR.37.2.85-102. URL https://www.tandfonline.com/doi/abs/10.3200/JMBR.37.2.85-102.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pp. 5998–6008, 2017.

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*, pp. 11089–11099,

October 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Wang_VQ-VLA_Improving_Vision-Language-Action_Models_via_Scaling_Vector-Quantized_Action_Tokenizers_ICCV_2025_paper.html.

Seonghyeon Ye, Joel Jang, Byeongguk Jeon, Se June Joo, Jianwei Yang, Baolin Peng, Ajay Mandlekar, Reuben Tan, Yu-Wei Chao, Bill Yuchen Lin, Lars Liden, Kimin Lee, Jianfeng Gao, Luke Zettlemoyer, Dieter Fox, and Minjoon Seo. Latent action pretraining from videos. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=VYOe2eBQeh.

Chuheng Zhang, Tim Pearce, Pushi Zhang, Kaixin Wang, Xiaoyu Chen, Wei Shen, Li Zhao, and Jiang Bian. What do latent action models actually learn? In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-4906. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/d7b3cef7c31b94a4a533db83d01a8882-Abstract-Conference.html.

Jiajie Zhang, Sören Schwertfeger, and Alexander Kleiner. From observation to action: Latent action-based primitive segmentation for VLA pre-training in industrial settings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6750–6759, June 2026. URL https://openaccess.thecvf.com/content/CVPR2026/html/Zhang_From_Observation_to_Action_Latent_Action-based_Primitive_Segmentation_for_VLA_CVPR_2026_paper.html.

Tony Z. Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.016. URL https://roboticsproceedings.org/rss19/p016.html.

Ruijie Zheng, Ching-An Cheng, Hal Daumé III, Furong Huang, and Andrey Kolobov. PRISE: LLM-style sequence compression for learning temporal action abstractions in control. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 61267–61286. PMLR, 2024. URL https://proceedings.mlr.press/v235/zheng24b.html.

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*, volume 38, pp. 191629–191654, 2025. doi: 10.52202/085713-5755.

Huayi Zhou and Kui Jia. VLBiMan: Vision-language anchored one-shot demonstration enables generalizable bimanual robotic manipulation. In *International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=he86smZzRk.

A METHOD DETAILS

A.1 ALGORITHMS FOR LEARNING, INFERENCE, AND PRIMITIVE RECOVERY

Algorithm 1 Training and Fitting the Hierarchical Primitive Model

Input: Training primitive recordings with temporal-support annotations, observed actors, and structure groupings.

Output: $D_{\mathrm{sup}}, \Gamma, \mathcal{C}^S, E_f, E_c, E_s, D_s$.

Support detection and canonical geometry

1. Train the shared support detector D_{sup} using temporal-support annotations.
2. Apply D_{sup} to the training recordings to obtain localized motion observations.
3. For each localized observation:
   - estimate its local execution;
   - canonicalize the observation;
   - obtain its coordinate estimate $\hat{c}_i$.
4. Group the training estimates $\hat{c}_i$ into canonical local-motion geometries and retain the assignment rule
$$g_i = \Gamma(\hat{c}_i).$$
5. For each geometry group g, jointly estimate a reusable canonical coordinate c_g^S from the assigned motions, forming
$$\mathcal{C}^S = \{c_g^S\}_g.$$

Basic-motion representation

6. Train E_f to encode actor identity.
7. Train E_c with $\hat{c}_i$ as input and the corresponding assigned canonical coordinate c_i^S as a canonical anchor, encouraging different executions of the same canonical motion to be represented consistently.
8. Form
$$\hat{h}_i = [E_f(f_i); E_c(\hat{c}_i)].$$
9. Freeze E_f and E_c.

Primitive structure representation

10. For each primitive training instance, compute
$$z_s = E_s(\hat{h}_1, \ldots, \hat{h}_K).$$
11. Form the corresponding canonical representation sequence by replacing each $\hat{c}_i$ with c_i^S.
12. Train the weak slot-conditioned decoder D_s to reconstruct this ordered canonical sequence and predict its length.
13. Apply an invariance objective across executions sharing the same structure.
14. Freeze E_s and D_s.

Return:
$$D_{\mathrm{sup}}, \Gamma, \mathcal{C}^S, E_f, E_c, E_s, D_s.$$

Algorithm 2 Inference from a Primitive Recording

Input: One primitive recording; frozen $D_{\mathrm{sup}}, \Gamma, \mathcal{C}^S, E_f, E_c, E_s$.

Output: $[z_s, \hat{\eta}_{1:K}, \hat{\rho}, \hat{L}]$.

1. Apply D_{sup} to localize and temporally order
$$\{[\hat{t}_{s,i}, \hat{t}_{e,i}]\}_{i=1}^{K}.$$
2. For each localized observation x_i:
   - obtain observed actor f_i;
   - estimate local execution $\hat{\eta}_i$;
   - invert the estimated transformation to canonicalize the observation;
   - represent the canonicalized observation by $\hat{c}_i$;
   - assign
   $$g_i = \Gamma(\hat{c}_i), \qquad c_i^S = \mathcal{C}^S[g_i];$$
   - form
   $$u_i = (f_i, c_i^S);$$
   - encode
   $$h_i = [E_f(f_i); E_c(c_i^S)];$$
   - determine realized semantic-anchor time $\hat{\tau}_i$ from its support, estimated local execution, and assigned canonical geometry.
3. Compute
$$z_s = E_s(h_1, \ldots, h_K).$$
4. Compute
$$\hat{L} = \hat{\tau}_K - \hat{\tau}_1.$$
5. Compute
$$\hat{\Delta}_i = \frac{\hat{\tau}_{i+1} - \hat{\tau}_i}{\hat{L}}, \qquad \hat{\rho} = (\hat{\Delta}_1, \ldots, \hat{\Delta}_{K-1}).$$
6. Return
$$[z_s, \hat{\eta}_{1:K}, \hat{\rho}, \hat{L}].$$

Algorithm 3 Closed-Bank Primitive Recovery from a Longer Trajectory

Input: A longer trajectory without primitive boundaries; known primitive bank

$$\mathcal{B} = \{(s_b, z_b^{\text{bank}}, K_b)\}_{b=1}^{B};$$

frozen length-specific acceptance thresholds $\{\gamma_K\}$; frozen components from Algorithms 1–2.

Output: Ordered sequence of recovered primitive occurrences.

1. Apply the frozen basic-motion inference frontend to obtain
$$\{([\hat{t}_{s,i}, \hat{t}_{e,i}], u_i, h_i, \hat{\eta}_i, \hat{\tau}_i)\}_{i=1}^{M}.$$
2. For each candidate start j and each structural length K represented in the primitive bank: compute
$$z_{j,K} = E_s(h_j, \ldots, h_{j+K-1});$$
find
$$b^*(j, K) = \arg \max_{b:K_b=K} \cos(z_{j,K}, z_b^{\text{bank}});$$
accept the candidate when
$$\cos\big(z_{j,K}, z_{b^*(j,K)}^{\text{bank}}\big) \geq \gamma_K.$$
3. Compute suffix feasibility:
$$F(M+1) = \text{true},$$
and for $j = M, \ldots, 1$,
$$F(j) = \bigvee_K \left[\text{Valid}(j, K) \wedge F(j+K)\right].$$
4. Parse the basic-motion sequence from left to right. At each position j, select the longest accepted candidate whose remaining suffix is feasible.
5. For each selected span:
   - assign the corresponding primitive structure from the known bank;
   - retain the local execution estimates of its basic motions;
   - compute relative rhythm and temporal scale from realized anchor times within the span;
   - assign its temporal support from the first to the last basic-motion support in the span.
6. Advance to the first unparsed basic motion and continue until the sequence is completely parsed.
7. Return the ordered sequence of recovered primitive occurrences.

The thresholds γ_K are independently calibrated and frozen before evaluation. They are operational recognition parameters and do not define primitive identity.

B Experimental Setup and Reproducibility

B.1 Benchmark Construction

The benchmark separates the base semantic bank from the augmented population used to train primitive structure (Table 1). Each canonical basic motion pairs an observed actor with a canonical local-motion geometry. Current raw inference uses 100-Hz position trajectories; actor identity is observed through the channel interface. Basic-motion representation training used saved B-spline coefficients extracted from annotated constituent supports, with saved canonical coefficients as geometry targets. Primitive-structure training used the resulting saved basic-motion vectors and canonical-vector targets computed by the frozen encoders. Current primitive-recording and longer-trajectory inference instead recover the corresponding quantities through the learned-support raw-trajectory frontend; the representation stages were not retrained through that frontend.

Table 1: Benchmark scale. Structure splits are training/validation/test; per-K counts follow $K = 2, 3, 4, 5$.

Item	Value
Structural length	$K = 2$–5
Base structures	168
Per-K structures	24 / 48 / 48 / 48
Structure split	120 / 24 / 24
Observed actors	3
Canonical geometries	3
Canonical basic motions	9
Execution configurations	117
Saved primitive instances	5,400
Augmented training structures	400
Representation-level training instances	10,008

Execution variation. Executions vary in local amplitude, local duration, relative rhythm, and global temporal scale. The neutral profile uses $a_i = 0.9$ and $\lambda_i = 0.33$ s; amplitude-only and duration-only profiles alternate $0.78/1.0$ and $0.28/0.38$ s, respectively. Rhythm archetypes are even, accelerating, decelerating, and alternating for $K \geq 3$, with a single interval for $K = 2$. Global scale factors $g \in \{0.9, 1.0, 1.1\}$ give $L = g(K-1)\,0.65$ s. Exact configurations and split assignments are retained in the saved manifests.

Train-only order augmentation. Augmentation permutes training representations without rendering new raw trajectories. Identity permutations and collisions with original training, validation, or test structures are excluded. The 120 original and 280 globally distinct augmented structures yield 400 training structures; 2,664 original and 7,344 augmented instances yield 10,008 training instances. The base semantic bank remains 168 structures. The held-out-structure evaluation in Appendix C.2 uses 24 structures with zero overlap with the 400 structures used for primitive-structure training after train-only augmentation.

Benchmark semantic closure. The deterministic benchmark renderer closes the specified structure and execution factors at saved precision. Across 2,304 controlled primitives, event order is exact, anchor, rhythm, and span errors are zero, and trajectory normalized MSE is zero; across 11,520 semantic counterfactuals, recombined structure and timing remain exact and valid. This verifies benchmark semantics rather than learned generative compositionality.

B.2 Model Architecture and Training

Tables 2 and 3 summarize the model and stagewise training. The basic-motion encoders are frozen before primitive-structure training.

The structure encoder uses learned positional embeddings and a learned `STRUCT` token; its encoded token is $z_s \in \mathbb{R}^{64}$. D_s reconstructs basic-motion representations with no teacher forcing or slot-to-slot communication; it is not a trajectory generator.

Table 2: Hierarchical model architecture.

Component	Implementation
Support detector	Shared 2-layer bidirectional GRU, 32 hidden units/direction, dropout 0.1; LayerNorm and activity/start/end/offset outputs
Canonical geometry	16-D cubic B-spline; train-only $k = 3$ clustering and pooling
Actor encoder E_f	$\mathrm{Embedding}(3, 32)$ + LayerNorm
Geometry encoder E_c	$16 \to 32 \to 32$ MLP, GELU and LayerNorm after each layer
Basic-motion representation	64-D concatenation of actor and geometry encodings
Structure encoder E_s	2-layer Transformer, 4 heads, $d_{\mathrm{model}} = 64$, FFN width 128, GELU, zero dropout
Structural readout D_s	Five 8-D slot queries; shared $72 \to 64 \to 64$ MLP with GELU; separate $64 \to 4$ structural-length head
Local execution	Analytic amplitude and duration
Relative rhythm and scale	Analytic ρ, L from realized semantic-anchor times

Canonical geometry and normalization. Geometry is fitted with a cubic clamped uniform B-spline ($M = 16$, degree 3) in normalized local phase. Least squares fits peak-normalized instance geometry; train-only KMeans with $k = 3$ groups the coefficient estimates, and canonical pooling gives each execution equal aggregate weight. Coefficient normalization is fitted on the basic-motion representation-training population and stored with the frozen basic-motion encoders. The selected primitive-structure checkpoint is paired with the normalization statistics saved from its augmented training population, and the same statistics are used for inverse normalization of structural readouts.

Table 3: Stagewise training. All stages use AdamW with learning rate 10^{-3} and weight decay 10^{-4}. Epoch counts are maxima.

Stage	Objective	Batch / epochs	Frozen route
Support detection	Support supervision	64 / 50	Clean seed 0 support detector
Basic-motion representation	$\mathcal{L}_f + \mathcal{L}_{\mathrm{geo}} + 0.1\mathcal{L}^c_{\mathrm{inv}}$	1024 / 80	Frozen basic-motion encoders
Primitive-structure representation	$\mathcal{L}_h + \mathcal{L}_K + 0.1\mathcal{L}^z_{\mathrm{inv}}$	256 / 120	Frozen structure encoder / readout

Support detection used seeds 0/1/2, with seed 0 predesignated as primary. Within each seed, checkpoints maximize validation event F1 at temporal IoU 0.75, breaking ties by the sum of start- and end-boundary MAE. Basic-motion representation training used only seed 0, as specified in its training protocol. Primitive-structure training used seeds 0/1; checkpoints for both representation stages minimize their respective validation total objectives within each seed. The reported inference configuration fixes the primitive-structure checkpoint to seed 0; seed 1 was also evaluated in the matched structure-learning and downstream compatibility tests. Exact training configurations are retained with the frozen artifacts.

Analytic execution and timing. For detected support $[\hat{t}_{s,i}, \hat{t}_{e,i}]$ and canonical semantic-anchor phase $\alpha_i = \mathcal{A}_{\mathrm{sem}}(c^S_i)$,

$$\hat{a}_i = \max_{t \in [\hat{t}_{s,i}, \hat{t}_{e,i}]} q_i(t), \qquad \hat{\lambda}_i = \hat{t}_{e,i} - \hat{t}_{s,i}, \qquad \hat{\tau}_i = \hat{t}_{s,i} + \alpha_i \hat{\lambda}_i,$$

$$\hat{L} = \hat{\tau}_K - \hat{\tau}_1, \qquad \hat{\rho}_i = \frac{\hat{\tau}_{i+1} - \hat{\tau}_i}{\hat{L}}.$$

No learned local-execution or neural timing head is used. Temporal scale is the span between the first and last realized semantic anchors.

B.3 Comparison Models and Calibration

Fixed-width temporal partition. Calibration uses an independent population of 300 trajectories and 1,500 primitive supports. Candidate widths are derived from support-duration percentiles and the

mean, with offsets $0, 0.25W, 0.50W, 0.75W$ searched for each width. Selection prioritizes alignment at IoU 0.75. The frozen result, $W^* = 2.21$ s and offset zero, defines one global partition applied across trajectories; exact search settings remain in the saved configuration.

Reconstruction controls. Table 4 lists the three reconstruction-only latent designs alongside the semantic comparisons. All use the same recurrent encoder/decoder family: a 2-layer bidirectional GRU encoder with 64 hidden units/direction and a 1-layer GRU trajectory decoder with hidden width 64. Inputs are the three active position channels at 100 Hz, standardized using training statistics. Full latent width is 64.

Table 4: Comparison-model designs. $\mathcal{L}_{\text{rec}}$ is position MSE plus 0.1 times normalized-length MSE; $\mathcal{L}_{\text{commit}}$ is the VQ commitment loss.

Model	Latent / interface	Objective / supervision
Continuous AE	64-D continuous	$\mathcal{L}_{\text{rec}}$
VQ AE	64-D VQ, 256 codes	$\mathcal{L}_{\text{rec}} + 0.25\mathcal{L}_{\text{commit}}$
Hybrid	32-D VQ + 32-D continuous	$\mathcal{L}_{\text{rec}} + 0.25\mathcal{L}_{\text{commit}}$
Sem-AE	64-D continuous z_s	$\mathcal{L}_{\text{rec}} + 0.1\mathcal{L}_{\text{sem}}$
Sem-Factor	64-D z_s + 64-D z_e	$\mathcal{L}_{\text{rec}} + 0.1\mathcal{L}_{\text{sem}} + \mathcal{L}_{\text{exec}}$

Both VQ models use 256 codes and EMA decay 0.99; the hybrid continuous component uses dropout 0.1. Fixed-partition and oracle-support controls use AdamW (learning rate 10^{-3}, weight decay 10^{-5}), batch 256, gradient clipping at 1, and seeds 0/1 for at most 100 epochs with patience 12. Checkpoint selection minimizes total validation loss on the independent raw calibration population. Each control has 151,620 trainable parameters, excluding EMA buffers; the hierarchy is not parameter-count matched. All reported vector-quantized controls remain non-collapsed (Appendix B.3).

Semantic comparisons. Sem-AE and Sem-Factor receive oracle primitive support and use the same raw primitive-recording recurrent autoencoder family as the continuous reconstruction control. Sem-AE learns a 64-dimensional structure latent z_s and combines trajectory reconstruction with primitive-level semantic shaping. Sem-Factor adds a parallel 64-dimensional execution projection from the pooled recurrent states and reconstructs from the concatenated (z_s, z_e). A 64–64–15 MLP with GELU predicts the execution factors from z_e. The two models have 151,620 and 189,971 trainable parameters, respectively.

Each anchor is paired with one same-structure, different-execution positive when available and one different-structure negative. Negative sampling cycles through structural-length, same-multiset order, and canonical basic-motion identity pools, sampling a structure and then a recording uniformly within the selected pool, with a fallback to all different structures. These structural-change categories are overlapping sampling and analysis metadata only; they are not supplied as structural-change labels to the model. Both models use $\mathcal{L}_{\text{sem}} = \mathcal{L}_{\text{pos}} + \mathcal{L}_{\text{neg}}$, where the positive term averages $1 - \cos(z_s^i, z_s^j)$ and the negative term averages $\max(0, \cos(z_s^i, z_s^j))$, with zero margin. Neither encoder is trained to predict the ordered constituent structure.

Sem-Factor directly supervises execution through targets $(a_{1:5}, \lambda_{1:5}, \rho_{1:4}, L)$, normalized using training-set statistics. Its execution loss is SmoothL1 with $\beta = 1$, averaged over valid target components; the validity mask is used only in this loss. Both models reconstruct matched instances during training. Crossed structure–execution donor/target combinations are evaluation-only and are not training targets.

Training uses the original 6,000 oracle-support training recordings and 1,500 independent calibration recordings, with seeds 0 and 1, AdamW (learning rate 10^{-3}, weight decay 10^{-5}), effective batch 256, gradient clipping at 1, at most 100 epochs, and patience 12. Checkpoints minimize the total validation objective in Table 4; validation pair sampling resets its random seed to 2027. Responsibility-test scores are not used for checkpoint selection or hyperparameter tuning.

Frozen-latent structural readout. After encoder training, a separate weak readout is trained on detached z_s from the original training population. An 8-dimensional learned slot query is concatenated with z_s and passed through a 72–64–64 MLP with GELU; a separate 64-to-4 head predicts structural

length. Targets are the frozen canonical basic-motion representations, normalized using training data. AdamW uses learning rate 10^{-3}, weight decay 10^{-4}, batch 256, at most 100 epochs, and patience 12. Selection minimizes calibration SmoothL1 representation loss plus structural-length cross-entropy, not held-out-structure, order-only, or recombination scores. The calibration structures include some structures used in the matched order-only evaluation; this is not a development-blind evaluation.

Responsibility coverage and comparison boundary. Table 5 summarizes the support-specific evaluation protocols. N/A denotes an unevaluated or inapplicable test, not zero performance or failure. A dash in the held-out-structure column denotes coverage not specified for the original reconstruction-only controls.

Table 5: Responsibility-test coverage. Hybrid denotes VQ/continuous latents.

Support	Representation	Reuse	Held-out structure	Order	Recombination
Fixed partition	Continuous AE	Available	—	N/A	N/A
Fixed partition	VQ AE	Available	—	N/A	N/A
Fixed partition	Hybrid	Available	—	N/A	N/A
Oracle primitive	Continuous AE	Available	—	N/A	N/A
Oracle primitive	VQ AE	Available	—	Available	N/A
Oracle primitive	Hybrid	Available	—	Available	Available
Oracle primitive	Sem-AE	Available	Available	Available	N/A
Oracle primitive	Sem-Factor	Available	Available	Available	Available

The controls share the corresponding evaluation responsibility and, where applicable, the same evaluation populations or donor–target manifests. Neither the reconstruction-only controls nor the semantic comparisons are matched to the proposed hierarchy in architecture, parameter count, supervision, or priors. They test whether the evaluated reconstruction and semantic-shaping objectives yield the required responsibilities under the relevant evaluation interface.

VQ codebook diagnostics. Table 6 reports the saved final-training codebook health. All vector-quantized conditions are non-collapsed.

Table 6: Saved codebook health for the reconstruction-trained controls.

		Seed 0		Seed 1	
Support	Model	Active codes	Perplexity	Active codes	Perplexity
Fixed partition	VQ AE	246	190.57	227	156.63
Fixed partition	Hybrid	253	196.29	250	187.72
Oracle primitive	VQ AE	252	207.94	238	175.15
Oracle primitive	Hybrid	221	159.61	249	197.58

C Definition-Derived Responsibility Experiments

C.1 Cross-Execution Identity Reuse

Cross-execution retrieval uses 6,000 primitive-occurrence queries. Candidates are restricted to the same structural length K and a different source trajectory; positives share primitive structure but use a different execution. Table 7 reports the exact overall retrieval results used in Sec. 4.2.

Table 7: Overall cross-execution identity retrieval.

Model	Seed	R@1	mAP
OURS	—	1.000	1.000
OB-AE	0	0.539	0.365
OB-AE	1	0.383	0.377
OB-VQ	0	0.326	0.243
OB-VQ	1	0.253	0.252
OB-VQ+continuous	0	0.524	0.369
OB-VQ+continuous	1	0.487	0.297
Sem-AE	0	0.473	0.402
Sem-AE	1	0.314	0.382
Sem-Factor	0	0.305	0.413
Sem-Factor	1	0.227	0.389

C.2 Complete Primitive Structure

Held-out structure generalization. We additionally evaluate the structural readout on 24 primitive structures held out from primitive-structure training, with zero overlap with the 400 structures used for training after train-only augmentation. These structures are unseen ordered compositions of the known canonical basic-motion vocabulary. The complete ordered structure is recovered exactly for all 468 evaluated instances under seen executions and all 120 instances under held-out executions. This evaluation tests generalization to unseen primitive compositions of known canonical basic motions; it does not test unseen canonical basic-motion identities or open-vocabulary primitive discovery.

Under the frozen-latent weak readout used in the evaluation, Sem-AE and Sem-Factor each obtain 0/468 exact ordered-structure recoveries under seen executions and 0/120 under held-out executions, in each of seeds 0 and 1. Both models also obtain 0/234 exact recoveries in the matched order-only test in each seed. These exact-recovery results are conditional on this weak-readout interface and do not establish that the semantic latents contain no structural information.

Figure 5 provides secondary illustrative representation analysis using four selected structures with two executions per structure.

C.3 Crossed Structure–Execution Recombination

Joint recombination success requires exact target primitive identity, ordered canonical basic-motion tokens, and event count, together with all specified execution criteria: amplitude MAE ≤ 0.02, local-duration MAE $\leq 0.01\,\mathrm{s}$, relative-rhythm L_1 error ≤ 0.02, and temporal-scale error $\leq 0.02\,\mathrm{s}$. Table 8 reports the population results by structural length.

Table 8: Crossed structure–execution recombination for OURS.

K	Directions	Structure success	Joint success
2	512	512	512 (1.000)
3	512	512	495 (0.967)
4	512	512	511 (0.998)
5	512	512	512 (1.000)
Overall	**2,048**	**2,048**	**2,030 (0.991)**

For OB-VQ+continuous, both evaluated seeds obtain 0/2,048 structure successes and 0/2,048 joint successes under the same crossed structure–execution protocol.

Sem-Factor obtains 0/2,048 structure successes and 0/2,048 joint successes in each of seeds 0 and 1. It uses separate structure and execution latents; crossed donor/target combinations are evaluation-only and were not used as training targets. Sem-Factor is evaluated through its learned $D(z_s^A, z_e^B)$ decoder/interface, whereas OURS uses its explicit inferred structure/execution route. This is a functional recombination test, not an architecture-matched generative comparison.

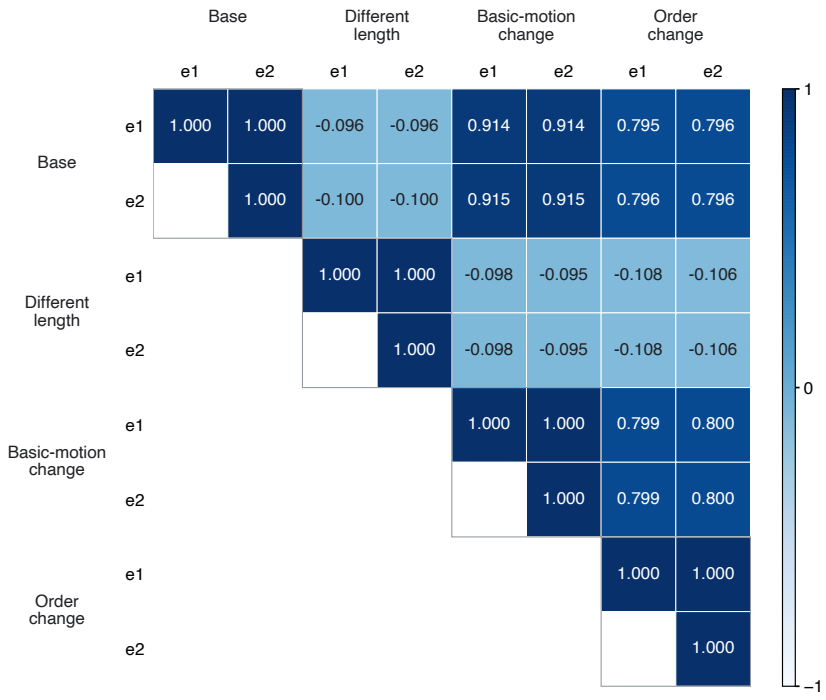

Figure 5: **Illustrative z_s similarity under selected structural changes.** The frozen four-structure, two-execution family retains its original ordering; e1 and e2 identify two executions within each structure. Numerical cosine values are shown to three decimals. Different length is a structural contrast, not an intervention changing only K; complete execution tuples are not matched across different structural lengths. Cosines illustrate representation organization, not a quantitative structural-distance scale.

Table 9: Population results of the responsibility tests.

Responsibility test	Evaluation population	OURS	Semantic comparisons
Cross-execution identity retrieval	6,000 queries	R@1 = 1.000; mAP = 1.000	Sem-AE R@1: 0.473 / 0.314; mAP: 0.402 / 0.382. Sem-Factor R@1: 0.305 / 0.227; mAP: 0.413 / 0.389.
Structural readout	588 primitives from 24 structures held out from primitive-structure training (468 seen-execution; 120 held-out-execution instances)	Exact ordered-structure recovery in both conditions	Sem-AE and Sem-Factor: 0/468 and 0/120 in both seeds.
Matched order-only recovery	234 matched pairs	234/234 exact ordered-structure recovery	Sem-AE and Sem-Factor: 0/234 in both seeds.
Structure–execution recombination	2,048 directions	2030/2048 joint successes (99.1%)	Sem-Factor: 0/2048 structure and 0/2048 joint in both seeds.

Structural readout evaluates recovery of structural length and the ordered canonical basic-motion representations for 24 structures held out from primitive-structure training, under both seen and held-out executions; the matched order-only test separately isolates constituent order. Semantic structural results are conditional on the frozen-latent weak readout used in the evaluation and do not establish absence of structural information. Slash-separated retrieval values list seeds 0 / 1.

Table 9 summarizes the evaluation populations and headline results across the responsibility tests reported in Sec. 4.2.

D Integrated Primitive-Recording Inference

We evaluate integrated inference from complete primitive recordings with seen and held-out executions. The two conditions contain 468 and 120 recordings, respectively. Primitive-structure recovery reports exact structural readout. Execution evaluation reports the saved joint primitive diagnostic, which tests structure together with local amplitude and duration, relative rhythm, and overall temporal scale against the specified evaluation criteria.

Table 10: Integrated primitive-recording inference. Execution evaluation is reported through the joint primitive diagnostic used in the evaluation.

Execution condition	Recordings	Primitive-structure recovery	Execution evaluation (joint diagnostic)
Seen executions	468	1.000	1.000
Held-out executions	120	1.000	1.000

E Closed-Bank Recovery in Longer Trajectories

Primitive boundaries are not provided at inference, and recovery uses a fixed bank of 168 primitive structures. Internal boundaries are matched monotonically within a tolerance of $0.020\,\mathrm{s}$. Under this protocol, the complete primitive sequence is recovered on all 1,200 evaluation trajectories, and joint boundary–primitive-identity recovery is 1.0 over 6,000 primitive occurrences.

F Robustness and Evaluation Extensions

F.1 Observation Noise

We evaluate additive zero-mean Gaussian observation noise using a separately trained noise-aware support detector and the unchanged clean downstream route. Noise severity specifies the standard deviation as a percentage of the fixed reference amplitude. The adapted route preserves recovery at the tested 0.5%, 1%, and 2% levels; the held-out 5% severity is an extrapolation condition and degrades performance (Table 11). The adapted detector also shows a small degradation in the 20% lower temporal-range control (joint F1 $= 0.987$; trajectory exact-match $= 0.970$).

Table 11: Observation-noise evaluation with the separately trained support detector. Paired mild-noise seeds have identical separately saved results.

Noise severity	Noise seed(s)	Joint boundary + identity F1	Trajectory sequence exact-match rate
0.5%	5101, 5102	1.000	1.000
1%	5101, 5102	1.000	1.000
2%	5101, 5102	1.000	1.000
5% (held out)	5101	0.847	0.703
5% (held out)	5102	0.850	0.695

F.2 Execution-Range Extrapolation

We test the frozen clean model in eight specified endpoint-extrapolation conditions. Amplitude and temporal interventions are evaluated separately, extending below or above the corresponding training range by 10% or 20% of that training-range width. Relative rhythm ρ is unchanged; temporal extensions couple the overall temporal scale L to local durations. Recovery is maintained in all tested conditions (Table 12).

Table 12: Endpoint extrapolation using the frozen clean model.

Extension	Tested endpoint conditions	Number of conditions	Trajectory sequence exact-match	Joint boundary + identity F1
Amplitude	Lower/upper at 10% and 20% of training-range width	4	1.000	1.000
Temporal	Lower/upper at 10% and 20% of training-range width	4	1.000	1.000

F.3 Longer Trajectories

We test the frozen clean model with $N = 8, 10, 12$ primitive occurrences per trajectory, evaluating 300 trajectories at each N. Here N counts primitive occurrences, rather than structural length K or temporal scale L. The model maintains recovery at each tested composition length (Table 13).

Table 13: Longer-trajectory evaluation using the frozen clean model.

Occurrences per trajectory N	Trajectories	Trajectory sequence exact-match rate	Joint boundary + identity F1
8	300	1.000	1.000
10	300	1.000	1.000
12	300	1.000	1.000

G Discussion and Limitations

G.1 Explicit Primitive Identity Aligns Learning and Evaluation

If reusable motion units are to serve as building blocks for complex skills, different realizations of a unit must still be recognized as the same primitive. Reuse therefore requires a criterion for primitive identity: what must remain the same across instances, and what may vary. Making this criterion explicit changes the learning problem. Without an explicit equivalence criterion, primitive identity is left as a semantic property to be examined after representation learning. Our formulation instead directs representation learning toward the ordered canonical basic-motion structure that defines identity, while estimating execution separately.

The definition also determines how the resulting representation should be evaluated. Identity should persist across execution changes, the identity-defining structure should remain recoverable, and structure and execution should remain usable under recombination. The experiments show that a representation satisfying these responsibilities can be learned and that the responsibilities can be tested directly. Defining primitive identity therefore aligns both what is learned and what is evaluated with what primitive reuse requires. The formulation remains distinct from its present implementation: support detection, canonical-geometry estimation, structural encoding, and the weak readout operationalize the identity criterion rather than define it.

G.2 Temporal Alignment, Reconstruction, and Semantic Shaping

The two comparison settings separate two questions: whether a temporal unit aligns with the support of a primitive instance, and whether a representation given the correct primitive support satisfies the responsibilities implied by the identity–execution formulation. The globally calibrated fixed-width partition is poorly aligned with the variable temporal support of primitive instances. Providing oracle primitive support removes this temporal mismatch, yet the reconstruction-only controls still do not satisfy the evaluated responsibilities for cross-execution reuse, complete-structure preservation, and recombination. Adding primitive-level semantic shaping does not close this gap: Sem-AE remains insufficient across the reuse and structure tests, while Sem-Factor additionally uses a separately supervised execution latent but still does not satisfy the recombination responsibility. Together, these results show that temporal alignment, reconstruction, and the tested semantic shaping objectives do

not by themselves yield representations that satisfy the full set of identity–execution responsibilities examined here.

This conclusion is specific to the evaluated comparison families, objectives, and interfaces. It does not imply that fixed-width partitioning, vector quantization, reconstruction, semantic shaping, or factorized latent models are inherently incompatible with reusable primitive semantics, nor does it establish that the proposed hierarchy is necessary. Rather, the experiments test whether these evaluated alternatives are sufficient to satisfy the representation responsibilities derived from the primitive-identity definition.

G.3 From Reusable Primitives to Skill Organization

The broader research program is hierarchical: basic motions can form reusable primitives, and reusable primitives can in turn provide the units from which more complex behavior is organized. The present work addresses this intermediate primitive level by defining reusable primitive identity, learning a representation of its identity-defining structure, and recovering primitive identities as a sequence within longer trajectories. This suggests a hierarchical research program in which representations established at one level become building blocks for studying organization at the next.

Once longer behavior can be represented as a sequence of reusable primitive identities, the next question is how those primitives are organized into higher-level skills. This is a different learning problem from primitive identity itself and is not addressed here. The present evidence for the primitive layer is established in a controlled synthetic setting, and broader generalization remains open. We therefore view skill organization as the next hierarchical problem built on, rather than solved by, the primitive representation studied in this work.