CODEBOOK-GEOMETRY CORRECTION: REPLACING TEST-TIME REFINEMENT WITH A FEW ANALYTIC STEPS IN TRAJECTORY-CONTROLLED TEXT-TO-MOTION GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Text-to-motion generators produce fluent, text-aligned motion, but steering a joint along a prescribed path usually costs either a duplicated control backbone or thousands of per-sample optimization steps at inference. We show that a handful of analytic steps can do the same job, provided the correction is measured in the right geometry. Our central contribution is Codebook-Geometry Correction (CodeGeo): the generator's part codebooks record which changes its frozen decoder was trained to absorb, and so define a metric in which a correction can be measured; CodeGeo moves a generated clip onto the commanded path by the smallest change that metric allows, in at most eight damped steps in place of $1{,}600$ refinement iterations. CodeGeo needs a start that is already close to the targets, which Key/Value Control (KV-Control) supplies: a thin interface (10.5 M trainable parameters) that feeds the trajectory into a frozen masked transformer as extra key/value memory at every attention layer, over Anatomy-Aware Part Tokenization (PartVQ), one codebook per body part unpacked so that every part-frame token is an addressable attention site. CodeGeo has one tuned hyperparameter, the anisotropy of its step metric, and its setting follows the anchor set: root and head anchors keep the codebook's raw anisotropy, limb end-effectors and body-spanning anchor sets relax it, and the effect grows with the number of anchors. On pelvis control the method tracks to 0.24 cm at FID 0.067 in $1/21$ the per-sample time of the iterative schedule it replaces; on multi-joint control it reaches 0.25 cm at FID 0.072, improving on MaskControl's published 0.72 cm and 0.083 on both axes.

1 INTRODUCTION

3D human motion generation drives animation, games, and embodied-AI pipelines, where a text prompt is the fastest way to author a moving character. Recent text-conditioned generators Guo et al. (2024); Pinyoanuntapong et al. (2024b); Zhang et al. (2024a); Tevet et al. (2023); Chen et al. (2023) already turn a prompt into diverse, semantically faithful motion. That motion, however, gives the user no handle on placement: even a high-quality clip cannot be pinned to a specified root path, a foot contact on a prop, or a planner's trajectory—geometric constraints no prompt can express. The problem we address is how to add this precise, extrinsic geometric control on top of such a generator while preserving its generation quality. We study *joint trajectory control*: constraining user-specified joints—a single end-effector keyframe, a root path, or a multi-joint trajectory—to hit given target positions while the prompt continues to dictate gait, style, and intent.

Existing methods. Learned trajectory control faces two recurring costs. ControlNet-style adapters Zhang et al. (2023b); Xie et al. (2024); Pinyoanuntapong et al. (2025); Wang et al. (2024) wire a near-duplicate backbone branch into the trunk, and several also lean on inference-time optimization to hit the targets—MaskControl on a $1{,}600$-iteration refinement, OmniControl on per-step guidance gradients. Test-time optimization methods Wan et al. (2024); Karunratanakul et al. (2024) add no trained branch but perturb latent codes or diffusion noise through a per-sample optimization loop at inference. Both costs raise the same question: must per-layer control be bought with duplicated

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.

parameters, and anchor satisfaction with optimization through the sampler? We answer no to both by making the motion representation do the work.

Approach. Our answer is a single idea: treat the anatomy-aligned part codebook as a *control surface*, and let it supply both the sites control enters through and the geometry the correction is measured in. The codebook's per-part token *addressability* lets control enter as attention memory during sampling: **KV-Control** attaches a control-conditioned key/value pair at every self-attention layer of a frozen masked transformer through a near-identity low-rank projection, over **PartVQ** part codebooks unpacked along the sequence axis (**T-Concat**) so that each (frame, part) pair owns an attention slot. This yields a control-conditioned feed-forward motion in PartVQ's codebook space. The codebook's *second-order statistics* then furnish the metric in which that motion is finished analytically: **Codebook-Geometry Correction** (CodeGeo) moves the clip by the smallest change the metric admits, in ≤ 8 damped steps in place of the inherited $1{,}600$-iteration test-time optimization; how strongly the metric follows codebook geometry is its one hyperparameter, set from the anchor set (§4.4). Figure 1 gives an overview.

Contributions. (1) **Codebook-Geometry Correction** (§3.4), the primary contribution: an analytic solve that replaces the inherited $1{,}600$-iteration test-time refinement with a $3A$-dimensional least-squares problem in ≤ 8 damped steps, under a training-free metric taken from the codebook's own second-order statistics. On pelvis control it reaches $0.24\,\text{cm}$ in $1/21$ the per-sample time of the iterative schedule it replaces, on the same model and hardware; on multi-joint control, $0.25\,\text{cm}$ at FID 0.072. (2) **A rule for CodeGeo's one tuned hyperparameter** (§4.4): the metric's anisotropy follows the anchor set, a scalar trust region serving only rows the metric prices comparably. Root and head anchors keep the raw codebook covariance and body-spanning sets relax it by one to two orders of magnitude, the two moving in opposite directions in FID; the effect vanishes with a single anchor, grows with the anchor count, and limb end-effectors side with the body-spanning sets (also on the validation split). (3) **The interface that produces CodeGeo's start** (§3.2): KV-Control injects trajectory targets as time-indexed key/value memory at every self-attention layer of a frozen masked transformer through a near-identity low-rank branch (10.5 M trainable parameters), over PartVQ's anatomy-aligned part codebooks unpacked so that each frame–part token is attention-addressable. The layout carries the accuracy: the same interface on a channel-stacked trunk of matched quality tracks with 2.2 times the error, and the interface transfers to an RVQ + MoMask backbone (§4.5). Efficiency throughout means trainable parameters and test-time steps; the unpacked layout raises backbone FLOPs $\approx 4.8\times$ over the channel-stacked one (§3.2).

2 Related Work

Text-to-motion generation and representation. Masked motion-token models Guo et al. (2024); Pinyoanuntapong et al. (2024b;a) tokenize motion with a VQ-VAE and predict masked tokens Chang et al. (2022) (autoregressively in Zhang et al. (2023a)); diffusion models Tevet et al. (2023); Zhang et al. (2024a); Chen et al. (2023); Zhang et al. (2023c); Meng et al. (2025a;b) diffuse continuous features, with extensions along orthogonal axes Zhang et al. (2024b); Zeng et al. (2025); Chi et al. (2024); Zou et al. (2024); Jiang et al. (2023). On representation, a global VQ codebook Zhang et al. (2023a) or its residual variant Guo et al. (2024) packs the whole body into one token sequence; TLControl Wan et al. (2024) factors geometry across hand-defined per-part VQs and ACMDM Meng et al. (2025a) argues for absolute coordinates. We follow the part-aware line but *derive* the partition from data.

Controllable text-to-motion. Prior methods differ in *where* control enters the generator. *Sampling-time guidance and test-time optimization*: GMD Karunratanakul et al. (2023) guides the sampling trajectory, PriorMDM Shafir et al. (2024) composes diffusion priors, DNO Karunratanakul et al. (2024) optimizes noise latents, TLControl Wan et al. (2024) refines part-VQ codes by L-BFGS, CondMDI Cohan et al. (2024) trains keyframe in-betweening, and MotionLCM Dai et al. (2024) accelerates inference orthogonally. *Duplicated branches*: OmniControl Xie et al. (2024), MaskControl Pinyoanuntapong et al. (2025) and InterControl Wang et al. (2024) wire a zero-initialized backbone replica layer by layer, buying per-layer access at a substantial fraction of the backbone's parameters; OmniControl also fine-tunes the trunk and MaskControl adds logit-side refinement at inference. Concurrent CMC Cai et al. (2026) cascades a trajectory stage into a text-conditioned one; semantic

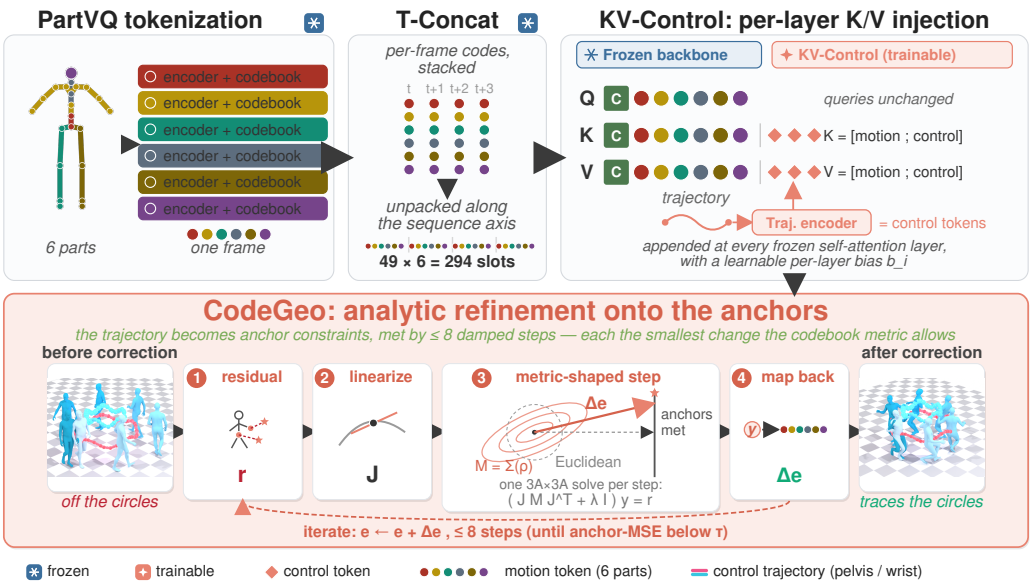


Figure 1: **KV-Control + CodeGeo.** PartVQ's $Q = 6$ part codebooks, unpacked by T-Concat into $294 = 49 \times 6$ (frame, part) attention slots on a frozen 20-layer backbone, turn a prompt into a feed-forward motion. **KV-Control** (10.5 M trainable, including a shared trajectory encoder) appends control-conditioned key/value entries at every self-attention layer, leaving the query stream unchanged. **CodeGeo** then replaces the inherited 1,600-iteration refinement with ≤ 8 damped analytic steps, each the smallest change a step metric $\mathbf{M} = \mathbf{\Sigma}(\rho)$ admits; $\rho \to \infty$ recovers the Euclidean minimum-norm step. Insets show multi-joint control of the pelvis and one wrist at $\rho = 10$.

editing Zhang et al. (2023d); Huang et al. (2024) targets semantics rather than spatial trajectories. *Attention-side memory (ours)*: the trajectory enters as time-indexed key/value memory *inside* self-attention, read by part-addressable queries. The primitive is shared with prefix-style conditioning Li & Liang (2021) and, more broadly, with bottleneck adapters Houlsby et al. (2019), low-rank updates Hu et al. (2022) and decoupled cross-attention Ye et al. (2023); new is the system built on it—a part-addressable substrate that gives the memory precise landing sites, and an analytic solve (§3.4) on the control-conditioned start it produces, which together reduce inference-time optimization Pinyoanuntapong et al. (2025); Wan et al. (2024) to an optional precision/compute knob.

3 Method

Our primary contribution, **CodeGeo** (§3.4), is an analytic anchor solve; it assumes a control-conditioned start that is already close to its anchors and a metric read off the generator's own codebooks. §3.1–3.3 describe the interface that supplies both (Figure 1): **PartVQ** factors body geometry across Q per-part codebooks; **T-Concat** unpacks them along the sequence axis so that each part-frame token is a distinct attention-addressable site; and **KV-Control** injects control-conditioned key/value pairs at every self-attention layer of the frozen backbone, at exactly those sites. PartVQ and T-Concat are pretrained once and frozen during control adaptation; §3.3 gives the training objective and the inherited inference-time refinement that CodeGeo replaces.

3.1 Anatomy-Aware Part Tokenization (PartVQ)

Partition derivation. Because control must address anatomy, the partition is clustered from motion data and then corrected for anatomical consistency (three limb chains repaired, bridge joints shared) rather than fixed by hand. Per-joint activations $a_j(t)$ (the ℓ_2 norm of joint j's parent-relative position, rotation and velocity blocks, which after z-scoring reduces to a joint-speed signal) are compared by maximum-lag cross-correlation, clustered hierarchically, and repaired so no kinematic limb chain splits across parts. Cutting at $Q = 6$ is the coarsest level at which each leg, the arms, the

root, the upper spine–neck group and the head resolve into their own parts; we fixed Q there before any control experiment (derivation in the appendix).

VQ-VAE architecture and objective. A 1D-conv encoder maps each part's motion window to latents quantized by per-part EMA-reset vector quantizers van den Oord et al. (2017); Zhang et al. (2023a); a shared decoder consumes the channel-concatenated quantized latents. Training combines smooth-L1 reconstruction Girshick (2015), VQ commitment van den Oord et al. (2017), and an auxiliary joint-position loss Zhang et al. (2023a).

3.2 KV-CONTROL

Addressable layout (T-Concat). A channel-concatenated layout (C-Concat)—our own layout-ablation baseline (§4), not a prior method—packs the Q part codebooks into one channel vector per timestep, forcing attention to recover per-part factors from a shared embedding. T-Concat unpacks them along the sequence axis into $S = T_{\text{tok}} \cdot Q$ tokens, making each part-frame token a distinct attention-addressable site—the landing positions the K/V branch below reads from. Each token at (t, q) receives an additive embedding $\mathbf{e}_t^{(\text{time})} + \mathbf{e}_q^{(\text{quant})}$. The Q-fold length increase is the price of per-quantizer identity: total backbone FLOPs rise $\approx 4.8\times$ over C-Concat (appendix), and our efficiency claims concern trainable parameters and test-time steps.

Text pathway and regularization. A frozen CLIP Radford et al. (2021) encoder feeds a jointly trained adapter whose outputs enter a gated text cross-attention block after every second self-attention layer; the post-norm backbone trains with masked-token prediction, a hybrid 2D masking regularizer, and classifier-free-guidance dropout Ho & Salimans (2022).

Control as attention memory. Given the addressable per-part layout, the trajectory signal enters as additional attention memory: at each self-attention layer, control-conditioned pseudo-tokens are appended to the key/value sets; the adapter only learns what control memory to add. The motion-token query stream and the backbone's forward pass are unchanged except for a low-rank side computation and an additive attention bias, and the text cross-attention weights stay frozen, preserving the text grounding the backbone already learned. At initialization the adapter is a *near-identity* mapping (below), and the frozen backbone is recovered exactly when the control branch is masked out.

Control encoder. At each sampling step s the controller decodes the current token state $\mathbf{z}_s$ via the frozen PartVQ decoder under stop-gradient, producing $\mathbf{p}_s^{\text{pred}} = \text{FK}(\text{sg}(D_{\text{VQ}}(\text{safe}(\mathbf{z}_s))))$.[1] $\mathbf{z}_s$ is the iterative unmask state at inference and the BERT-style random-masked ground-truth state at training, so the training residual $\mathbf{p}^{\text{tgt}} - \mathbf{p}_s^{\text{pred}}$ is non-trivial. The control input $\mathbf{c}_s = [\mathbf{p}^{\text{tgt}} - \mathbf{p}_s^{\text{pred}};\, \mathbf{p}^{\text{tgt}}] \odot \mathbf{m}$ is gated by the joint–frame mask $\mathbf{m}$. A 1D-conv stack downsamples to T_{tok} tokens of dimension d_{model}, the control features $\mathbf{f}_{\text{ctrl}}$ shared across injection layers. $\mathbf{p}_s^{\text{pred}}$ is read from the transformer's existing forward pass, and no extra frozen-base forward is needed (the per-sample costs in Table 1 include this).

Per-layer K/V injection. Each self-attention layer i is augmented as follows. Let $\mathbf{X}_i \in \mathbb{R}^{B\times S\times d_{\text{model}}}$ be the layer's motion-token input. The shared control features $\mathbf{f}_{\text{ctrl}}$ are mapped through a layer-specific *zero-initialized* low-rank down-projection followed by two parallel up-projections, $\mathbf{h}_i = \mathbf{f}_{\text{ctrl}}\mathbf{W}_i^{\text{down}}$, $\mathbf{C}_i^{\text{K}} = \mathbf{h}_i\mathbf{W}_i^{\text{K,up}}$, $\mathbf{C}_i^{\text{V}} = \mathbf{h}_i\mathbf{W}_i^{\text{V,up}}$, with $\mathbf{W}_i^{\text{down}} \in \mathbb{R}^{d_{\text{model}}\times r}$ initialized to zero and $\mathbf{W}_i^{\text{K,up}}, \mathbf{W}_i^{\text{V,up}} \in \mathbb{R}^{r\times d_{\text{model}}}$ randomly initialized. Augmented key/value input sequences are formed by concatenation, $\widetilde{\mathbf{K}}_i^{\text{in}} = [\mathbf{X}_i;\, \mathbf{C}_i^{\text{K}}]$ and $\widetilde{\mathbf{V}}_i^{\text{in}} = [\mathbf{X}_i;\, \mathbf{C}_i^{\text{V}}]$ of length $S + T_{\text{tok}}$. The query input remains $\mathbf{X}_i$, unchanged. The attention output of layer i is then the standard

$$\text{Attn}_i = \text{softmax}\Big(\frac{\mathbf{Q}_i\widetilde{\mathbf{K}}_i^{\top}}{\sqrt{d_h}} + \mathbf{B}_i\Big)\,\widetilde{\mathbf{V}}_i,$$

with $\mathbf{Q}_i = \mathbf{X}_i\mathbf{W}_i^{\text{Q}}$, $\widetilde{\mathbf{K}}_i = \widetilde{\mathbf{K}}_i^{\text{in}}\mathbf{W}_i^{\text{K,attn}}$, $\widetilde{\mathbf{V}}_i = \widetilde{\mathbf{V}}_i^{\text{in}}\mathbf{W}_i^{\text{V,attn}}$ formed from the frozen backbone projections, and $\mathbf{B}_i \in \mathbb{R}^{S\times(S+T_{\text{tok}})}$ a learnable additive bias that is zero on motion-token columns

[1] $\text{safe}(\mathbf{z}_s)$ replaces mask or out-of-codebook ids with a valid zero code before the frozen decoder.

and a single learnable scalar b_i on the control-token columns, initialized to a strongly negative value. Together with the zero-initialized down-projection, this double near-identity initialization lets the adapter learn its own layer-wise influence curve without a hand-tuned schedule.

3.3 Frozen-Backbone Training and Inference Refinement

Frozen-backbone discipline. During control adaptation the PartVQ tokenizer, CLIP text adapter, and T-Concat backbone are held fixed. The forward pass still traverses the full frozen graph—token embeddings, attention layers, PartVQ decoder, forward kinematics—gradients therefore route through it, but only the control encoder, per-layer K/V projections, and bias scalars b_i are trainable. The losses supervise the K/V-augmented transformer's output estimate $\hat{\mathbf{x}}_0^{\text{out}}$, not the detached state estimate $\mathbf{p}_s^{\text{pred}}$ the controller consumes (§3.2).

Adapter training objective. KV-Control is trained with AdamW on a masked cross-entropy term over motion tokens combined with a forward-kinematics trajectory loss on the controlled joints, $\mathcal{L}_{\text{KV}} = \lambda_{\text{CE}}\mathcal{L}_{\text{CE}} + \lambda_{\text{traj}}\mathcal{L}_{\text{traj}}$, where $\mathcal{L}_{\text{CE}}$ is the masked-token loss of MoMask Guo et al. (2024) on the per-part codebooks, and

$$\mathcal{L}_{\text{traj}} = \|\mathbf{m} \odot (\text{FK}(\hat{\mathbf{x}}_0^{\text{out}}) - \mathbf{p}^{\text{tgt}})\|_1$$

applies FK supervision to the predicted clean-token estimates. Because the predictions are discrete codebook tokens, $\mathcal{L}_{\text{traj}}$ is evaluated through a soft codebook relaxation: token logits define an expectation over each per-part codebook, which the frozen PartVQ decoder maps to motion before FK.

Inference-time refinement. The same differentiable path supports two-stage inference-time refinement—the shared inference protocol of the MaskControl/TLControl family Pinyoanuntapong et al. (2025); Wan et al. (2024): logits (Stage 1) or post-sampling token embeddings (Stage 2) are optimized while all model parameters stay fixed. §4.1 reports the adapter both feed-forward and under this refinement. Because this iterative protocol dominates inference cost, §3.4 develops its replacement—the primary contribution of this paper.

3.4 Codebook-Geometry Correction

We call the resulting refiner **Codebook-Geometry Correction** (CodeGeo): a few damped analytic steps that move a generated clip, by the smallest change its step metric admits, onto the set of motions that pass through the control anchors. It replaces the inherited $1{,}600$-iteration test-time optimization on both pelvis and multi-joint control.

Two structural observations. Write $\mathbf{e} \in \mathbb{R}^P$ for the decoded embeddings of the feed-forward output and $\mathbf{r}(\mathbf{e}) \in \mathbb{R}^{3A}$ for the position errors of the A commanded anchors under the frozen decoder–FK map. First, the constraints are few: $3A \ll P$ for every anchor set the protocol commands, making anchor satisfaction a small, massively underdetermined nonlinear least-squares problem whose update can be computed in the $3A$-dimensional dual space. Second, the KV-Control start is already control-conditioned—close enough to its anchors for a linearized solve, which leaves the inherited $1{,}600$-iteration budget with little to do (probe traces in the appendix).

The correction step. Each step linearizes the decoder–FK map at the current embeddings, $\mathbf{r}(\mathbf{e} + \Delta) \approx \mathbf{r} + \mathbf{J}\Delta$ with $\mathbf{J} \in \mathbb{R}^{3A \times P}$ the residual Jacobian, and asks for the correction that cancels that residual while moving as little as possible in a metric $\mathbf{M}$: $\min_\Delta \ \Delta^\top \mathbf{M}^{-1} \Delta$ subject to $\mathbf{J}\Delta = -\mathbf{r}$. Stationarity of the Lagrangian puts the solution in the $\mathbf{M}$-weighted row space of $\mathbf{J}$, $\Delta = -\mathbf{M}\mathbf{J}^\top \mathbf{y}$, and substituting it into the constraint leaves the $3A$-dimensional system $\mathbf{J}\mathbf{M}\mathbf{J}^\top \mathbf{y} = \mathbf{r}$. Damping that system for conditioning and for the trust region below gives the step we take,

$$\Delta = -\,\mathbf{M}(\rho)\mathbf{J}^\top \left(\mathbf{J}\mathbf{M}(\rho)\mathbf{J}^\top + \lambda\, \mathbf{I}\right)^{-1} \mathbf{r}. \tag{1}$$

The Jacobian is formed one reverse-mode row per residual component, batched across rows, and the linear system is only $3A \times 3A$—the whole benefit of solving in the dual. Each update is a damped, metric-weighted Gauss–Newton step; what is new is where it is taken and in what metric: in the

$3A$-dimensional dual, from a control-conditioned start close enough that ≤ 8 steps suffice, under a metric read off the generator's own codebooks whose one hyperparameter follows a rule (§4.4). Any symmetric positive-definite $\mathbf{M}$ defines a valid correction; we take it from the second-order statistics the generator already carries. For quantizer q, $\mathbf{M}_q = \mathbf{\Sigma}_q(\rho)$ is the covariance of that codebook's vectors, ridge-regularized by ρ and scale-normalized, assembled block-diagonally; it requires no training. Directions the codebook has spread along are ones the frozen decoder was trained to decode and are cheap to move along, while directions it never spans are expensive. The ridge ρ sets the metric's *anisotropy*—how strongly the step follows that codebook geometry: as $\rho \to \infty$, $\mathbf{M} \to \mathbf{I}$ and Eq. (1) degenerates to the standard Euclidean minimum-norm step, which is therefore a special case of CodeGeo rather than an alternative to it. Because the damping is a single scalar scaled to the mean diagonal of $\mathbf{J}\mathbf{M}\mathbf{J}^\top$, a uniform rescaling of $\mathbf{M}$ is inert; what changes the solve is how unevenly $\mathbf{M}$ prices the anchor rows—so ρ is a first-class hyperparameter, not an implementation detail. We study its effect and give a selection rule in §4.4.

Damping, acceptance, and cost. The damping λ is scaled to the system—the mean diagonal of $\mathbf{J}\mathbf{M}\mathbf{J}^\top$—and follows a trust region: a trial step is accepted if it reduces anchor MSE ($\lambda \times 0.5$), otherwise rolled back exactly ($\lambda \times 10$); the solver stops below an anchor-MSE tolerance τ or after at most 8 steps and returns its best observed iterate. Per step the cost is $3A$ reverse-mode passes for the Jacobian and one $3A \times 3A$ linear solve per damping trial (at most four). Solver hyperparameters and reproducibility details are in the appendix.

4 Experiments

4.1 Experimental Setup

We evaluate on HumanML3D Guo et al. (2022) under the MaskControl protocol Pinyoanuntapong et al. (2025) (CFG 3.25, $T = 10$ steps, seed 3407, batch 32). PartVQ uses $Q = 6$ parts (codebook 128, code dim 128); the T-Concat backbone is a 20-layer post-norm Transformer ($d_{\text{model}} = 384$, 8 heads, $S = 294$ tokens) with frozen CLIP ViT-B/32; KV-Control uses per-layer rank $r = 64$, zero-initialized $\mathbf{W}_i^{\text{down}}$ and control bias $b_i = -5$, for 10.5 M trainable parameters in total. Because test-time refinement of control targets is standard in this family Wan et al. (2024), every method is reported at an explicit, matched budget. We use three: **feed-forward** (none); **MaskControl-protocol refinement**, that paper's own released two-stage scheme (AdamW, 100 Stage-1 iterations per unmask step optimizing motion-token logits, then 600 Stage-2 iterations optimizing token embeddings), which is the regime all our refined rows use while prior-work rows are printed as published; and **CodeGeo** (§3.4), which replaces that refinement with no Stage-1 and ≤ 8 analytic steps ($\tau = 10^{-5}$ m^2, full anchor capacity (§4.3), and a step-metric ridge of $\rho = 0.1$ on pelvis and $\rho = 10$ on multi-joint control, per §4.4). Unless noted, numbers are means over *five repeats*: five evaluations of one trained adapter under different generation seeds (sampling, control density, anchor draws), not retraining runs; training variability is probed separately by an independent end-to-end retraining of the C-Concat adapter, which reproduces FID and Top-3 within one interval half-width and position error to 0.09 cm (Appendix C.4). Table 1 subscripts each mean with its 95% t-interval half-width (failure-rate columns: means only). A same-depth, same-width backbone with the codebooks channel-stacked (C-Concat) is of comparable unconditional quality (0.078 vs. 0.081 FID), yet the same adapter on it tracks 2.2 times worse under the shared refinement (1.10 vs. 0.51 cm) and 1.5 times worse under CodeGeo (0.34 vs. 0.24 cm at 0.036 higher FID, Table 13): the layout, not added capacity, carries the control gain.

4.2 Trajectory Control

The same KV-Control adapter architecture applies to two trajectory-control settings on HumanML3D: *single-joint* (pelvis only) and *multi-joint* (random subset of up to six joints from {pelvis, head, ℓ/r foot, ℓ/r wrist}, sampled per training example). Table 1 reports our rows across the inference regimes of §4.1 next to prior-work rows. The prior-work accuracy numbers are quoted as published in Pinyoanuntapong et al. (2025). The latencies are ours for every method, measured on one H200 with each method's released inference code, and the released MaskControl model is re-run for the same-trajectory qualitative comparisons in the appendix.

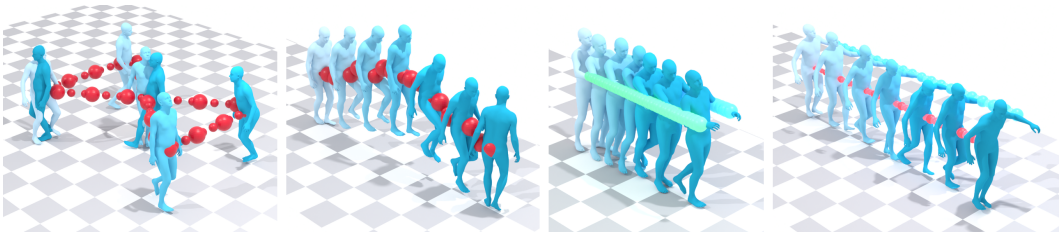


Figure 2: **Codebook-Geometry Correction** with ≤ 8 analytic steps. Spheres and tubes are *commanded* targets; poses are overlaid along each motion. Left two: pelvis control ($\rho = 0.1$) on an hourglass path and a descending staircase. Right two: multi-joint control ($\rho = 10$) on MaskControl's zombie-walk hand path, mirrored to both wrists Pinyoanuntapong et al. (2025), and a wall-slide commanding one hand while the pelvis follows its own path.

Method	Top-3 ↑	FID ↓	Skate ↓	Traj. E ↓	Loc. E ↓	Err. (cm) ↓	s/sam. ↓
Train on Pelvis (single-joint), prior-work Pinyoanuntapong et al. (2025)							
OmniControl	0.687	0.218	0.0547	3.87	0.96	3.38	24.61
TLControl	0.779	0.271	–	0.00	0.00	1.08	4.20
MaskControl	0.809	0.061	0.0547	0.00	0.00	0.98	3.83
Train on Pelvis, our re-runs (KV-Control)							
Ours, feed-forward	$0.809_{\pm 0.008}$	$0.076_{\pm 0.007}$	$0.0425_{\pm 0.0011}$	27.55	17.05	$29.88_{\pm 0.71}$	0.17
Ours, iterative refinement	$0.801_{\pm 0.005}$	$0.064_{\pm 0.009}$	$0.0444_{\pm 0.0011}$	0.00	0.00	$0.51_{\pm 0.01}$	4.48
Ours (CodeGeo, $\rho = 0.1$)	$\mathbf{0.802}_{\pm 0.008}$	$\mathbf{0.067}_{\pm 0.003}$	$\mathbf{0.0477}_{\pm 0.0014}$	**0.02**	**0.00**	$\mathbf{0.24}_{\pm 0.02}$	**0.21**
Train on All Joints (multi-joint), prior-work Pinyoanuntapong et al. (2025)							
OmniControl	0.693	0.310	0.0608	6.17	1.07	4.04	24.61
TLControl	0.782	0.256	–	0.00	0.00	1.11	4.20
MaskControl	0.805	0.083	0.0545	0.00	0.00	0.72	3.83
Train on All Joints, our re-runs (multi-joint KV-Control)							
Ours, feed-forward	$0.811_{\pm 0.006}$	$0.074_{\pm 0.009}$	$0.0433_{\pm 0.0006}$	37.04	22.56	$40.63_{\pm 0.48}$	0.19
Ours, iterative refinement	$0.802_{\pm 0.008}$	$0.053_{\pm 0.004}$	$0.0486_{\pm 0.0004}$	0.00	0.00	$0.90_{\pm 0.10}$	4.73
Ours (CodeGeo, $\rho = 10$)	$\mathbf{0.797}_{\pm 0.006}$	$\mathbf{0.072}_{\pm 0.009}$	$\mathbf{0.0545}_{\pm 0.0005}$	**0.00**	**0.00**	$\mathbf{0.25}_{\pm 0.011}$	**0.39**

Table 1: Trajectory control on HumanML3D. Prior-work rows (OmniControl Xie et al. (2024), TLControl Wan et al. (2024), MaskControl Pinyoanuntapong et al. (2025)) are as published in Pinyoanuntapong et al. (2025); ours are five-repeat means under that paper's released evaluation setting. Bold marks our analytic method per block. CodeGeo uses full anchor capacity at the step-metric ridge of §4.4 ($\rho = 0.1$ pelvis, $\rho = 10$ multi-joint). Traj. E and Loc. E are > 50 cm failure rates (%). *s/sam.* is the batch-1 median over the same 64 test clips on one H200, every method at its own released inference configuration; prior-work multi-joint rows reuse the pelvis timings, whose released settings are shared. Test-time budget per sample: MaskControl and our iterative row run the same 1,600-iteration refinement schedule; CodeGeo runs ≤ 8 analytic steps; OmniControl and TLControl run a diffusion or L-BFGS loop instead.

Findings. Under MaskControl's own released evaluation setting the parameter-thin K/V interface already halves their published pelvis error at parity on FID and Top-3, with less foot skate; the layout carries that accuracy (§4.5). **CodeGeo** then turns that feed-forward start into our headline row, halving that error again with ≤ 8 analytic steps in place of the inherited 1,600-iteration refinement. The same solver carries to multi-joint control once its step metric is re-tuned to the wider anchor spread ($\rho = 10$; §4.4), where it improves on MaskControl's published numbers on both axes. Top-3 slips by about one interval width as anchors are enforced ($0.811 \to 0.802 \to 0.797$): pulling a clip onto commanded positions moves it slightly off the text-conditioned mode.

Qualitative results. Figure 3 separates the two components on one sample; Figure 2 shows the corrected state on hand-designed trajectories at the reported operating points: pelvis control on an hourglass path and a descending staircase, and multi-joint control on MaskControl's published zombie-walk hand path and on a wall-slide demo commanding a hand and the pelvis together. Every

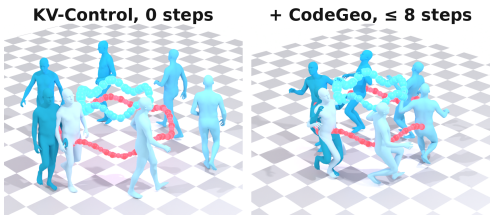


Figure 3: **What each component contributes.** One sample under multi-joint control, commanded pelvis (red) and left-wrist (cyan) targets, before and after CodeGeo. Left: KV-Control's feed-forward output with zero test-time steps is already control-conditioned but drifts off the targets. Right: CodeGeo's ≤ 8 analytic steps put it on them. Figure 2 shows only the right-hand state.

panel uses ≤ 8 analytic steps and no warm start. The appendix adds fixed-camera renders on a metric floor (making foot–ground contact directly inspectable) and same-trajectory comparisons against MaskControl and OmniControl on identical inputs and prompts, plus side-by-side video.

4.3 CodeGeo Replaces the Inherited Refinement

The two-stage refinement this family inherits was designed for adapters that start far from the anchors—MaskControl degrades from 0.98 to 40.41 cm with logit optimization off (Pinyoanuntapong et al., 2025, Table 5). On our adapter that budget idles: 97% of samples are already below the 10^{-5} m^2 anchor tolerance when Stage-2 begins, and 75% end Stage-2 worse than their best intermediate iterate (per-iteration probe in the appendix). Without the K/V adapter the identical schedule reaches 0.013 cm but collapses FID to 104 (appendix). Evaluated under the same protocol on both sides, with no Stage-1 and at most 8 analytic steps, CodeGeo halves the iterative row's position error at comparable FID and Top-3 in $1/21$ the per-sample time of that schedule—the same model, checkpoint and H200, so the ratio measures the solver alone. Discarding the schedule's wasted iterations does not close the gap: a plateau-stopped variant still trails CodeGeo (0.46 vs. 0.24 cm at the same FID, two-repeat probe) and remains $\sim 17\times$ slower (appendix). The same solver carries to multi-joint control at full anchor capacity (every commanded target, up to $1{,}176$ per sample, one batched solve per step) with no iterative warm-up: with the anisotropy matched to that wider spread ($\rho = 10$, §4.4) it improves on MaskControl's published numbers on both axes at ~ 0.39 s per sample, the same ≤ 8-step regime as pelvis.

4.4 Choosing the Step Metric: Anisotropy Follows the Anchor Set

CodeGeo has one free design quantity, the ridge ρ of its step metric (Eq. 1; smaller ρ, stronger anisotropy). We sweep it with everything else held fixed (solver, ≤ 8-step budget, tolerance and acceptance rule, full anchor capacity, checkpoints, seeds) and find that the two control settings respond in *opposite* directions in FID, while tracking error falls with weaker anisotropy in both (Figure 4, Table 2).

On pelvis control, where every anchor targets a joint of one part, strong anisotropy costs nothing in FID: the curve is flat over $\rho \in [0.03, 0.3]$ and rises toward the identity, whose tracking is the better of the two ends (0.21 vs. 0.24 cm). On multi-joint control, where anchors are drawn across up to six part codebooks, the same strong anisotropy is harmful. The default $\rho = 0.1$ is the one setting the sweep rules out: its five-repeat FID interval is separated from those of $\rho = 3, 10, 30$ and the Euclidean limit, and on tracking from all but $\rho = 3$. Above it the optimum is broad—within $\rho \in [3, 30]$ the five-repeat intervals overlap one another on both axes, and $\rho = 10$ and 30 also overlap the Euclidean limit on both—and we adopt $\rho = 10$ for its best FID point estimate in that band. Re-run on the validation split, the sweep selects the same operating points: $\rho = 0.1$ on pelvis (-0.024 ± 0.009 FID against the Euclidean limit, paired over five repeats) and the $[3, 30]$ band on multi-joint control, with $\rho = 10$ again the best point estimate (Table 4).

Selection rule. Read ρ off the anchor set: root and head anchors keep the raw covariance; limb end-effectors and body-spanning anchor sets relax it by one to two orders of magnitude. The damping in Eq. 1 is one scalar, so only uneven pricing across anchor rows acts, and with a

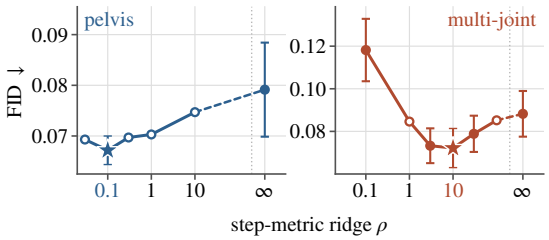


Figure 4: **Anisotropy sweep** of the ridge ρ (Eq. 1), all else fixed; the panels use different FID scales. Stars: adopted ρ. Filled: five-repeat means, 95% t-intervals. Open: single-repeat probes. $\rho \to \infty$: the Euclidean minimum-norm step, a limiting case of the same update.

ρ	FID ↓	Err. (cm) ↓
0.1 (pelvis)	0.118 ± 0.015	0.326 ± 0.017
3	0.073 ± 0.008	0.278 ± 0.031
10 (ours)	$\mathbf{0.072 \pm 0.009}$	$\mathbf{0.25 \pm 0.011}$
30	0.079 ± 0.009	0.248 ± 0.016
∞ (Eucl.)	0.088 ± 0.011	0.236 ± 0.007

Table 2: Step-metric anisotropy on multi-joint control (five repeats, $\pm$ 95% t). $\rho \to \infty$ is the Euclidean limit of Eq. 1. Pelvis control moves the other way in FID (Figure 4).

single anchor ρ is inert (below). In the solver census the mismatch shows up as refused trial steps ($4.5\times$ more often on multi-joint control at the default anisotropy than under the Euclidean metric, with 7.3% of samples short of tolerance against 2.0%), and the steps it does accept land where the decoder scores poorly: extra steps recover tracking but not FID, which stays at 0.119 through four times the budget (appendix).

Anchor count and controlled joint. Fixing one joint and one anchor density per run separates the two (Table 11; six joints, three paired repeats per cell). With a single anchor ρ is inert for every joint ($|\Delta\text{FID}| \leq 0.003$): one row gives the metric no uneven pricing to act on. From five anchors up the effect grows with the count, and the joint sets its sign: pelvis and head anchors gain 0.007–0.028 FID from strong anisotropy at every multi-anchor density, while both wrists and both feet lose up to 0.27, the same sign in every repeat of 14 of the 15 multi-anchor cells (per-cell numbers and the protocol-mixture head and wrist tests in the appendix). ρ is the only quantity of the correction selected by sweeping, and every tested value in $[3, 30]$ reproduces its conclusions.

4.5 Substrate Portability

The key/value interface is not tied to our substrate. Trained onto MaskControl's RVQ+MoMask stack Pinyoanuntapong et al. (2025) and run against the authors' released ControlNet under the shared refinement protocol (five repeats), it matches them on Top-3 (0.789 vs. 0.794) and foot skate (0.0475 vs. 0.0465), tracks to 1.09 ± 0.01 cm against their 0.96 cm, and reaches FID 0.130 against 0.069—on under half the trainable parameters (9.6M vs. 21.4M). RVQ+MoMask exposes one token per frame, so the injected memory has no per-part landing sites: the mechanism transfers, but its precision needs the part-addressable layout.

5 Conclusion

CodeGeo replaces the 1,600-iteration test-time refinement that trajectory-controlled masked motion transformers inherit with ≤ 8 damped analytic steps under a metric read off the generator's own part codebooks: on pelvis control it halves the refined error again (0.51 → 0.24 cm) in 1/21 the per-sample time of that schedule, and on multi-joint control it reaches 0.25 cm at FID 0.072, better than MaskControl's published 0.72 cm and 0.083 on both axes. It needs a control-conditioned start, which **KV-Control** supplies with 10.5 M trainable parameters over a frozen backbone, and one tuned hyperparameter, whose setting follows the anchor set: root and head anchors keep the raw codebook anisotropy, limb end-effectors and body-spanning sets relax it. Next steps are KIT-ML Plappert et al. (2016), cross-skeleton transfer, and a perceptual study.

AI use statement

Large language models were used as coding and writing assistants throughout this work. Concretely: (i) the training, evaluation, mesh-fitting and rendering tooling was written with LLM assistance;

every script was reviewed and executed by the authors, and every number we report for our own runs was produced by running that code, never by a model. (ii) LLM assistance was used to draft and edit prose and to check the manuscript for internal consistency; all claims and citations were verified by the authors against the cited sources. (iii) An LLM-based review loop was used as an internal code-review and manuscript-audit step. No LLM was used to generate, label or augment any training or evaluation data. No LLM is an author, and the authors take full responsibility for all content, including any errors.

Reproducibility statement

Every number we computed derives from macros locked to the evaluation logs; prior-work rows are quoted as published. Appendix A lists the training and solver settings, the hyperparameter-selection protocol (the ridge ρ is the only quantity selected by sweeping, every tested value in the band it identifies reproduces the conclusions, and the selection reproduces on the validation split), and the solver's numerical census; Appendix B gives the exact evaluation protocol, which is MaskControl's released setting applied unchanged, together with the timing harness and FLOPs accounting. The evaluator is the HumanML3D benchmark release. Training, evaluation and analysis code will be released as an open-source repository upon acceptance.

Ethics statement

This work uses HumanML3D, an existing, licence-gated motion-capture corpus derived from AMASS; it contains no personally identifying imagery and we redistribute no part of it. No human-subject study was conducted. The intended use is character animation and embodied-agent tooling; as with any motion generator, outputs could be used to animate characters without a performer's involvement, which we note but do not consider a novel risk introduced by this work. The method adds no new data-collection surface.

References

Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J. Black. Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image. In *European Conference on Computer Vision (ECCV)*, 2016.

Deli Cai, Haoyang Ma, and Changxing Ding. Coordinating multiple conditions for trajectory-controlled human motion generation. *arXiv preprint arXiv:2605.13729*, 2026.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. MaskGIT: Masked generative image transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Xin Chen, Biao Jiang, Wen Liu, Zilong Huang, Bin Fu, Tao Chen, and Gang Yu. Executing your commands via motion diffusion in latent space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Seunggeun Chi, Hyung-gun Chi, Hengbo Ma, Nakul Agarwal, Faizan Siddiqui, Karthik Ramani, and Kwonjoon Lee. M2D2M: Multi-motion generation from text with discrete diffusion models. In *European Conference on Computer Vision (ECCV)*, 2024.

Setareh Cohan, Guy Tevet, Daniele Reda, Xue Bin Peng, and Michiel van de Panne. Flexible motion in-betweening with diffusion models. In *ACM SIGGRAPH 2024 Conference Proceedings*, 2024.

Wenxun Dai, Ling-Hao Chen, Jingbo Wang, Jinpeng Liu, Bo Dai, and Yansong Tang. MotionLCM: Real-time controllable motion generation via latent consistency model. In *European Conference on Computer Vision (ECCV)*, 2024.

Ross Girshick. Fast R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 1440–1448, 2015.

Chuan Guo, Shihao Zou, Xinxin Zuo, Sen Wang, Wei Ji, Xingyu Li, and Li Cheng. Generating diverse and natural 3d human motions from text. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Chuan Guo, Yuxuan Mu, Muhammad Gohar Javed, Sen Wang, and Li Cheng. MoMask: Generative masked modeling of 3D human motions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning (ICML)*, 2019.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

Yiming Huang, Weilin Wan, Yue Yang, Chris Callison-Burch, Mark Yatskar, and Lingjie Liu. CoMo: Controllable motion generation through language guided pose code editing. In *European Conference on Computer Vision (ECCV)*, 2024.

Biao Jiang, Xin Chen, Wen Liu, Jingyi Yu, Gang Yu, and Tao Chen. MotionGPT: Human motion as a foreign language. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Korrawe Karunratanakul, Konpat Preechakul, Supasorn Suwajanakorn, and Siyu Tang. GMD: Guided motion diffusion for controllable human motion synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Korrawe Karunratanakul, Konpat Preechakul, Emre Aksan, Thabo Beeler, Supasorn Suwajanakorn, and Siyu Tang. Optimizing diffusion noise can serve as universal motion priors. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pp. 4582–4597, 2021.

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 34(6):1–16, 2015.

Zichong Meng, Zeyu Han, Xiaogang Peng, Yiming Xie, and Huaizu Jiang. Absolute coordinates make motion generation easy, 2025a.

Zichong Meng, Yiming Xie, Xiaogang Peng, Zeyu Han, and Huaizu Jiang. Rethinking diffusion for text-driven human motion generation: Redundant representations, evaluation, and masked autoregression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025b.

Ekkasit Pinyoanuntapong, Muhammad Usama Saleem, Pu Wang, Minwoo Lee, Srijan Das, and Chen Chen. BAMM: Bidirectional autoregressive motion model. In *European Conference on Computer Vision (ECCV)*, 2024a.

Ekkasit Pinyoanuntapong, Pu Wang, Minwoo Lee, and Chen Chen. MMM: Generative masked motion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024b.

Ekkasit Pinyoanuntapong, Muhammad Usama Saleem, Korrawe Karunratanakul, Pu Wang, Hongfei Xue, Chen Chen, Chuan Guo, Junli Cao, Jian Ren, and Sergey Tulyakov. MaskControl: Spatio-temporal control for masked motion synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

Matthias Plappert, Christian Mandery, and Tamim Asfour. The KIT motion-language dataset. *Big Data*, 4(4):236–252, 2016.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, 2021.

Yonatan Shafir, Guy Tevet, Roy Kapon, and Amit H. Bermano. Human motion diffusion as a generative prior. In *International Conference on Learning Representations (ICLR)*, 2024.

Guy Tevet, Sigal Raab, Brian Gordon, Yonatan Shafir, Daniel Cohen-Or, and Amit H. Bermano. Human motion diffusion model. In *International Conference on Learning Representations (ICLR)*, 2023.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Weilin Wan, Zhiyang Dou, Taku Komura, Wenping Wang, Dinesh Jayaraman, and Lingjie Liu. TLControl: Trajectory and language control for human motion synthesis. In *European Conference on Computer Vision (ECCV)*, 2024.

Zhenzhi Wang, Jingbo Wang, Yixuan Li, Dahua Lin, and Bo Dai. InterControl: Zero-shot human interaction generation by controlling every joint. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Yiming Xie, Varun Jampani, Lei Zhong, Deqing Sun, and Huaizu Jiang. OmniControl: Control any joint at any time for human motion generation. In *International Conference on Learning Representations (ICLR)*, 2024.

Hu Ye, Jun Zhang, Sibo Liu, Xiao Han, and Wei Yang. Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models, 2023.

Ling-An Zeng, Guohong Huang, Gaojie Wu, and Wei-Shi Zheng. Light-T2M: A lightweight and fast model for text-to-motion generation. In *AAAI Conference on Artificial Intelligence*, 2025.

Jianrong Zhang, Yangsong Zhang, Xiaodong Cun, Yong Zhang, Hongwei Zhao, Hongtao Lu, Xi Shen, and Ying Shan. T2M-GPT: Generating human motion from textual descriptions with discrete representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023a.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023b.

Mingyuan Zhang, Xinying Guo, Liang Pan, Zhongang Cai, Fangzhou Hong, Huirong Li, Lei Yang, and Ziwei Liu. ReMoDiffuse: Retrieval-augmented motion diffusion model. In *International Conference on Computer Vision (ICCV)*, 2023c.

Mingyuan Zhang, Huirong Li, Zhongang Cai, Jiawei Ren, Lei Yang, and Ziwei Liu. FineMoGen: Fine-grained spatio-temporal motion generation and editing. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023d.

Mingyuan Zhang, Zhongang Cai, Liang Pan, Fangzhou Hong, Xinying Guo, Lei Yang, and Ziwei Liu. MotionDiffuse: Text-driven human motion generation with diffusion model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024a.

Zeyu Zhang, Akide Liu, Ian Reid, Richard Hartley, Bohan Zhuang, and Hao Tang. Motion mamba: Efficient and long sequence motion generation. In *European Conference on Computer Vision (ECCV)*, 2024b.

Qiran Zou, Shangyuan Yuan, Shian Du, Yu Wang, Chang Liu, Yi Xu, Jie Chen, and Xiangyang Ji. ParCo: Part-coordinating text-to-motion synthesis. In *European Conference on Computer Vision (ECCV)*, pp. 126–143, Milan, Italy, 2024. Springer.

A METHOD DETAILS

PartVQ part listing. The data-derived, anatomy-corrected partition used throughout the paper (`skeleton_partition.json`) decomposes the 22-joint HumanML3D skeleton into $Q = 6$ parts via Algorithm 1: per-joint activations are the ℓ_2 norm of the concatenated parent-relative position, 6D rotation, and velocity blocks of joint j. Two of these blocks have constant norm per joint—the parent-relative position is a bone vector of fixed length, and a 6D rotation always has norm $\sqrt{2}$—so $a_j(t)$ is a monotone function of joint speed, and the subsequent z-scoring makes the clustering a grouping of joints by the co-activation of this monotone speed proxy. Pairwise lag-aware cross-correlation similarities ($L=4$ frames, 200,000 training frames sampled from the full train split of 23,368 motions) are clustered by average-linkage hierarchical clustering, cut at 6 clusters, and a kinematic-chain integrity post-step reassigns each predefined limb chain to its majority label. Three chain fixes are applied (left_leg {1,4,7,10}, right_leg {2,5,8,11}, right_arm {14,17,19,21} all moved to their majority labels) so no chain splits across parts. A boundary-overlap preset then duplicates a small set of bridge joints (pelvis/spine1/spine2/neck/collars) into adjacent parts to ease cross-part attention at anatomical interfaces. The resulting six parts are summarized in Table 3.

ALGORITHM 1: PartVQ partition derivation (data-driven, anatomy-aligned).

Input : Motion corpus $\{M^{(n)}\}_n$ with $M^{(n)} \in \mathbb{R}^{T_n \times 263}$;
parent map π;
limb chains $\mathcal{C} = \{$l/r-arm, l/r-leg$\}$;
lag window $L = 4$;
target parts Q.
Output: Per-joint partition label $\ell \in \{1, \ldots, Q\}^{22}$.

`/* (1) per-joint activation` $a_j(t)$`: norm of the concatenated parent-relative position, rotation and velocity blocks */`
for $M^{(n)}$, t, j **do**
  $\mathbf{r}_j(t) \leftarrow M_t^{(n)}[\text{ric}_j] - M_t^{(n)}[\text{ric}_{\pi(j)}]$ (raw if $j=0$);
  $f_j(t) \leftarrow [\,\mathbf{r}_j(t);\ M_t^{(n)}[\text{rot6d}_j];\ M_t^{(n)}[\text{vel}_j]\,]$;
  $a_j(t) \leftarrow \| f_j(t) \|_2$;
end
$A \leftarrow \text{zscore}(a)$ across all frames;
`/* (2) lag-aware similarity */`
for *joint pair* (i, j) **do**
  $s_{ij} \leftarrow \max_{|\tau| \le L} |\text{corr}(A_i[t], A_j[t+\tau])|$
end
`/* (3) hierarchical clustering on` $1 - s$ `*/`
$\ell \leftarrow$ average-linkage agglomerative clustering on the condensed dissimilarity $1 - s$, cut at Q clusters;
`/* (4) kinematic-chain integrity post-step */`
for $c \in \mathcal{C}$ **do**
  $\ell[c] \leftarrow \text{Majority}(\ell[c])$;
end
return ℓ;

Training hyperparameters (beyond §4.1). **PartVQ**: motion windows $T = 64$, encoder downsamples to $T_{\text{tok}} = 16$ via 2 stride-2 ResNet1D blocks; trained 300,000 iters at batch 256 with AdamW (lr 2×10^{-4}), loss weights $\lambda_{\text{commit}} = 0.02$, $\lambda_{\text{vel}} = 0.5$. PartVQ is trained on 64-frame crops; at evaluation and generation time the same convolutional tokenizer is applied to 196-frame sequences, yielding 49 tokens per part. **T-Concat backbone**: $d_{\text{ff}} = 1536$; CLIP text adapter is a 4-layer TransformerEncoder; cross-attention scalar gates initialized to 0.01; hybrid 2D masking applied to 30% of training samples; CFG dropout 0.1. Base backbone converges to FID $= 0.081$, Top-3 $= 0.797$. **KV-Control adapter**: control encoder downsamples 196-frame controls to $T_{\text{tok}} = 49$ tokens; loss weights $\lambda_{\text{CE}} = 0.1$, $\lambda_{\text{traj}} = 0.9$; AdamW (lr 2×10^{-4}), 4,000-iter warm-up, gradient-clip 1.0, step-decay (factor 0.1) at iter 50,000; per-rank batch was hardware-dependent (512/rank on 2×H200, or 128/rank on 4×A100). **Training cost**: adapting the control side on the frozen backbone took ≈ 58 h wall-clock on 2×H200 for the pelvis adapter (≈ 120 GPU-h) and ≈ 131 h on 4×A100 for the multi-joint adapter (≈ 525 GPU-h), measured as the span of the TensorBoard event files and

P	Anatomical role	Base joints	Overlap sources	Dims
0	root/torso anchor	pelvis, spine1	—	19
1	upper body & arms	spine2, l/r-collar, l/r-shoulder, l/r-elbow, l/r-wrist	spine1, neck	132
2	right leg	r-hip, r-knee, r-ankle, r-foot	pelvis, spine1, spine2	81
3	upper spine / neck	spine3, neck	spine2, l/r-collar	60
4	left leg	l-hip, l-knee, l-ankle, l-foot	pelvis, spine1, spine2	81
5	head	head	neck	24

Table 3: Final PartVQ 6-part decomposition used throughout the paper. *Base joints* are the clustering output after chain integrity; *overlap sources* are bridge joints duplicated into the part by the boundary-overlap preset; *dims* is the resulting channel count of that part's codebook input within the 263-dim HumanML3D feature (summed channels of base ∪ overlap joints).

therefore inclusive of restarts. The T-Concat backbone is pretrained once and reused across the T-Concat KV-Control configurations; the C-Concat layout ablation and the RVQ+MoMask port each required their own backbone. **Software**: Python 3.10, PyTorch 2.8 (CUDA 12.6), on Linux (RHEL 8); evaluation uses the feature extractors and metric code of the HumanML3D benchmark release.

Inference algorithm. Algorithm 2 summarizes how the two shared-protocol regimes (feed-forward and MaskControl-protocol refinement, §4.1) traverse the same trained adapter; at our CodeGeo operating point the two iterative loops below are replaced outright by the analytic solve of §A.1. At every sampling step the controller decodes the partially revealed token state via the frozen PartVQ decoder under stop-gradient, builds the joint–frame masked control input $\mathbf{c}_s$ (target + residual), and injects K/V pseudo-tokens at every self-attention layer. The feed-forward point returns immediately; the refinement point adds the inherited Stage-1 logit-optimization inner loop (100 iterations per unmask step) and a Stage-2 embedding-refinement loop (600 iterations, AdamW lr 0.06) on the post-sampling token embeddings.

ALGORITHM 2: KV-Control inference under the shared-protocol regimes (R=off: feed-forward; R=on: MaskControl-protocol refinement). At our CodeGeo operating point both R=on loops are replaced by the analytic solve of §A.1.

Input: text y; targets $\mathbf{p}^{\text{tgt}}$, mask $\mathbf{m}$; refine flag $R \in \{\text{off}, \text{on}\}$
Output: motion tokens $\mathbf{z}_0$ (R=off) or the motion decoded from the refined embeddings (R=on)
$\mathbf{z}_T \leftarrow$ all-mask;
for $s = T, \ldots, 1$ **do**
  $\mathbf{p}_s^{\text{pred}} \leftarrow \text{FK}(\text{sg}(D_{\text{VQ}}(\text{safe}(\mathbf{z}_s))))$;
  $\mathbf{c}_s \leftarrow [\mathbf{p}^{\text{tgt}} - \mathbf{p}_s^{\text{pred}};\ \mathbf{p}^{\text{tgt}}] \odot \mathbf{m}$;
  $\boldsymbol{\ell}_s \leftarrow \text{Transformer}(\mathbf{z}_s, y;\ \mathbf{c}_s)$ (K/V injection);
  if $R = \textit{on}$ **then**
    Stage-1: AdamW on logits $\boldsymbol{\ell}_s$, 100 iters, $\nabla \mathcal{L}_{\text{traj}}$;
  end
  $\mathbf{z}_{s-1} \leftarrow$ unmask via $\boldsymbol{\ell}_s$ with K/V context $\mathbf{c}_s$;
end
if $R = \textit{on}$ **then**
  Stage-2: $\mathbf{e} \leftarrow \text{softmax}(\boldsymbol{\ell}_1/\tau_{\text{s}})\,\mathbf{E}$ (soft code mixture of the final logits, sampling temperature τ_{s});
  AdamW on $\mathbf{e}$, 600 iters, $\nabla \mathcal{L}_{\text{traj}}$; **return** $D_{\text{VQ}}(\mathbf{e})$;
end
return $\mathbf{z}_0$

A.1 Analytic-Step Solver: Reproducibility Details

Hyperparameters. Damping is initialized at $\lambda_0 = 10^{-2}$ and follows a trust region: an accepted trial halves λ (floor 10^{-6}), a rejected trial multiplies it by 10 and restores the embeddings from an exact snapshot (up to four trials per step; a step with no accepted trial terminates the solve at the best observed iterate). The damping term is scaled by $\mu = \text{mean}\,\text{diag}(\mathbf{J}\boldsymbol{\Sigma}\mathbf{J}^\top)$. The stopping tolerance is $\tau = 10^{-5}\,\text{m}^2$ full-mask anchor MSE; the step cap is 8 for both single- and multi-joint control.

setting	ρ	FID	Pos. Err. (cm)	ΔFID vs. $\rho \to \infty$
pelvis	0.1	0.118 ± 0.024	0.24 ± 0.02	-0.024 ± 0.009
pelvis	∞	0.142 ± 0.024	0.22 ± 0.02	—
multi-joint	0.1	0.166 ± 0.014	0.33 ± 0.03	$+0.044 \pm 0.009$
multi-joint	3	0.108 ± 0.010	0.27 ± 0.01	-0.015 ± 0.006
multi-joint	10	0.105 ± 0.013	0.25 ± 0.00	-0.018 ± 0.006
multi-joint	30	0.111 ± 0.012	0.24 ± 0.00	-0.012 ± 0.004
multi-joint	∞	0.122 ± 0.015	0.24 ± 0.01	—

Table 4: The ρ sweep re-run on the HumanML3D validation split, five repeats, $\pm 95\%$ t; every other setting as in Table 2. The last column is the paired per-repeat FID difference against the Euclidean limit of the same setting.

Each quantizer's metric $\mathbf{\Sigma}_q$ is the covariance of its 128 codebook vectors plus a ridge of ρ times the mean diagonal ($\rho = 0.1$ on pelvis and 10 on multi-joint control, §4.4), then scale-normalized to unit mean diagonal. The anchor capacity is set to each task's full commanded target set (up to 196 for single-joint dense control, 1,176 for multi-joint), assembled and solved as one batched step; acceptance, stopping, and reporting always use the full anchor mask. Trial acceptance requires a decrease of the full-mask anchor MSE.

Hyperparameter selection protocol. The ridge ρ is the one quantity selected by sweeping: the sweep of §4.4 runs under the same released evaluation protocol as every reported number, with five repeats at $\rho \in \{0.1, 3, 10, 30, \infty\}$ (multi-joint) and at $\rho = 0.1$ and the Euclidean limit (pelvis). On multi-joint control we adopt $\rho = 10$ for its best FID point estimate inside a band, $\rho \in [3, 30]$, whose five-repeat intervals overlap one another on both axes ($\rho = 10$ and 30 also overlap the Euclidean limit on both)—the selection is of a band, not a tuned point, and every tested value in it reproduces the paper's conclusions. The remaining solver constants were fixed before any sweep and never re-selected: $\lambda_0 = 10^{-2}$ is the standard Levenberg–Marquardt initialization, $\tau = 10^{-5}\,\mathrm{m}^2$ matches the refinement protocol's tolerance, and the 8-step cap was set by the first configuration tried; Table 8 probes the step cap after the fact and finds the adopted operating points insensitive to it; Table 9 sweeps λ_0 at the ruled-out anisotropy and finds no value that rescues it. Across the five-repeat cells the 95% t-intervals span ± 0.003–0.016 FID, ± 0.007–0.10 cm position error, and ± 0.005–0.011 Top-3 (feed-forward cells are wider); per-cell intervals appear in the sweep tables and wherever a $\pm$ is printed. The selection also reproduces on the validation split (1,460 clips, disjoint from test): re-running every cell there with five repeats keeps $\rho = 0.1$ on pelvis and the $[3, 30]$ band on multi-joint control, with $\rho = 10$ again the best FID point estimate (Table 4; FID rises as the sample count falls, so only within-split comparisons are read).

Step-metric spectrum. Each quantizer's metric block $\mathbf{M}_q = \mathbf{\Sigma}_q(\rho)$ is far from a mild reweighting at the single-joint default $\rho = 0.1$ (Table 5). Trace-normalized to 128, its participation-ratio effective rank is 2.0–4.9 of 128 directions, and 93–97% of directions sit at the ridge floor 0.0909, eleven times more expensive to move along than under $\mathbf{M} = \mathbf{I}$, while 4–9 directions per block are cheaper by up to 46–89×.

part	cond. number	eff. rank	top direction
P5 neck+head	980	2.02	69.6%
P2 right leg	846	2.50	60.1%
P4 left leg	706	3.24	50.1%
P3 upper spine/neck	701	3.05	49.8%
P0 root+spine1	655	2.83	46.6%
P1 spine-neck+arms	504	4.89	35.8%

Table 5: Codebook-covariance metric blocks at $\rho = 0.1$ (trace-normalized to 128). Effective rank is the participation ratio $(\sum_i \lambda_i)^2 / \sum_i \lambda_i^2$.

Solver census. Table 6 instruments the solve itself over $4{,}640$ samples per cell (every line reporting a complete, self-consistent outcome partition). No sample ever stalls, no batched solve ever raises, and no solution is ever non-finite, in any cell: the multi-joint regression at the default anisotropy is not a numerical failure. The only rejection cause that occurs is a trial step that does not reduce anchor error, and it is $4.5\times$ more frequent at $\rho = 0.1$ on multi-joint control than under the Euclidean limit, with 39% more trials and 7.3% of samples still above tolerance when the step budget ends.

setting	metric	stall	exc.	non-fin.	no-impr.	exh.
pelvis	$\rho = 0.1$	0	$0/1039$	0	166	0.9%
pelvis	Euclidean	0	$0/899$	0	80	0.3%
multi-joint	$\rho = 0.1$	0	$0/1786$	0	1129	7.3%
multi-joint	Euclidean	0	$0/1287$	0	250	2.0%

Table 6: Solver census, one repeat per cell, $4{,}640$ samples each. *Exhausted* = above tolerance when the ≤ 8-step budget ended.

How many steps the solve actually uses. The 8-step budget is a cap, not a typical cost. Instrumenting the adopted multi-joint configuration (one repeat, $4{,}640$ samples), the $4{,}551$ samples that reach tolerance do so at a median of 3 steps and a mean of 3.46: 82% finish within four, and 1.3% use the full budget (Table 7). No sample stalls. The 89 that remain above tolerance when the budget ends correspond to the 1.9% unconverged share at $K = 8$ in Table 8; raising the cap recovers tolerance for nearly all of them (0.1% remain at $K = 32$) without moving FID, so the shipped $K = 8$ budget loses nothing.

steps	1	2	3	4	5	6	7	8
samples	86	1023	1555	1084	436	205	101	61
%	1.9	22.5	34.2	23.8	9.6	4.5	2.2	1.3

Table 7: Steps used by the $4{,}551$ of $4{,}640$ multi-joint samples that reach tolerance, one repeat at the reported operating point. The 8 column is samples still active when the budget ended that were inside tolerance on the final decode; the 89 that were not are excluded and appear as the 1.9% unconverged share at $K = 8$ in Table 8. Peak allocated memory over this run, generator included, is 29.5 GiB on one A100 80 GB.

Where the corrected embeddings sit. CodeGeo never re-quantizes, and the embedding it starts from is not on the lattice either: like Stage-2, it starts from $\mathrm{softmax}(\boldsymbol{\ell}_1/\tau_{\mathrm{s}})\,\mathbf{E}$, the convex mixture of code vectors formed from the final logits at the sampling temperature (Algorithm 2); the feed-forward rows of Table 1 decode the sampled tokens instead. The solution is therefore a point in the space the codebook spans rather than a nearby code—the metric is read off the codebook, the solution is not confined to it—and the corrected row's FID is no worse than the feed-forward start it corrects (Table 1).

Step-budget sweep. The main text (§4.4) summarizes a sweep of the solver's step cap K; Table 8 and Figure 5 report it in full. Four cells—pelvis and multi-joint control, each at $\rho = 0.1$ and $\rho = 10$—are run at $K \in \{1, 2, 3, 4, 6, 8, 16, 24, 32\}$ with every other flag fixed to the operating configuration, one repeat per point, $4{,}640$ samples each. Because a trial step is accepted only if it reduces anchor MSE, that quantity is monotone in K by construction and position error tracks it; the informative quantities are the fraction of samples still above tolerance when the budget ends and the FID of what ships. Three readings. First, the adopted operating points saturate the paper's budget: at $K = 8$ only 1.9% (multi-joint) and 0.9% (pelvis) of samples remain above tolerance, and FID does not move from $K = 8$ to $K = 32$. Second, at the ruled-out anisotropy tracking does recover with more steps (7.3% above tolerance at $K = 8$, 0.8% at $K = 32$), but FID settles at ≈ 0.119 by $K = 8$ and stays there through four times the budget: the accepted steps land somewhere the decoder scores poorly, and extra budget does not repair them. Third, the two pelvis curves converge at nearly identical rates while their FID stays separated throughout, so on single-part anchors the strong anisotropy costs no convergence at all.

	Pelvis, $\rho=0.1$		Pelvis, $\rho=10$		Multi-joint, $\rho=10$		Multi-joint, $\rho=0.1$	
K	unc.	FID	unc.	FID	unc.	FID	unc.	FID
1	77.2%	0.081	75.3%	0.092	97.9%	0.094	98.5%	0.185
2	30.7%	0.078	30.9%	0.084	76.2%	0.077	83.4%	0.126
3	12.9%	0.076	11.6%	0.082	42.0%	0.072	56.4%	0.119
4	5.8%	0.075	4.5%	0.081	19.1%	0.071	35.5%	0.118
6	1.6%	0.075	1.1%	0.081	5.4%	0.071	15.8%	0.119
8	0.9%	0.075	0.5%	0.080	1.9%	0.071	7.3%	0.119
16	0.3%	0.075	0.1%	0.080	0.3%	0.071	2.0%	0.119
24	0.2%	0.075	0.0%	0.080	0.2%	0.071	1.4%	0.120
32	0.1%	0.075	0.0%	0.080	0.1%	0.071	0.8%	0.120

Table 8: Step-budget sweep: share of samples above the anchor tolerance when the budget ends (unc.) and final FID, per step cap K. One repeat per point, 4,640 samples per cell; every flag other than K is the operating configuration. The paper's budget is $K=8$.

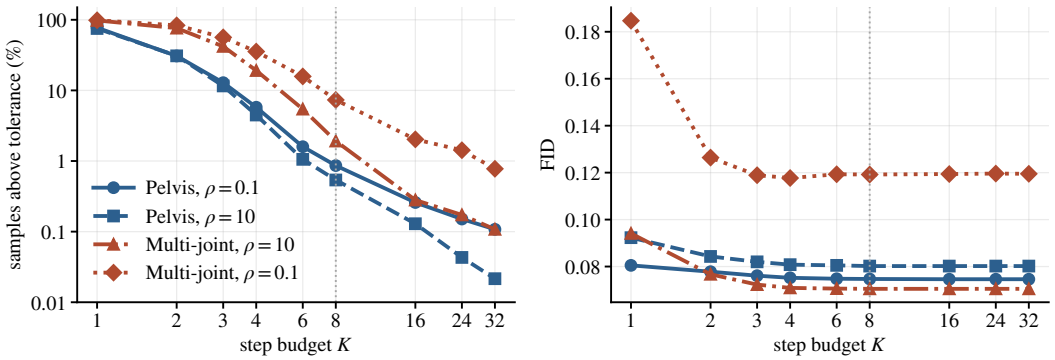


Figure 5: Step-budget sweep. Left: share of samples above tolerance (log scale). Right: final FID. Dotted line marks the paper's $K=8$ budget.

λ_0	FID	Pos. Err. (cm)	unc.
10^{-3}	0.189	0.36	9.0%
$3\cdot10^{-3}$	0.149	0.31	8.1%
10^{-2}	0.119	0.32	7.3%
$3\cdot10^{-2}$	0.096	0.35	8.8%
10^{-1}	0.081	0.41	11.9%
1	0.066	0.70	33.3%

Table 9: Damping-constant sweep at the ruled-out anisotropy (multi-joint, $\rho=0.1$, $K=8$). The paper's default is $\lambda_0=10^{-2}$. For reference, the adopted $\rho=10$ point reaches 0.24 cm at FID 0.071.

No damping constant rescues the ruled-out anisotropy. The mechanism of §4.4 says one scalar damping cannot serve rows the metric re-prices against each other; the nearest alternative reading is that the multi-joint regression at $\rho=0.1$ merely reflects a damping constant tuned for the wrong scale. Sweeping λ_0 over three orders of magnitude at that setting ($K=8$, Table 9) rules this out: small λ_0 leaves FID at 0.12–0.19, large λ_0 buys FID back only by suppressing the correction itself—at $\lambda_0=1$, a third of samples end above tolerance and position error triples. No point on the λ_0 curve reaches the adopted operating point's 0.24 cm at FID 0.071; the metric, not the constant, is what is mismatched.

Prospective tests: anchor sets outside the sweep, anchor count, and the controlled joint. Two single-joint anchor sets the rule was not derived from, the head and the left wrist, were run on the multi-joint adapter under the protocol's density mixture at $\rho=0.1$ and $\rho=10$, five repeats each (Table 10): strong anisotropy is worth 0.013 FID on head control (paired per-repeat difference

anchors	ρ	FID	Pos. Err. (cm)
left wrist	0.1	0.091 ± 0.009	0.236 ± 0.011
left wrist	10	0.096 ± 0.010	0.200 ± 0.009
head	0.1	$\mathbf{0.072 \pm 0.013}$	0.232 ± 0.016
head	10	0.085 ± 0.013	0.198 ± 0.010

Table 10: Single-joint head and left-wrist control on the multi-joint adapter under the protocol's density mixture, five repeats, $\pm 95\%$ t.

	$\Delta\text{FID} = \text{FID}(\rho{=}0.1) - \text{FID}(\rho{=}10)$ at anchor density			
controlled joint	1	5	25%	100%
pelvis	$-0.001 \pm 0.003\,\cdot$	$-0.012 \pm 0.002\,\downarrow$	$-0.024 \pm 0.002\,\downarrow$	$-0.019 \pm 0.005\,\downarrow$
head	$-0.003 \pm 0.005\,\downarrow$	$-0.007 \pm 0.006\,\downarrow$	$-0.028 \pm 0.004\,\downarrow$	$-0.019 \pm 0.008\,\downarrow$
left wrist	$-0.002 \pm 0.004\,\downarrow$	$+0.014 \pm 0.009\,\uparrow$	$+0.041 \pm 0.005\,\uparrow$	$+0.027 \pm 0.017\,\uparrow$
right wrist	—	$+0.000 \pm 0.011\,\cdot$	$+0.056 \pm 0.022\,\uparrow$	—
left foot	—	$+0.053 \pm 0.012\,\uparrow$	$+0.267 \pm 0.052\,\uparrow$	—
right foot	—	$+0.039 \pm 0.008\,\uparrow$	$+0.175 \pm 0.032\,\uparrow$	—

Table 11: Anchor count versus controlled joint on the multi-joint adapter: paired per-repeat difference in FID between strong ($\rho = 0.1$) and relaxed ($\rho = 10$) anisotropy, three repeats, $\pm 95\%$ t; $\downarrow/\uparrow$ mark cells whose three repeats all favour the strong/relaxed setting, $\cdot$ a tie. Densities 25% and 100% are the protocol's 49- and 196-frame categories; the right wrist and the feet were run at 5 and 25%.

-0.0131 ± 0.0019, the same sign in all five repeats) and the two settings tie on the wrist (-0.0047 ± 0.0067). Fixing the anchor density then separates the anchor count from the joint (Table 11: six joints, densities 1, 5, 25% and 100% of the clip's frames, three paired repeats per cell, test split, cap 196, $K = 8$). With a single anchor ρ is inert for every joint: one row gives the metric no uneven pricing to act on, and the scalar damping absorbs the uniform gain. From five anchors up the effect grows with the count, and the joint sets its sign—pelvis and head gain from strong anisotropy at every multi-anchor density, both wrists and both feet lose—with the same sign in every repeat of 14 of the 15 multi-anchor cells (the right wrist at five anchors ties). The wrist's tie under the density mixture is the average of an inert single-anchor cell and losses at higher densities. Tracking moves by at most 0.09 cm between the two settings in any cell.

Bone length in this representation. HumanML3D represents and evaluates motion as joint *positions*; joint rotations—and the twist about each bone in particular—are not identifiable from positions, and a rigid body is recovered only downstream, by fitting SMPL to those positions Guo et al. (2022); Loper et al. (2015); Bogo et al. (2016), which re-imposes fixed bone lengths. Decoded positions on this benchmark are therefore not bone-length preserving—our outputs and a VQ round-trip of the ground truth alike show the drift. No metric in this protocol scores it and we do not treat it as an evaluation axis; it does not reach the fitted meshes our figures show, where a mesh-fit gate applied uniformly to every method serves visual clarity only, and every quantitative result is computed on the full, unfiltered test split.

Per-iteration probe protocol. Tracing every refinement iteration shows the inherited schedule spends most of its budget after convergence: 97% of samples are already below $10^{-5}\,\text{m}^2$ anchor MSE at Stage-2 entry, and 75% exit Stage 2 worse than their best intermediate iterate. These were collected on 32 train-split clips with training-distribution anchor sampling, tracing every refinement iteration of a dynamic-Stage-1 refinement schedule; the same traces show 95% of each unmask step's gain arriving in a median of 12–23 iterations.

Multi-joint CodeGeo: full anchor capacity. Multi-joint control commands far more simultaneous targets than pelvis control—up to 1,176 joint-frame anchors. Capping the Jacobian to a small subsample of them leaves the omitted anchors unconstrained and the solve drifts on them, so the solver is always given its *full* anchor capacity: every commanded target enters the residual and the metric-regularized dual solve stays well-conditioned. At full capacity (every commanded anchor,

up to 1,176 per sample), with the step-metric anisotropy set to $\rho = 10$ (§4.4), the same ≤ 8 analytic steps reach 0.25 ± 0.011 cm at Top-3 0.797 (five repeats), below the MaskControl-protocol row's 0.90 cm (Table 1) and MaskControl's published 0.72 cm Pinyoanuntapong et al. (2025), with no Stage-1 warm-up.

Anisotropy at full capacity. Full anchor capacity leaves no commanded anchor unconstrained, but it also widens the anchor set across body parts, which is exactly the regime in which the step metric's anisotropy has to be relaxed (§4.4). At the default anisotropy ($\rho = 0.1$) the multi-joint solve reaches only 0.326 ± 0.017 cm at FID 0.118 ± 0.015; at $\rho = 10$ it reaches 0.25 ± 0.011 cm at FID 0.072 ± 0.009, far better on tracking than the MaskControl-protocol row (0.90 cm), at a higher FID than that row's 0.053, and better than MaskControl's published 0.72 cm and 0.083 on both axes; FID stays far below OmniControl's 0.310 and TLControl's 0.256.

B Evaluation Protocol Details

Control targets and anchor sampling. Targets are taken from the ground-truth motion paired with each test prompt. Per sample, a density category is drawn uniformly from $\{1, 2, 5, 49, 196\}$: categories 1/2/5 pin that exact number of frames, while 49 and 196 pin 25% and 100% of the clip's frames respectively ($\lfloor L \cdot c/196 \rfloor$ for clip length L). Anchor frames are then drawn uniformly without replacement. The controlled joint is the pelvis in the single-joint setting; in the multi-joint setting each sample draws up to six joints from {pelvis, head, left/right foot, left/right wrist}. This is MaskControl's `density -1` evaluation procedure, reproduced from their released code, and it is applied identically to every configuration we run.

Aggregation. Following the benchmark convention, we report single aggregate metrics in the main text; R-precision is ranked within batches of 32. For the batched-throughput measurement (Table 16) generation is grouped 16 batches (512 samples) at a time—9 full groups covering 144 of the split's 145 batches, the final partial batch being dropped from that timing only; the same grouping is used for every method. Features are extracted batch by batch over the full 145-batch split; FID is computed from the pooled features, Top-3 is ranked within each batch of 32, and position error is averaged over all controlled joint–frames.

Metrics. FID and R-precision (Top-3) are computed with the HumanML3D evaluator's motion and text feature extractors Guo et al. (2022): FID measures distributional distance to the real motion set and Top-3 the retrieval rank of the paired caption, so the pair captures generation quality and text faithfulness independently of control. *Avg. Pos. Err.* is the mean Euclidean distance between each controlled joint–frame and its commanded target, the quantity the anchor constraints target. *Traj. E* and *Loc. E* are the percentages of sequences and of joint–frames exceeding a 50 cm error (thresholded control metrics introduced by OmniControl Xie et al. (2024)), reported because a mean can hide a heavy tail. *Skate* is the foot-skating ratio (originating in GMD Karunratanakul et al. (2023)) as implemented by Pinyoanuntapong et al. (2025), included because trajectory control can satisfy anchors by sliding contacts. All are computed with the benchmark's released code.

Computing infrastructure. Training and evaluation ran on NVIDIA H200 (141 GB), A100 (80 GB), and H100 (80 GB) GPUs under Linux (RHEL 8), Python 3.10, PyTorch 2.8 with CUDA 12.6. Each figure reporting latency names the GPU it was measured on; latencies are never compared across device types.

Code availability. The training, evaluation, and analysis code will be organized into an open-source repository and released upon acceptance.

Batch-1 latency measurement. All single-sample latencies (§Experiments; Table 15) are measured on an H200 under one timing convention: batch 1, `torch.cuda.synchronize()` at both timer boundaries, and the median over the same 64 test samples, each method at its released inference configuration. The rows of Table 15 come from a single unified run on one card. The speedup confidence interval is a per-clip paired bootstrap. The iterative MaskControl-protocol rows run the released two-stage refinement schedule unchanged. Logs and the analysis script are included with the code release.

Inference FLOPs. Table 12 reports analytical dense-matmul forward-pass FLOPs ($=2\times$MACs) for the generation backbone at the checkpoint configuration ($d=384$, ff $=1536$, $L=20$ layers, 8 heads; K/V injection at every layer with 49 appended control memories; gated text cross-attention at interval 2, i.e. 10 blocks; a 4-layer CLIP text adapter per forward); a released script (`scripts/flops_analysis.py`) reproduces the count. It is a matmul estimate and excludes softmax, normalization, activations, the CLIP text encoder, VQ decoding, the trajectory encoder, and CodeGeo. The T-Concat backbone ($S=294$) costs 27.8 GFLOPs per forward; under classifier-free guidance (two passes) over $T=10$ unmask steps ($\times 20$), 557 GFLOPs per generated sample. The C-Concat layout ($S=49$) costs 5.8 GFLOPs per forward. Total backbone compute is therefore $\approx 4.8\times$ that of C-Concat, because the linear projection and feed-forward terms scale as S and dominate at these dimensions. The K/V injection (low-rank projectors plus attention over the 49 appended memories) adds 0.59 GFLOPs per forward, 2.1% of the backbone: the interface is compute-light as well as parameter-thin. CodeGeo runs ≤ 8 damped Gauss–Newton iterations; its per-sample cost is the measured batch-1 latency of Table 15, an order of magnitude below the inherited iterative refinement.

Backbone / layout	S	GFLOPs/fwd	GFLOPs/sample
T-Concat (ours)	294	27.8	557
C-Concat	49	5.8	116
K/V injection (added)	—	+0.59 (2.1%)	—

Table 12: Analytical dense-matmul backbone inference FLOPs ($2\times$MACs) at the checkpoint configuration; only the token count S differs between layouts. Per-sample assumes classifier-free guidance (2 passes) over $T=10$ unmask steps ($\times 20$). The K/V injection row is the additive adapter overhead (projectors + memory attention). Excludes softmax, norms, activations, CLIP, VQ decoding, control encoder, and CodeGeo. Reproduced by `scripts/flops_analysis.py`.

Model selection. Checkpoints are selected by each training run's built-in periodic evaluation criterion and frozen before any controlled evaluation in this paper is run: the single-joint pelvis adapter by position error, the multi-joint adapter by Top-3, and the base backbones by evaluator FID (matching the block labels of Table 17).

Per-backbone unconditional optimum (Base rows in Table 14). The two unconditional *Base* rows report each backbone at the CFG that attains the best joint FID/Top-3 operating point, identified by an offline CFG sweep (step 0.25) at a unified sampling-step setting $T = 18$; metrics are reported at the selected CFG (T-Concat $CFG = 3.75$; C-Concat $CFG = 3.25$). Controlled-generation rows fix CFG/T per the protocol declared in §4.1.

Prior-work numbers. The upper blocks of Table 1 retain three prior-work rows (OmniControl, TLControl, MaskControl), printed as published by Pinyoanuntapong et al. (2025); further published baselines (GMD, MotionLCM, and the "MaskControl w/o Logits Opt." diagnostic of their Table 5) are in Pinyoanuntapong et al. (2025). Those rows all follow the same control-evaluation protocol—OmniControl's metric suite and density sampling, which MaskControl's released code implements—so the cross-block comparison is protocol-matched; what differs is re-execution (implementation, hardware, and seed), which we do not repeat for the prior-work rows.

Refinement-only baseline. The refinement-only row in Table 17 is produced by removing the K/V adapter entirely from the inference graph (no control K/V pseudo-tokens injected at any layer) and running the Stage-1 logit AdamW and Stage-2 token-embedding AdamW schedules unchanged on the bare frozen T-Concat backbone. Targets and the masked control feature $\mathbf{c}_s$ enter only through $\mathcal{L}_{\text{traj}}$; backbone parameters remain frozen throughout.

C Additional Results and Qualitative Examples

The anonymous supplemental video collects side-by-side clips of the demos rendered in the figures of the main text and this appendix—each clip built from the same solved motion as its figure panel—



Figure 6: **Same-trajectory comparison, ours vs. MaskControl.** Both methods generate motion for the identical pelvis control path and prompt, so path following is judged on the same targets. Our panels (KV-Control + CodeGeo) use the analytic full-capacity operating point (≤ 8 damped Gauss–Newton steps); MaskControl panels use the public released model (RVQ+MoMask substrate) under its own iterative two-stage refinement. Meshes for every panel are built by the same fixed-identity SMPLify fit (GMM pose prior) as the main-text demos, and rendered in the same style: seven poses overlaid along each motion on a metric checker floor, with *red spheres* the sampled targets at their true pelvis height and the *orange ribbon* the commanded path projected to the ground; the bodies themselves are the produced motion, so adherence is read off how they sit on the ribbon. Both methods stay sub-centimetre on all four of these dense-control clips—this figure is a qualitative check that the two mechanisms behave comparably on the same saturated signal; the sparse designed-trajectory setting of Fig. 7 is where they separate. Quantitative results under the MaskControl evaluation protocol are in the main text (Table 1).

including the same-trajectory and designed-trajectory comparisons and a before/after clip of the correction (Figure 3).

C.1 Mechanism and Layout Ablation

The backbone-layout ablation isolates how PartVQ codebooks are presented to the transformer. *T-Concat* unpacks $Q = 6$ codebooks along the sequence axis ($S = 294$); *C-Concat* channel-stacks them into one 768-channel frame token ($S = 49$). Both backbones share the trunk of §4.1, are trained from scratch under matched protocols, then frozen before attaching an identical KV-Control adapter.

Findings. Table 14 isolates the layout as the operative factor by ruling out the two alternative readings of the K/V-side gap. *Backbone quality*: the two trunks are comparable in unconditional generation (C-Concat even has slightly lower FID), so the post-adapter gap is not attributable to trunk quality. *Refinement protocol*: under the shared refinement schedule C-Concat's position error is 2.2 times as large as T-Concat's (1.10 vs. 0.51 cm), with Top-3 within 0.01 across the two layouts; the gap reproduces under an independent retraining (§C.4, Table 19).

Under CodeGeo. The gap does not depend on the refiner. With the same analytic solve on both layouts (pelvis control, $\rho = 0.1$, cap 196, $K = 8$, the per-sample solver on both, five paired repeats; Table 13), C-Concat tracks $1.46\times$ worse than T-Concat (0.34 vs. 0.24 cm) at 0.036 ± 0.004 higher FID, worse in every repeat, and the T-Concat control reproduces the batched-solver row of Table 1.

Why layout matters. T-Concat's advantage stems from a representational property, and not from a larger control branch: both layouts append the same number of control key/value entries at every

Layout	FID ↓	Top-3 ↑	Skate ↓	Pos. Err. (cm) ↓
T-Concat	0.063 ± 0.007	0.797 ± 0.010	0.0500 ± 0.0009	0.24 ± 0.01
C-Concat	0.099 ± 0.003	0.785 ± 0.007	0.0563 ± 0.0009	0.34 ± 0.05

Table 13: The layout ablation under CodeGeo: the same adapter architecture on the two layouts, each with its own frozen backbone, pelvis control, five repeats ($\pm\, 95\%\ t$), identical solver settings, per-sample generation over every test clip.

Backbone	Cfg.	FID	Top-3	Avg. Pos. Err. (cm)
Unconditional generation (no control adapter)				
T-Concat	Base	0.081	0.797	–
C-Concat	Base	0.078	0.790	–
KV-Control adapter (identical config.), root-trajectory control				
T-Concat (main)	**MC-protocol**	**0.064**	**0.801**	**0.51**
C-Concat	MC-protocol	0.093	0.795	1.10

Table 14: Backbone-layout and mechanism ablations on the same 20-layer trunk; controlled rows under MaskControl's released evaluation setting (§4.1), unconditional Base rows at each backbone's own CFG/T optimum (Appendix B). KV-Control is evaluated on T-Concat and C-Concat layouts: same trunk architecture, same adapter, only the codebook layout changes. Bold indicates the main layout; both rows use the shared MaskControl-protocol refinement (MC = MaskControl).

Method	Top-3 ↑	FID ↓	Skate ↓	Pos. Err. (cm) ↓	Lat. (s)
Iterative refinement	0.801	0.064	0.0444	0.51 ± 0.01	4.48
Iterative, early-stopped	0.803	0.067	0.044	0.46	3.59
CodeGeo	0.802	0.067	0.0477	$\mathbf{0.24 \pm 0.02}$	**0.21**

Table 15: CodeGeo vs. iterative refinement on pelvis control, five repeats under MaskControl's released evaluation setting. *Iterative, early-stopped* is a stronger iterative baseline—plateau-based early stopping (checked every 10 iterations, stopping once the anchor error improves by less than 1% over the window) with per-sample best-iterate return, which discards the fixed schedule's wasted iterations (§A.1 notes 97% of samples enter Stage-2 already converged, measured on 32 train-split clips)—so the comparison is not against an over-provisioned schedule. CodeGeo still nearly halves its position error at a fraction of its latency. Latency is the batch-1 median over 64 test samples, all rows measured on the same H200. $\pm$ are 95% t-intervals; the early-stopped row's accuracy is a two-repeat probe.

layer through the same adapter, so the injection side is identical. What differs is the read side—each (frame, codebook) pair occupies a distinct attention slot, so a low-rank K/V residual at a query position can selectively modulate one anatomical sub-token. C-Concat collapses all six codebooks into one frame-shared state (49 query slots rather than 294), forcing any K/V residual to steer six entangled codebooks through one latent. This makes *representation-layout co-design* central to parameter-efficient controllable text-to-motion adapters.

C.2 CodeGeo: Full Results Table

Every qualitative figure in this appendix renders on a metric checker floor with a fixed camera, so foot placement and sliding are directly inspectable against the grid (Figs. 6, 7, 8); Table 15 reports the quantitative comparison against the iterative refinement.

Discarding the fixed schedule's wasted iterations recovers little: the early-stopped iterative baseline still trails CodeGeo on position error (0.46 vs. 0.24 cm) at the same FID, and remains $\sim 17\times$ slower per sample (3.59 vs. 0.21 s). A paired bootstrap over clips puts CodeGeo's speed-up against the protocol schedule at $18\times$–$22\times$. The public baselines timed in Table 1 run loops of a different kind: 1,000 diffusion denoising steps for OmniControl and an L-BFGS cap of 1,000 iterations for TLControl.

Stage	A100 (s/group)	H200 (s/group)
Generation only (no refinement)	22.95	10.89
Iterative refinement	101.6	63.42
CodeGeo, per-sample solves	–	194.3
CodeGeo, batched solves	**32.51**	**15.68**
Throughput vs. iterative	3.1×	4.0×
Control overhead, net of generation	12.2%	9.1%

Table 16: Throughput over nine 512-sample groups of the test split (final partial batch excluded), seconds per group. Single-joint pelvis control under MaskControl's released evaluation setting; identical checkpoint, protocol, and seed throughout, with the CUDA device synchronized at both timer boundaries. A100 = 80 GB PCIe with an Intel Xeon Gold 6336Y host; H200 = NVL with an AMD EPYC 9275F host. *Generation only* disables control refinement entirely and is the cost both refinement stages share. The per-sample-solves row is a per-group mean; it exceeds the batch-1 *median* of Table 15 because the anchor-density mix is right-skewed. – = not measured.

Throughput under cross-sample batching. Table 15 measures one sample at a time, which is where the iterative loop is most exposed. Batch service is the opposite regime, so we also time whole groups of 512 samples (16 batches of 32; nine groups cover the test split). CodeGeo's Gauss–Newton system is per-sample, but the samples in a group are independent, so the solves batch: one padded system replaces 512 sequential ones. Table 16 reports both an A100 and an H200; all cells share a checkpoint, protocol, seed, code tree, and timing convention, and differ only in the refinement stage.

Two readings matter. Batched CodeGeo sustains 3.1× the throughput of the iterative refinement on the A100 and 4.0× on the H200, at roughly half the position error (0.24 vs. 0.51 cm). But a large part of any group-level gain is generic: batching the generation alone, with no solver involved, is already worth ∼8.0× on the H200 (0.17 s at batch 1 vs. 10.89 s per 512-sample group). The *Generation only* row isolates that shared cost, and net of it CodeGeo's control overhead is 12.2% of the iterative stage's on the A100 (9.1% on the H200). That residual, not the raw ratio, is what the analytic step contributes.

The ratio is host-dependent; the table reports both devices. The four configurations sit at different points between dispatch-bound and GPU-resident, so they do not scale together across machines. The serial solver issues one gradient call per Jacobian row and is almost purely dispatch-bound (on the A100 host it runs at 25% device utilization with one core saturated), whereas the iterative stage's Adam loop is the most device-resident of the four and therefore the least sensitive to the host. A weaker host slows the solver more than the baseline, which compresses the ratio rather than inflating it.

C.3 Extended Diagnostics

Extended diagnostics (full table). Table 17 adds matching-score (Match., lower better) and intra-set Diversity (Div., real-data ∼ 9.5) to the two shared-protocol regimes (feed-forward and MaskControl-protocol refinement; our CodeGeo operating points are in Table 15 and the main text's Table 1), plus a refinement-only baseline (frozen T-Concat, no K/V adapter). Stripping the K/V adapter and running the identical refinement schedule drives position error to near zero but collapses FID to ∼ 104 and Top-3 to ∼ 0.12—the adapter, not the refinement loop, supplies the structural conditioning that keeps refinement inside the text-conditioned motion manifold.

Per-joint decomposition (full table). Table 18 reports the multi-joint adapter with the controlled set fixed to one anatomical joint at a time. Aggregate single-joint position error is 0.41 cm (FID 0.086, Top-3 0.795); per-joint position error ranges from 0.31 cm (left wrist) to 0.69 cm (pelvis). FID is highest on the two feet (0.117 ± 0.016 and 0.102 ± 0.008), whose tokens carry the strongest locomotion priors and resist keyframe pinning.

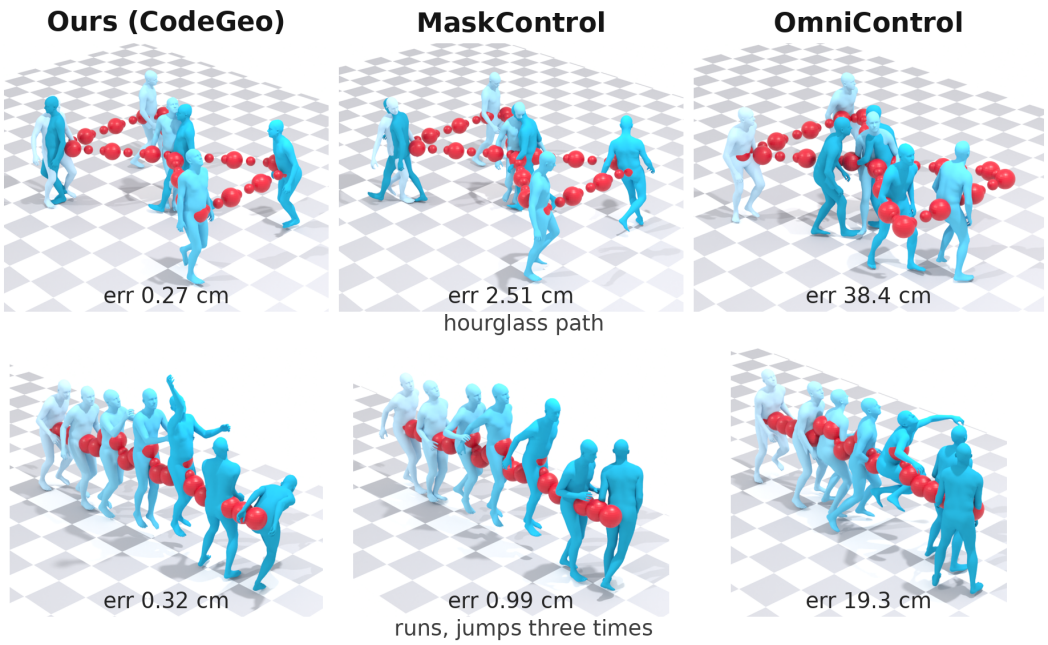


Figure 7: **Designed trajectories, sparse keyframes, semantic prompts.** Red spheres are the 20 commanded pelvis anchors—identical input to all three methods; bodies are each method's output, meshed by the same fixed-identity SMPLify (GMM prior) pipeline. Prompts: "walks along a crossing path shaped like an hourglass" (top), "runs forward and jumps three times" (bottom); the stairs and zigzag demos are reported in Table 20. Each cell is annotated with its measured mean keyframe error. OmniControl's panels are rendered with three mesh-pipeline gates relaxed, applied uniformly to every method (only its cells need them): the control-adherence gate for both its cells (the 38.4/19.3 cm misses *are* the content), and the skeleton-rigidity ($40 \rightarrow 60$ mm) and rigid-body fit ($3.0 \rightarrow 4.0$ cm) gates for the jumps, whose skeleton drifts 55 mm in bone length over the clip so the best fixed-identity mesh fit sits 3.9 cm from its joints; the stairs demo exceeds even the raised fit gate (9.6 cm) and is reported in Table 20 only.

Regime	Avg. Pos. Err. (cm)	FID	Top-3	Match. ↓	Div. →
Single-joint pelvis checkpoint selected by Avg. Pos. Err.					
Feed-forward	29.88	0.076	0.809	2.890	9.566
MaskControl-protocol	**0.51**	**0.064**	**0.801**	**2.939**	**9.481**
Multi-joint checkpoint selected by Top-3					
Feed-forward	40.63	0.074	0.811	2.866	9.566
MaskControl-protocol	**0.90**	**0.053**	**0.802**	**2.934**	**9.465**
Refinement-only baseline: frozen T-Concat, no K/V adapter					
Frozen T-Concat (refined)	0.013	103.86	0.122	10.049	1.377

Table 17: Extended diagnostics under the MaskControl protocol; means unless noted. Regimes: *feed-forward* (no test-time optimization) and *MaskControl-protocol* refinement, as defined in §4.1. Match. is the motion-text matching score of Guo et al. (2022); Div. is intra-set feature variance with real-data Diversity ~ 9.5 ($\rightarrow$: closer to the real value is better). Bold marks the regime our refined rows use, not the better cell: on Top-3 and Match. the feed-forward rows score higher.

C.4 C-Concat Reproduction Note

The C-Concat KV-Control rows of the backbone-layout ablation (Table 14) report an independent retraining of the KV-on-C-Concat adapter, run end-to-end from a freshly trained C-Concat base under the identical recipe and the same five-repeat evaluation. Table 19 places the retrained values next to the original run. FID and Top-3 reproduce to within one interval half-width—each mean sits

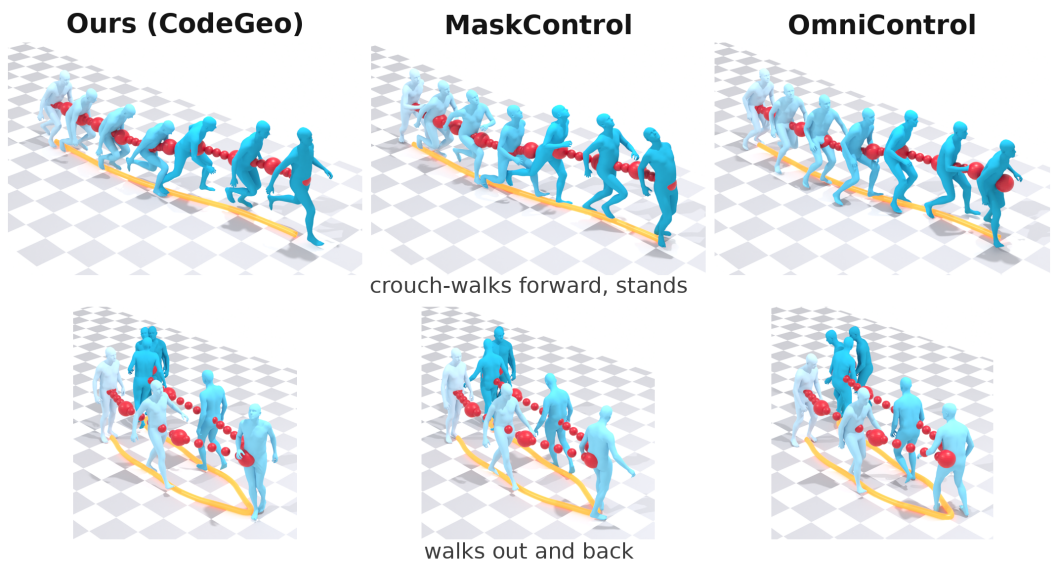


Figure 8: **Three-way qualitative comparison (reference).** Ours (KV-Control + CodeGeo), MaskControl, and OmniControl on the identical dense pelvis path and prompt, in the same style as Fig. 6: seven overlaid poses, *red spheres* the sampled targets, *orange ribbon* the commanded path. The two clips ("a man crouches down while quickly walking forward and then stands"; "a man walks forward, then turns around and walks back") appear in no other figure, and were chosen among the held-out picks where all three methods' skeletons could be meshed. Mean pelvis error: ours 0.42/0.23 cm, MaskControl 0.92/1.08 cm, OmniControl (rerun with its classifier-free guidance verified applied, scale 2.5) 15.1/18.7 cm—visibly short of the crouch clip's path end. Mesh gates were raised uniformly (bone 45 mm, leg 6 cm) to admit the competitors' skeletons: ours pass the default gates (≤ 34 mm bone drift, legs within 2 cm of canonical), while MaskControl's crouch-clip skeleton walks on legs 5.3 cm short and OmniControl drifts up to 31 mm here. OmniControl applies sampling-time guidance with no anchor-solving finish. Rendering is identical for all panels.

Controlled joint	FID	Top-3	Pos. Err. (cm)
Pelvis	0.064	0.791	0.69
Head	0.060	0.797	0.37
Left foot	0.117	0.790	0.40
Right foot	0.102	0.794	0.37
Left wrist	0.080	0.798	0.31
Right wrist	0.092	0.799	0.32
Average (6 single-joint)	0.086	0.795	0.41
Multi-joint (all joints)	0.054	0.799	0.80

Table 18: Per-joint single-joint control on the multi-joint adapter, under a heavier internal iterative schedule (dynamic Stage-1—per-step iteration budget growing across unmask steps—+ Stage-2). Each row holds the controlled set fixed to one anatomical joint at inference; the Average row is the unweighted mean of the six rows above; the all-joints row is measured under the same schedule for direct comparison (it is not a Table 1 row).

at the edge of the other run's five-repeat 95% interval—while position error reproduces +0.09 cm higher (1.01 $\rightarrow$ 1.10 cm under the shared refinement protocol). Every comparison retains its sign, and all conclusions hold: C-Concat K/V tracks at near-centimeter accuracy but trails T-Concat, consistent with the layout argument of the main text.

Run	FID ↓	Top-3 ↑	Pos. Err. (cm) ↓
Retrained	0.093 ± 0.011	0.795 ± 0.011	1.10 ± 0.01
Original	0.082 ± 0.011	0.801 ± 0.006	1.01 ± 0.02

Table 19: C-Concat KV-Control reproduction, both rows under the shared MC-protocol refinement: an independent end-to-end retraining (used in the ablation table) versus the original run, at the same training budget, five repeats ($\pm\,95\%$ t-CI), under MaskControl's released evaluation setting. FID/Top-3 agree within CI; position error reproduces $+0.09$ cm; conclusions unchanged.

C.5 Same-Trajectory Comparison with MaskControl

Figure 6 places our method (KV-Control + CodeGeo), at its analytic full-capacity CodeGeo operating point ($\leq$ 8 damped Gauss–Newton steps), and MaskControl, under its released iterative two-stage refinement, on the identical pelvis control path and prompt, so path adherence is judged on the same targets.

C.6 Designed Trajectories with Semantic Prompts

The comparisons above use ordinary HumanML3D test clips, where dense pelvis control is nearly saturated: MaskControl and ours are both sub-centimetre. The paper's demo regime is harder: designed trajectories, 20 sparse pelvis keyframes over 196 frames, and prompts whose semantics must coexist with the geometry—the triple-jump path dictates *exactly three* pelvis-height peaks, and the hourglass path forces a self-crossing walk. Figure 7 feeds all three methods the identical sparse signal and prompt (the same target/mask arrays our published solves consumed; MaskControl's mask conditioning and OmniControl's zero-row hint encoding both support sparse keyframes natively, and both were run at their released settings, OmniControl with its classifier-free guidance verified applied). Table 20 reports keyframe error over the 20 commanded anchors.

demo	Ours	MaskControl	OmniControl
hourglass	$\mathbf{0.27}/0.85$	$2.51/5.52$	$38.4/147.7$
stairs	$\mathbf{0.29}/0.76$	$1.25/2.21$	$21.9/44.1$
jumps×3	$\mathbf{0.32}/0.74$	$0.99/2.71$	$19.3/56.4$
zigzag	$\mathbf{1.25}/3.88$	$4.31/13.25$	$37.6/218.9$

Table 20: Keyframe error, mean/max cm over the 20 commanded anchors of each designed demo; identical sparse control signal and prompt fed to all three methods.

Under sparse designed control the ordering is no longer close: our analytic solve is 3–9× more accurate than MaskControl's 1,600-iteration refinement on every demo, and OmniControl's guidance does not find these paths at all, and the accuracy comes at a fraction of the test-time cost: under the shared protocol of the main text's Table 1, one sample takes 0.21 s for our analytic solve against 3.83 s for MaskControl's released pipeline and 24.61 s for OmniControl's guided diffusion. The zigzag row is reported in Table 20 only: on that sharp-cornered path the meshes do not clear the figure pipeline's fit gate.

C.7 Three-Way Qualitative Comparison with OmniControl

Figure 6 compares our method and MaskControl, each at its own operating point (our analytic CodeGeo correction vs. its released iterative refinement). Figure 8 adds the published OmniControl model Xie et al. (2024) as a sampling-time-guidance reference on two further test clips that appear in no other figure, and notes the caveats that accompany the comparison.