SAMoR: Motion Modelling for Articulated Objects of Any Skeleton and Topology

Yuhao Zhang
Imperial College London

Gerard Pons-Moll
University of Tübingen, Tübingen AI Center

Tolga Birdal
Imperial College London

Abstract

Modeling motion for articulated objects of arbitrary skeleton topology remains difficult: existing motion generators target a fixed human skeleton, and prior adaptations either fail to share a vocabulary across rigs or discard motion detail through global pooling. Our key observation is that while joint-level motion does not correspond cleanly across species, motion of functional joint groups does—a human arm, a wolf foreleg, and a bird wing share semantic motion structure despite differences in joint count and connectivity, a correspondence that joint names (e.g., "forearm", "wing_L1") partially expose even when topology does not. We introduce SAMoR (Skeleton-Aware Motion Representation for Articulated Objects), a cross-topology motion representation that encodes each motion segment as a small fixed number ($K{=}8$) of part tokens shared across skeletons with different joint counts and connectivity. SAMoR takes three input signals—per-joint motion features, kinematic graph structure, and joint-name embeddings—and processes them with a graph-transformer encoder, then compresses the resulting heterogeneous per-joint features into part-level tokens via cross-attention pooling and residual vector quantization, yielding a discrete motion codebook shared across rigs. To prevent the part queries from collapsing into redundant global representations, we introduce a topology-agnostic attention supervision loss, combined with random joint-name dropout to prevent over-reliance on text labels; together these encourage the part tokens to cluster joints into functional groups from names, structure, and motion jointly. We curate a unified heterogeneous motion corpus from HumanML3D, Truebones Zoo, and animated Objaverse-XL assets, and evaluate SAMoR on held-out characters with unseen skeletons. The resulting representation supports accurate reconstruction and cross-topology motion transfer, and further enables text-conditioned generation and localized part-wise editing through a MaskGIT token generator. SAMoR reaches $2.75{\times}10^{-2}$ normalized MPJPE on cross-topology reconstruction—$5.8\times$ below the strongest adapted variable-J tokenizer baseline—and remains competitive with fixed-skeleton specialists on HumanML3D for both VQ-VAE reconstruction and text-to-motion generation. Our code and extended results can be found at `http://circle-group.github.io/research/SaMoR`.

1 Introduction

Modeling motion for articulated objects of arbitrary skeleton topology (humans, quadrupeds, birds, insects, mechanical rigs, and stylized 3D assets) remains a difficult problem. Recent text-to-motion generators [27, 11] achieve impressive fidelity on HumanML3D [13] and related human benchmarks [2], but their motion representations are tied to a fixed SMPL human skeleton [24] with fixed joint count, ordering, and connectivity. Extensions to animal categories have been proposed [18, 44, 40, 35, 22], but they likewise assume a predefined skeleton topology one way or the other. Cross-topology motion generators have recently surfaced: SinMDM [30] trains a separate model from a single motion sequence, while AnyTop [9] uses a skeleton-conditioned diffusion model that operates at the per-joint level. These methods demonstrate important progress, but they do not provide a fixed-size discrete

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

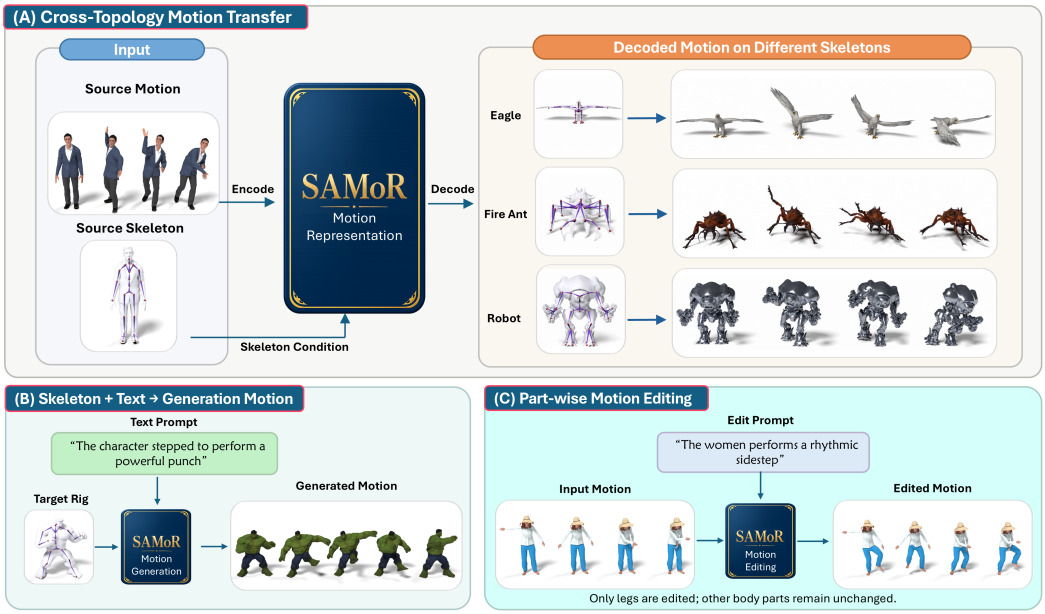


Figure 1: **SAMoR enables three downstream tasks from a single cross-topology motion representation.** **(A)** Cross-topology motion transfer by swapping the skeleton conditioning at decode time (eagle, fire ant, robot). **(B)** Text-to-motion generation conditioned on a target rig. **(C)** Part-wise editing: replacing selected part tokens edits only the selected functional groups (e.g., legs) while preserving the rest. Extended generation, editing, and mesh-animation results are on the project page.

motion vocabulary shared across rigs. As a result, directly reusing a learned motion codebook for heterogeneous source-to-target motion transfer or token-based generation remains difficult across characters with different joint counts and connectivity.

Arguing that the appropriate abstraction level for cross-topology motion is neither the whole body nor individual joints, but functional groups of joints, we introduce **SAMoR**, a **s**keleton-**a**ware cross-topology **m**otion **r**epresentation for articulated objects. SAMoR uses a graph-transformer encoder to model skeleton structure, cross-attention pooling to compress heterogeneous per-joint motion features into $K = 8$ part tokens, and residual vector quantization to form a shared discrete motion codebook [38]. A target-skeleton-conditioned decoder reconstructs motion on the same or a different rig, enabling cross-topology motion transfer directly at decode time. The key challenge is query collapse: without explicit guidance, all K queries attend broadly and produce redundant representations, reducing $K{=}8$ to the effective equivalent of $K{=}1$; we address this with an attention supervision loss that anchors each query toward a distinct functional joint group. We train and evaluate SAMoR on a unified heterogeneous corpus curated from HumanML3D [13], Truebones Zoo [37], and animated Objaverse-XL [7] assets, with held-out characters used to test generalization to unseen skeletons. Importantly, when instantiated in a HumanML3D-compatible setting, SAMoR remains competitive with fixed-skeleton specialists on standard HumanML3D VQ-VAE reconstruction and text-to-motion benchmarks. Yet unlike these specialists, the same part-token architecture extends naturally to heterogeneous skeletons in our cross-topology setting, without requiring per-character models.

Overall, our contributions are:

- **Graph-Transformer-based Motion Tokenization:** A cross-topology motion VQ-VAE that compresses motion on skeletons of different topologies into $K{=}8$ shared part tokens via graph-transformer encoding and cross-attention pooling, yielding a single discrete motion vocabulary across heterogeneous skeletons with varying joint counts and connectivity.
- **Attention Supervision for Query Specialisation:** An attention supervision loss that breaks query symmetry by seeding each query toward a distinct functional joint group, preventing collapse. Combined with joint-name dropout, this generalises to unseen skeletons with missing or non-canonical joint names.
- **Unified Heterogeneous Motion Corpus:** We curate a cross-topology corpus of $\sim$71k characters and $\sim$100k motions from 1 SMPL skeleton (HumanML3D), 65 animal species (Truebones Zoo), and $\sim$71k animated Objaverse-XL characters, with automated canonical joint-matching, character-

level train/val splits, functional-group annotations across diverse morphologies, and per-character motion captions generated via video rendering and vision-language captioning [10].
- **Downstream Animation Tasks:** We show that the shared part-token space supports cross-topology motion transfer, text-conditioned generation via MaskGIT [5], and localized part-wise token editing.

Extensive evaluations show that SAMoR substantially outperforms both single-token global representations ($K = 1$) and per-joint tokenizations ($K = J$), validating intermediate-granularity part tokenization as an effective representation for cross-topology motion modeling. We will make our implementation publicly available.

2 Related Work

Text-to-Motion Generation. Beyond unconditional pose [26, 15] and motion [32, 45] generation, text-driven human motion generation has advanced rapidly on benchmarks such as HumanML3D [13] and KIT-ML [29]. Existing methods include diffusion-based generators [36, 6], autoregressive discrete-token models [47, 19], and masked-token generators [12, 28, 41, 46, 16], which achieve strong motion quality, but their motion representations are tied to a fixed human skeleton, typically SMPL [24] with a fixed joint ordering and connectivity. In contrast, SAMoR studies the orthogonal problem of learning a shared discrete motion representation across heterogeneous skeleton topologies.

Cross-Topology Motion Learning. Cross-skeleton retargeting [1] and cross-morphology alignment [21] enable topology-flexible motion transfer on specific source–target pairs but do not yield a shared representation reusable across arbitrary rigs. SAME [20] learns a per-frame motion embedding shared across human-like skeletons and decodes joint rotations for a target skeleton given in T-pose. Part-based articulated-object methods such as LEPARD [23] study part discovery for 3D articulated shapes rather than motion generation. On the generative side, SinMDM [30] trains a dedicated model per character. AnyTop [9] handles heterogeneous skeletons at the per-joint level ($K{=}J$), preventing vocabulary sharing across different joint counts. Concurrent with our work, NECromancer [42] adopts a single-token ($K{=}1$) formulation. SAMoR takes an intermediate approach: $K{=}8$ learned tokens via cross-attention pooling yield a shared codebook that neither $K{=}1$ nor $K{=}J$ supports natively.

Part-Based Motion Representations. Part-aware methods for human motion synthesis include per-part tokenizers [50], part-level attention [48], text-to-part conditioning [39], and compositional generation [3]. All assume a fixed human skeleton with hard-coded part definitions. SAMoR instead learns K functional clusters of joints that generalise across heterogeneous skeletons without requiring hand-defined per-skeleton part assignments at inference time.

Automatic Rigging and Skinning. Automatic rigging converts raw 3D meshes into animation-ready characters. Classical approaches such as Pinocchio [4] embed a predefined skeleton template via geometric fitting. Recent learned methods predict skeleton and skinning weights directly from mesh geometry: RigNet [43] uses a graph neural network, while MagicArticulate [34], AnyMate [8], and Puppeteer [33] extend this to end-to-end skeleton and skinning prediction for diverse 3D objects. SAMoR is complementary to these systems: given a rigged skeleton from any such frontend, SAMoR provides the motion generation and transfer layer.

3 Method

Overview. SAMoR is a cross-topology motion representation: each motion chunk on an arbitrary skeleton is encoded into a discrete token sequence factorized over $K = 8$ part-level tokens at every downsampled temporal position (Fig. 2). A skeleton-aware part VQ-VAE pools per-joint features into K part queries via cross-attention and quantizes them with residual VQ. A target-skeleton-conditioned decoder expands the part tokens back to any rig, enabling reconstruction and cross-topology transfer.

Topology representation. We represent a skeleton as a *rooted kinematic tree* with J joints, parent map $\mathrm{pa}(j)$, and a per-joint pose anchor $\{\mathbf{a}_j \in \mathbb{R}^3\}_{j=0}^{J-1}$ describing the target joint geometry used for conditioning. Rooted kinematic trees cover virtually all character animation rigs; the graph structural biases (kinematic-tree hop distance, chain membership, edge type) are defined for tree structures, though the transformer architecture itself does not assume a tree. SAMoR assumes a single rooted tree with a valid rig, and mesh animation additionally requires valid skinning weights. Cyclic graphs, disconnected or multi-root skeletons, and severely corrupted rigs are outside our scope.

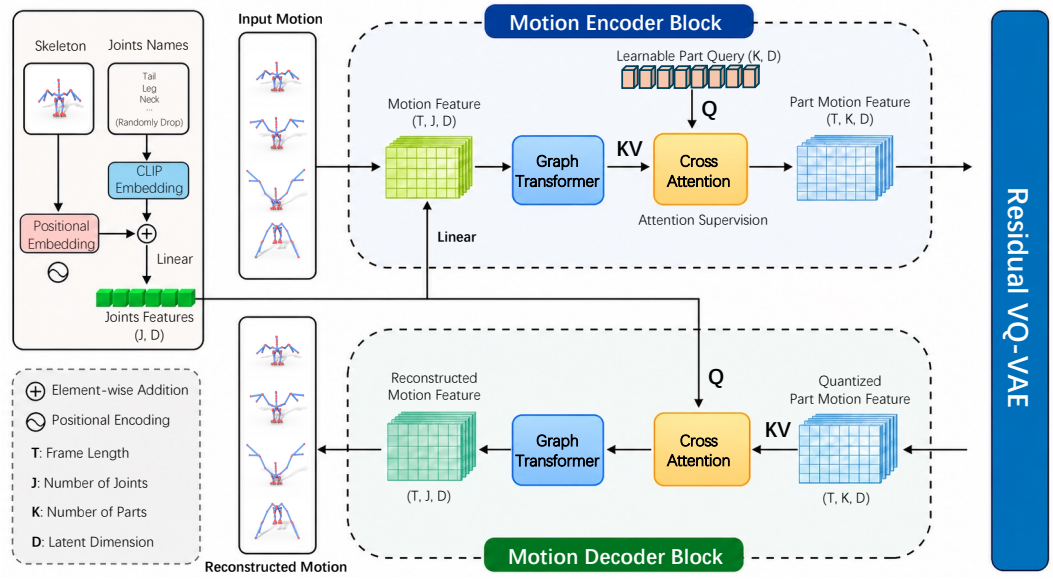


Figure 2: **SAMoR architecture.** Per-joint motion features (T, J, D) are fused with skeleton conditioning (pose anchor, joint-name embeddings, and graph biases) by a graph transformer. $K = 8$ part queries pool the joints via attention-supervised cross-attention into part tokens (T, K, D), which are quantized by residual VQ. The decoder mirrors the encoder with inverse cross-attention conditioned on the target skeleton, enabling cross-topology motion transfer at decode time.

In the cross-topology setting, $\mathbf{a}_j$ is the first-frame joint location; in the HumanML3D-compatible setting, where a canonical SMPL rest pose is available, it is the rest-pose joint location. A motion chunk of length T is represented as $\mathbf{X} \in \mathbb{R}^{T \times J \times d}$, where $d = 12$ in the full-feature setting (position, velocity, and 6D rotation per joint) and $d = 6$ when rotation is omitted. SAMoR encodes this chunk into a discrete token tensor $\mathbf{Z} \in \{1, \ldots, V\}^{(T/4) \times K \times R}$, with $T/4$ temporal positions after $4\times$ downsampling, $K = 8$ part queries, and R residual quantization levels from a shared codebook of size V.

3.1 Skeleton-Aware Part VQ-VAE

Topology-agnostic motion input. For a joint j at frame t, the full-feature representation is $\mathbf{x}_j^t = [\mathbf{p}_j^t, \mathbf{v}_j^t, \mathbf{R}_j^t] \in \mathbb{R}^{12}$, where $\mathbf{p}_j^t$ and $\mathbf{v}_j^t$ are the joint position and velocity, and $\mathbf{R}_j^t$ is the local joint rotation in the 6D continuous representation [49]. In the cross-topology setting, local rotations are defined relative to rig-specific rest poses and are not directly comparable across arbitrary skeletons. We therefore use root-normalized joint positions and velocities, $\mathbf{x}_j^t = [\mathbf{p}_j^t, \mathbf{v}_j^t] \in \mathbb{R}^6$, as the shared cross-rig signal. Global root translation is excluded because heterogeneous assets have inconsistent reference frames and rest-pose definitions; for HumanML3D, where the standard evaluator expects the 263-D representation, root channels are modelled by a separate lightweight VQ branch (Sec. 4.2).

Skeleton conditioning with pose anchors. We condition the encoder and decoder on the target skeleton through three signals. First, a per-joint *pose-anchor embedding* maps anchor locations $\mathbf{a}_j$ to the model dimension through an MLP and re-injects them at every graph-transformer layer. For cross-topology settings we use the first frame as the anchor (more reliable than asset-provided rest poses for heterogeneous sources such as OXL); for HumanML3D we use the canonical SMPL rest-pose anchor. The pose anchor is a static skeleton-geometry signal only: at transfer or generation time it can be any available posed frame or reference pose of the target rig and is not required to correspond to the source or target motion. Second, we add CLIP [31] embeddings of raw joint names, projected to the model dimension. Third, we inject graph structure by adding learned biases to spatial attention logits: kinematic-tree hop distance, same-chain indicator, and edge type.

Graph-transformer encoder. The encoder is a stack of L transformer blocks, each alternating temporal self-attention along T for each joint with spatial self-attention along J for each frame. Spatial attention uses the graph structural biases above; we refer to these blocks as *graph-transformer blocks* throughout. A strided 1D convolution downsamples the temporal axis by $4\times$, producing $\mathbf{H} \in \mathbb{R}^{B \times (T/4) \times J \times D}$. Full architectural details and hyperparameters are in Sec. A.

Part pooling: joints to part tokens. K learnable query vectors compress the J encoded joints into K fixed-size part tokens via cross-attention, followed by a residual connection, layer normalization, and a $K \times K$ self-attention layer for part-to-part communication. This operation is agnostic to skeleton identity: J is variable and the queries impose no constraints on which joints are present—the same architecture processes a 5-joint prop, a 22-joint humanoid, and a 96-joint creature without modification.

Without supervision, the K queries have no incentive to specialize and collapse toward near-identical attention distributions, making $K{=}8$ behave no better than $K{=}1$ in practice (Sec. 3.2). To break this symmetry, we supervise the cross-attention maps during training using canonical functional-group labels $y_j \in \{0, \ldots, K{-}1\} \cup \{-1\}$ derived automatically from joint names. Crucially, joints that cannot be matched to any label receive $y_j{=}{-}1$ and contribute *zero* terms to the supervision loss; they are still encoded and routed freely by the learned attention. For skeletons with no canonical joint names at all (mechanical rigs, procedural assets), every joint has $y_j{=}{-}1$ and the K slots become purely unsupervised functional buckets, driven by kinematic structure and motion patterns alone.

We set $K{=}8$ to identify eight functional clusters that, as abstract skeleton semantics, encapsulate the major motion-producing structures across the diverse topologies in our training corpus. For readability, we label these clusters as *lower torso*, *upper torso*, *head/neck*, *left arm* (forelimb/wing), *right arm*, *left leg* (hindlimb), *right leg*, and *tail* — though these captions are interpretive handles rather than strict anatomical requirements: a cluster labelled 'arm' may capture insect appendages, mechanical links, or wing joints depending on the input skeleton. The attention map $\boldsymbol{\alpha} \in [0, 1]^{K \times J}$ is the target of the supervision loss in Sec. 3.2. Inference-time part-assignment visualisations for diverse topologies, including non-biological rigs, are in Sec. J.

Quantization, decoder, and training objective. Each part token is quantized by an R-level residual VQ module with a shared codebook; quantize-dropout randomly drops deeper RVQ levels during training to improve codebook utilization. The decoder mirrors the encoder: it expands the K quantized part tokens back to J target joints through inverse cross-attention, where target-joint queries are conditioned on the target skeleton's pose anchor, joint-name embeddings, and graph biases. Since the decoder is conditioned on the *target* skeleton, the same source tokens can be decoded on a different rig for cross-topology transfer without retraining. The VQ-VAE is trained with smooth-ℓ_1 position and velocity reconstruction losses, a VQ commitment loss, a bone-length consistency loss (squared parent-to-child distance error, ablated in Table 3), and the attention supervision loss $\mathcal{L}_{\text{attn}}$ (Sec. 3.2). A rotation reconstruction loss is added in the HumanML3D-compatible setting.

3.2 Attention Supervision for Query Specialisation

Part-query collapse in naive training. Given per-frame joint features $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_J) \in \mathbb{R}^{d \times J}$ and K learnable part queries $\{\mathbf{q}_k\}_{k=1}^{K} \subset \mathbb{R}^d$, the part-pool cross-attention computes each part token:

$$\mathbf{p}_k = \mathrm{LN}\left(\mathbf{q}_k + \sum_{j=1}^{J} \alpha_{k,j} W_V \mathbf{x}_j\right), \qquad \alpha_{k,j} = \frac{\exp\left((W_Q \mathbf{q}_k)^\top (W_K \mathbf{x}_j)/\sqrt{d}\right)}{\sum_{j'=1}^{J} \exp\left((W_Q \mathbf{q}_k)^\top (W_K \mathbf{x}_{j'})/\sqrt{d}\right)}. \quad (1)$$

Without supervision on $\boldsymbol{\alpha}$, the residual connection lets queries differentiate via their learned embeddings alone, with no incentive for $\alpha_{k,j}$ to differ across k. Empirically, attention maps drift toward near-uniform distributions, and reconstruction quality at $K = 8$ equals that at $K = 1$ (Table 3).

Attention supervision loss. We supervise $\boldsymbol{\alpha}$ directly with per-joint functional-group labels $y_j \in \{0, \ldots, K{-}1\} \cup \{-1\}$, where -1 marks unknown joints using the row-wise cross-entropy:

$$\mathcal{L}_{\text{attn}} = -\sum_{k:|S_k|>0} \sum_{j=1}^{J} T_{k,j} \log \alpha_{k,j}, \quad \text{where} \quad T_{k,j} = \begin{cases} 1/|S_k|, & \text{if } y_j = k, \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

$S_k = \{j : y_j = k\}$ and T is the uniform target distribution for each non-empty part k. Unlabeled joints ($y_j{=}{-}1$) contribute no terms; labels are obtained automatically from joint names via canonical matching (Sec. B). Labels are used only during training; at inference, joints are routed by the learned cross-attention alone. Gradients flow through the softmax to W_Q, W_K, and the encoder features, encouraging functional cluster specialisation. Eqs. (1) and (2) are written per temporal position; in practice the loss is averaged over batch and time.

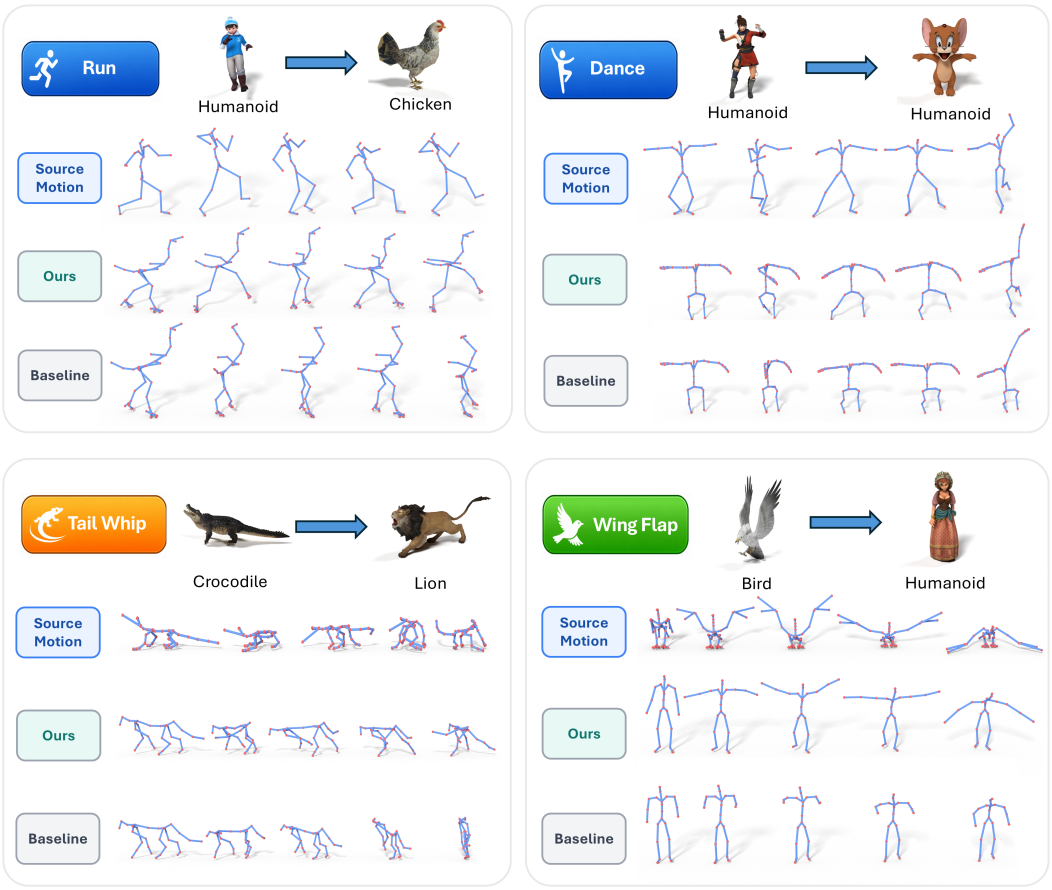


Figure 3: **Cross-topology motion transfer.** Each 2×2 panel shows *Source Motion*, *SAMoR* (K=8), and the single-query *Baseline* (K=1, same cross-topology decoder). SAMoR preserves appendage dynamics across topology shifts; K=1 retains coarse body rhythm but loses limb, tail, and wing detail, consistent with the reconstruction and round-trip gaps in Table 3. All visualizations are root-relative.

Joint-name dropout. Joint names enter the encoder as CLIP embeddings. We randomly replace each embedding with a shared zero vector during training; since $\mathcal{L}_{\text{attn}}$ is still applied, the model learns to route joints from motion patterns and skeleton topology rather than name strings alone, improving robustness to unseen rigs with missing or non-semantic names.

3.3 Token Generation, Editing, and Mesh Animation

Text-conditioned token generation. We train a MaskGIT [5] transformer over the $(T/4) \times K$ grid of SAMoR tokens, conditioned on text and the target skeleton. The main transformer predicts level-0 RVQ code indices, and a lightweight residual transformer predicts the remaining RVQ levels conditioned on previously decoded codes, following masked-token motion generators. Since generation happens in SAMoR's shared part-token space, the predicted tokens can be decoded on arbitrary target skeletons through the same target-skeleton-conditioned decoder.

Part-wise editing and mesh animation. For editing, we encode a motion into SAMoR tokens and mask a selected subset of part slots; the generator in-fills only the masked slots, localizing changes to the selected functional groups. For raw meshes, any off-the-shelf auto-rigging system [33, 34, 8] provides the skeleton and skinning weights; SAMoR then generates or transfers motion, recovering local rotations via IK before applying LBS.

4 Experiments

4.1 Experimental Setup

Datasets. We use three datasets covering the cross-topology spectrum. *HumanML3D (H3D)* [13] provides ∼23K SMPL-22 humanoid motion-caption pairs in the standard 263-D feature format expected by the MoMask evaluator (root trajectory, relative joint positions, 6D joint rotations, joint velocities, and foot-contact indicators concatenated per frame). *Truebones Zoo* [37] is a motion-capture library of 65 animal species (quadrupeds, birds, reptiles, insects) with varying joint counts and named motion clips. *Objaverse-XL (OXL)* [7] is our curated subset of rigged and skinned animated characters from Objaverse-XL spanning humanoid, quadruped, and non-biological rigs. Curation removes broken or misaligned rigs, near-static and duplicated assets, rigs outside 5 to 96 joints, and clips shorter than 16 frames (Sec. F). All motions are converted to a unified format of per-joint positions $\mathbf{p}_j^t$ and velocities $\mathbf{v}_j^t$. For OXL and Zoo, the first frame of each motion serves as the pose anchor, and for HumanML3D-compatible experiments the canonical SMPL rest pose is used. Character-level splits guarantee that OXL validation characters are never seen during training. For the optional text-conditioned generator, one motion caption per OXL character is generated by rendering the skeleton and textured mesh motion and captioning the renders with a vision-language model (Gemini Flash [10]). These captions are only used to train the SAMoR MaskGIT generation model. HumanML3D is built from AMASS [25] and HumanAct12 [14], and AMASS is used under its non-commercial research license. Objaverse-XL is used under ODC-By v1.0 with per-object licenses. Truebones Zoo [37] is a commercial royalty-free library, and we do not redistribute its motions. We will release the code, the OXL split, the functional-group annotations, and the captions, and we reference OXL assets by identifier without redistributing them.

All characters are normalized by mesh bounding box, up-axis, and facing direction. During training, $T{=}64$ frames are randomly sampled from each motion clip; full preprocessing details are in Sec. F. Table 1 summarises dataset scale.

Table 1: Dataset statistics. During training, $T{=}64$ frames are randomly sampled from each motion clip at 20 fps. OXL and Zoo splits are character-disjoint; H3D uses the standard split protocol with a single shared SMPL skeleton. Joint counts are the range across the corpus.

Dataset	Train entities	Val entities	Train motions	Val motions	Joints
HumanML3D [13]	1 (SMPL)	1 (SMPL)	23,384	4,384	22
Truebones Zoo [37]	54 species	11 species	939	110	9–71
Objaverse-XL (ours)	67,646	3,528	67,646	3,528	5–96
Total	67,701	3,540	91,969	8,022	5–96

Baselines. For HumanML3D (Table 2) we compare against T2M-GPT [47], MoMask [12], and MoGenTS [46] for both text-to-motion generation and VQ-VAE reconstruction. For cross-topology reconstruction (Table 3) we adapt both tokenizers to variable-J skeletons: T2M-GPT is extended with a single-stream wrapper that treats all J joints as a flat per-frame feature ($K{=}1$ style), while MoGenTS is extended to process per-joint position-and-velocity features for arbitrary J, producing a $T{\times}J$ grid of per-joint tokens ($K{=}J$ style). Both are padded to the max joint count ($J{=}96$); padded positions are masked during attention and loss computation for fairness. These adaptations instantiate two natural tokenizer extremes—global fixed-size tokenization and per-joint tokenization—and are not intended as optimised arbitrary-topology systems. Motion generators such as AnyTop [9] synthesize new motions for a given skeleton topology rather than encoding an existing motion into a reusable discrete latent, and therefore do not participate in the reconstruction or transfer evaluations.

Metrics. On HumanML3D we report FID, R-Precision at Top-1/2/3, MM-Dist, and MPJPE (mm) for VQ-VAE reconstruction. The MPJPE convention is computed on the 22-joint relative-pose representation in the canonical body frame (root excluded); SAMoR-unified enters the same column under the body-joint-only protocol (global root and foot-contact channels excluded) after inverting bounding-box normalization, and is not evaluated under the full 263-D protocol. For cross-topology reconstruction we report MPJPE in 10^{-2} normalized units (bounding-box frame, per-frame root subtracted) and bone-length error. Unless otherwise stated, all cross-topology experiments use the shared $d{=}6$ position-and-velocity representation; local rotations are used only in HumanML3D-compatible settings where a consistent rest pose is available. Metric definitions are in Sec. C.

Table 2: HumanML3D benchmark. Top: text-to-motion generation; bottom: VQ-VAE reconstruction. MPJPE in mm (22-joint canonical-frame, root excluded). † numbers from papers.

Method	FID↓	Top1↑	Top2↑	Top3↑	MM-Dist↓	MPJPE↓
Real motion	$0.002_{\pm.000}$	$0.511_{\pm.003}$	$0.703_{\pm.003}$	$0.797_{\pm.002}$	$2.974_{\pm.008}$	—
Text-to-motion generation						
T2M-GPT†	$0.141_{\pm.005}$	$0.492_{\pm.003}$	$0.679_{\pm.002}$	$0.775_{\pm.002}$	$3.121_{\pm.009}$	—
MoMask†	$0.045_{\pm.002}$	$0.521_{\pm.002}$	$0.713_{\pm.002}$	$0.807_{\pm.002}$	$2.958_{\pm.008}$	—
MoGenTS†	$\mathbf{0.033}_{\pm.001}$	$\mathbf{0.529}_{\pm.003}$	$\mathbf{0.719}_{\pm.002}$	$\mathbf{0.812}_{\pm.002}$	$\mathbf{2.867}_{\pm.006}$	—
SAMoR-H3D	$0.039_{\pm.002}$	$0.523_{\pm.002}$	$0.715_{\pm.003}$	$\mathbf{0.812}_{\pm.001}$	$2.897_{\pm.008}$	—
VQ-VAE reconstruction						
T2M-GPT†	0.070	0.489	0.676	0.775	3.130	58.0
MoMask†	$0.019_{\pm.000}$	$0.508_{\pm.003}$	$0.701_{\pm.002}$	$0.795_{\pm.002}$	$2.999_{\pm.006}$	29.5
MoGenTS†	$\mathbf{0.005}_{\pm.000}$	$0.504_{\pm.002}$	$\mathbf{0.702}_{\pm.002}$	$0.797_{\pm.002}$	$2.978_{\pm.006}$	**13.8**
SAMoR-H3D	$0.008_{\pm.000}$	$\mathbf{0.514}_{\pm.001}$	$0.701_{\pm.002}$	$\mathbf{0.798}_{\pm.002}$	$\mathbf{2.977}_{\pm.005}$	17.3
Cross-topology model, evaluated on H3D body joints						
SAMoR-unified‡ (body-joint only)	—	—	—	—	—	15.1

‡Trained on the full cross-topology corpus; MPJPE is computed under the same root-excluded 22-joint body protocol as the reconstruction baselines and is comparable within this root-excluded MPJPE protocol. Full 263-D metrics (FID, R-Precision, MM-Dist) involving root and foot-contact channels are not reported.

4.2 HumanML3D Benchmark

To handle the root trajectory and foot-contact channels in HumanML3D's 263-D representation, we add a parallel 1D-convolutional VQ branch for global channels following MoGenTS [46]. Table 2 reports both text-to-motion generation (top) and VQ-VAE reconstruction (bottom).

Results. On generation, SAMoR-H3D's FID of 0.039 sits between MoGenTS (0.033) and MoMask (0.045); Top-3 R-Precision ties MoGenTS at 0.812 and MM-Dist (2.897) is competitive, confirming that the cross-topology part tokenizer does not hurt fixed-skeleton generation quality. On reconstruction, SAMoR-H3D (17.3 mm MPJPE, 0.008 FID) beats MoMask and T2M-GPT by large margins and leads MoGenTS on Top-1, Top-3, and MM-Dist despite higher MPJPE. SAMoR-unified is evaluated under the root-excluded body-joint protocol only (no root or foot-contact channels) and does not participate in the full 263-D metrics (FID, R-Precision, MM-Dist); its body-joint MPJPE of 15.1 mm is comparable within this root-excluded MPJPE protocol, indicating that heterogeneous training preserves a useful humanoid body-joint sub-vocabulary.

4.3 Cross-Topology Reconstruction and Ablations

Setup and baselines. We evaluate on the curated SAMoR validation set: HumanML3D humanoid clips, held-out OXL characters (*zero character overlap* with training), and Truebones Zoo animals. Per-source columns in Table 3 isolate each regime. Baseline adaptations are as described in Sec. 4.1; the strongest, MoGenTS (per-joint adapt., $K{=}J$), reaches $16.14{\times}10^{-2}$ MPJPE, $5.8\times$ above SAMoR.

Per-source analysis. SAMoR leads on every source; the gap is largest on H3D (2.28 vs. 16.02, $7.0\times$). Ablation rows trace the gains to structural inductive biases: K=1 raises Zoo MPJPE by $+82\%$; removing $\mathcal{L}_{\text{attn}}$ nearly doubles the average error and consistently degrades every source, while also raising Bone-Error from 0.79 to 1.16, showing the model loses skeleton fidelity, not just positional accuracy. RVQ provides consistent gains across sources; removing it raises source-macro MPJPE by 47%.

Ablation summary. The ablations confirm three contribution tiers. *Core:* $K{>}1$ queries with $\mathcal{L}_{\text{attn}}$ are essential—removing either raises MPJPE by $83{-}103\%$. Crucially, $K{=}8$ without supervision (5.04) nearly matches $K{=}1$ (5.59), confirming that supervision—not query count—drives structured routing; we therefore ablate these two alternatives to isolate query count from cluster specialisation. *Structural:* the graph transformer contributes 10%, the bone-length loss mainly lowers Bone-Error (0.91 to 0.79), and RVQ accounts for $+47\%$ when removed. *Name conditioning* costs only $13{-}16\%$ when omitted, keeping the pipeline robust to non-semantic rigs; this moderate impact is expected given that canonical label coverage is 100% on H3D but only 69% on OXL and 81% on Zoo (details in Sec. E).

Table 3: Cross-topology reconstruction and round-trip transfer consistency on the SAMoR validation set (character-disjoint from training). **MPJPE** and **BoneErr**: reconstruction quality per source split (H3D, OXL, Zoo); *All* is the macro-average. **RT-MPJPE**: round-trip MPJPE—source→canonical SMPL-22→source (pelvis-aligned); reported for OXL and Zoo only (H3D shares the canonical SMPL-22 topology, so cross-topology transfer cost is not defined). − indicates the method does not support cross-topology transfer. Values in $\times 10^{-2}$ normalized units.

	MPJPE ($\times 10^{-2}$) ↓				BoneErr	RT-MPJPE ($\times 10^{-2}$) ↓	
Method / Variant	All	OXL	H3D	Zoo	($\times 10^{-2}$) ↓	OXL	Zoo
Baselines (variable-J adaptations)							
T2M-GPT [47]	35.77	35.03	36.59	35.64	11.76	–	–
MoGenTS [46] (per-joint adapt.)	16.14	16.36	16.02	16.07	10.42	–	–
SAMoR ablations — Architecture							
K=1 (single part query)	5.59	5.72	4.67	6.37	1.25	8.78	10.03
w/o RVQ (single-level VQ)	4.04	4.08	3.38	4.66	0.90	8.16	9.83
w/o graph transformer	3.03	2.93	2.56	3.69	0.88	7.53	9.65
SAMoR ablations — Losses							
w/o $\mathcal{L}_{\text{attn}}$ (K=8)	5.04	4.67	4.53	5.98	1.16	8.38	9.89
w/o bone-length loss	2.78	2.62	2.29	3.44	0.91	6.97	9.67
SAMoR ablations — Joint name conditioning							
w/o joint-name emb. (train)	3.10	2.97	2.50	3.86	0.85	5.87	9.48
w/o joint-name emb. (infer.)	3.19	2.73	2.69	4.06	0.84	6.06	9.27
SAMoR (full)	**2.75**	**2.46**	**2.28**	**3.50**	**0.79**	**5.74**	**8.72**

Table 4: Zero-shot paired transfer on the Mixamo test set of Aberman et al. [1]. MPJPE ($\times 10^{-2}$, root-relative) against the Mixamo motion on the target, over 107 clips and the 8×7 ordered character pairs. *Within* and *cross* pairs come from the same or from different structural groups.

Method	All	Within	Cross
K=1 (single part query)	8.96	8.84	9.05
w/o $\mathcal{L}_{\text{attn}}$ (K=8)	7.67	7.29	7.93
SAMoR (K=8)	**4.80**	**4.66**	**4.90**

Joint-name conditioning. Both name-ablation rows stay within $+16\%$ of the full model, confirming SAMoR does not rely on joint-name strings as a shortcut. The full dropout-probability sweep is in Sec. E. Part-query assignment visualizations in Sec. J show that appendages such as insect legs, mechanical limbs, and bird wings are routed to functional arm / leg slots even without joint-name embeddings, indicating that skeleton topology and motion patterns alone recover functionally meaningful routing.

4.4 Cross-Topology Transfer and Round-Trip Consistency

In general, cross-topology transfer does not have paired ground-truth target motions: given a source motion decoded onto a different target skeleton, there is no unique correct target trajectory. Hence, we evaluate transfer with (i) qualitative comparisons in Fig. 3, (ii) *round-trip MPJPE* (RT-MPJPE) in Table 3, where a motion is mapped source→canonical SMPL-22→source and compared against the original, and (iii) paired references on Mixamo in Table 4. RT-MPJPE is a transfer-consistency proxy; Fig. 3 provides the perceptual complement.

Paired transfer on Mixamo. The Mixamo test set of Aberman et al. [1] provides the same 107 clips on eight characters. We use the simplified skeletons of their evaluation, which remove the fingers. Four characters share a standard humanoid structure with 23 joints and differ mainly in proportions, and the other four have modified skeletal structures with 28 joints. For every ordered pair of distinct characters and every clip, we encode the first 64 frames on the source, decode the tokens on the target, and compare the result with the motion that Mixamo provides for the same clip on the target ($5{,}992$ zero-shot transfers). Mixamo data are used only for evaluation. The reference motions come from Mixamo's own retargeting, so this evaluation measures agreement with a rotation-based retargeting of the same action and does not measure contact or physical plausibility. In Table 4, SAMoR reaches

4.80. Pairs within and across the two structural groups give similar errors (4.66 and 4.90). The $K{=}1$ and w/o $\mathcal{L}_{\text{attn}}$ variants use the same data and training configuration as SAMoR and differ only in the ablated component. Sec. H gives the protocol.

Transfer baseline and qualitative results. Fixed-skeleton tokenizers cannot decode to unseen skeleton topologies (marked "$-$" in Table 3); we therefore use the $K{=}1$ variant of SAMoR as the transfer baseline—the same architecture and cross-topology decoder, but with a single part query that collapses all joint features into one global token instead of $K{=}8$ specialized part tokens. Fig. 3 shows SAMoR preserves appendage-specific dynamics across topology shifts, while $K{=}1$ retains only coarse body rhythm and loses limb, tail, and wing detail, consistent with the reconstruction and round-trip gaps in Table 3. Part-query assignment visualizations are in Sec. J.

Round-trip consistency. The RT-MPJPE columns in Table 3 provide a quantitative proxy for transfer consistency, routing source motions through a canonical SMPL-22 skeleton and back. SAMoR achieves the lowest OXL RT-MPJPE (5.74) and Zoo RT-MPJPE (8.72) among all high-fidelity variants, indicating that the part-token representation preserves source motion through an intermediate topology. $K{=}1$ and w/o $\mathcal{L}_{\text{attn}}$ substantially increase RT-MPJPE (8.78/8.38 OXL; 10.03/9.89 Zoo), showing that collapsed or unsupervised pooling loses transfer-relevant part structure. Removing RVQ or the graph transformer also degrades round-trip consistency, suggesting that both discrete residual coding and graph-aware encoding contribute to robust cross-topology decoding.

5 Conclusion

We presented SAMoR, a cross-topology motion VQ-VAE that encodes motion of different skeleton topologies into a fixed set of $K{=}8$ shared part-level tokens. The central challenge is that naive multi-query pooling collapses without supervision; our topology-agnostic attention supervision loss addresses this by guiding each query toward a distinct functional joint cluster, while joint-name dropout ensures robustness to unseen rigs with missing or non-semantic joint names. Trained on a unified corpus of humanoid, animal, and cross-topology characters, SAMoR reaches 2.75×10^{-2} normalized MPJPE—5.8× below the strongest adapted variable-J tokenizer baseline—while remaining competitive with fixed-skeleton specialists on HumanML3D. The shared discrete codebook directly enables cross-topology transfer, text-conditioned generation, and localized part-wise editing, with representative qualitative examples in the main paper, additional qualitative results in Sec. J, and videos on our project website.

Limitations and future work. SAMoR operates on root-relative joint positions in the cross-topology setting, because heterogeneous assets lack a unified rest-pose reference that makes local rotations comparable across rigs. Mesh animation therefore requires inverse kinematics, which cannot determine the rotation of a bone about its own axis (Fig. 4). On Mixamo, this twist ambiguity alone gives a mean vertex error of 2.26×10^{-2} (Sec. I). The tokenizer does not model the global root trajectory, and our mesh evaluations are root-relative. Transferring a root trajectory between characters with different proportions and contact patterns needs a target-aware trajectory model. We do not report foot-sliding or contact metrics, because our corpus lacks consistent end-effector, contact, and floor annotations across rigs. The number of functional groups is fixed before training. We compare $K{=}4$, 8, and 12 in Table 8, and learning the number and structure of groups automatically is future work. SAMoR also assumes a valid rig (Sec. 3).

Acknowledgments and Disclosure of Funding

TB acknowledges support from the UKRI Engineering and Physical Sciences Research Council (EPSRC) through the Future Leaders Fellowship [grant number MR/Y018818/1]. GPM is a member of the Machine Learning Cluster of Excellence, EXC number 2064/1 – Project number 390727645, and is endowed by the Carl Zeiss Foundation.

References

[1] Kfir Aberman, Peizhuo Li, Dani Lischinski, Olga Sorkine-Hornung, Daniel Cohen-Or, and Baoquan Chen. Skeleton-aware networks for deep motion retargeting. *ACM Transactions on Graphics*, 39(4):62:1–62:14, 2020.

[2] Nikos Athanasiou, Alpár Ceske, Markos Diomataris, Michael J. Black, and Gül Varol. MotionFix: Text-driven 3d human motion editing. In *SIGGRAPH Asia 2024 Conference Papers*, 2024.

[3] Nikos Athanasiou, Mathis Petrovich, Michael J Black, and Gül Varol. Sinc: Spatial composition of 3d human motions for simultaneous action generation. In *ICCV*, pages 9984–9995, 2023.

[4] Ilya Baran and Jovan Popović. Automatic rigging and animation of 3d characters. *ACM TOG*, 26(3):72–es, 2007.

[5] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *CVPR*, pages 11315–11325, 2022.

[6] 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 *CVPR*, pages 18000–18010, 2023.

[7] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *NeurIPS*, 36:35799–35813, 2023.

[8] Yufan Deng, Yuhao Zhang, Chen Geng, Shangzhe Wu, and Jiajun Wu. Anymate: A dataset and baselines for learning 3d object rigging. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Papers*, pages 1–10, 2025.

[9] Inbar Gat, Sigal Raab, Guy Tevet, Yuval Reshef, Amit Haim Bermano, and Daniel Cohen-Or. Anytop: Character animation diffusion with any topology. In *SIGGRAPH*, pages 1–10, 2025.

[10] Gemini Team, Google. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[11] Chuan Guo, Inwoo Hwang, Jian Wang, and Bing Zhou. Snapmogen: Human motion generation from expressive texts. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[12] Chuan Guo, Yuxuan Mu, Muhammad Gohar Javed, Sen Wang, and Li Cheng. Momask: Generative masked modeling of 3d human motions. In *CVPR*, pages 1900–1910, 2024.

[13] 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 *CVPR*, pages 5152–5161, 2022.

[14] Chuan Guo, Xinxin Zuo, Sen Wang, Shihao Zou, Qingyao Sun, Annan Deng, Minglun Gong, and Li Cheng. Action2motion: Conditioned generation of 3d human motions. In *Proceedings of the ACM International Conference on Multimedia*, 2020.

[15] Yannan He, Garvita Tiwari, Tolga Birdal, Jan Eric Lenssen, and Gerard Pons-Moll. Nrdf: Neural riemannian distance fields for learning articulated pose priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1661–1671, 2024.

[16] Yannan He, Garvita Tiwari, Xiaohan Zhang, Pankaj Bora, Tolga Birdal, Jan Eric Lenssen, and Gerard Pons-Moll. Molingo: Motion–language alignment for text-to-human motion generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

[17] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2022.

[18] Tomas Jakab, Ruining Li, Shangzhe Wu, Christian Rupprecht, and Andrea Vedaldi. Farm3d: Learning articulated 3d animals by distilling 2d diffusion. In *2024 International Conference on 3D Vision (3DV)*, pages 852–861. IEEE, 2024.

[19] Biao Jiang, Xin Chen, Wen Liu, Jingyi Yu, Gang Yu, and Tao Chen. Motiongpt: Human motion as a foreign language. *NeurIPS*, 36:20067–20079, 2023.

[20] Sunmin Lee, Taeho Kang, Jungnam Park, Jehee Lee, and Jungdam Won. SAME: Skeleton-agnostic motion embedding for character animation. In *SIGGRAPH Asia 2023 Conference Papers*, pages 1–11, 2023.

[21] Peizhuo Li, Sebastian Starke, Yuting Ye, and Olga Sorkine-Hornung. WalkTheDog: Cross-morphology motion alignment via phase manifolds. In *ACM SIGGRAPH 2024 Conference Papers*, 2024.

[22] Zizhang Li, Dor Litvak, Ruining Li, Yunzhi Zhang, Tomas Jakab, Christian Rupprecht, Shangzhe Wu, Andrea Vedaldi, and Jiajun Wu. Learning the 3d fauna of the web. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9752–9762, 2024.

[23] Di Liu, Anastasis Stathopoulos, Qilong Zhangli, Yunhe Gao, and Dimitris Metaxas. Lepard: Learning explicit part discovery for 3d articulated shape reconstruction. *Advances in Neural Information Processing Systems*, 36:54187–54198, 2023.

[24] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, October 2015.

[25] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. AMASS: Archive of motion capture as surface shapes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5442–5451, 2019.

[26] Jebastin Nadar, Simone Foti, and Tolga Birdal. Posed-flow: Versatile and guided flow matching model of human pose. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21165–21175, 2026.

[27] Mathis Petrovich, Michael J Black, and Gül Varol. Tmr: Text-to-motion retrieval using contrastive 3d human motion synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9488–9497, 2023.

[28] Ekkasit Pinyoanuntapong, Pu Wang, Minwoo Lee, and Chen Chen. Mmm: Generative masked motion model. In *CVPR*, pages 1546–1555, 2024.

[29] Matthias Plappert, Christian Mandery, and Tamim Asfour. The kit motion-language dataset. *Big data*, 4(4):236–252, 2016.

[30] Sigal Raab, Inbal Leibovitch, Guy Tevet, Moab Arar, Amit H Bermano, and Daniel Cohen-Or. Single motion diffusion. *arXiv preprint arXiv:2302.05905*, 2023.

[31] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, pages 8748–8763. PmLR, 2021.

[32] Davis Rempe, Tolga Birdal, Aaron Hertzmann, Jimei Yang, Srinath Sridhar, and Leonidas J Guibas. Humor: 3d human motion model for robust pose estimation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 11488–11499, 2021.

[33] Chaoyue Song, Xiu Li, Fan Yang, Zhongcong Xu, Jiacheng Wei, Fayao Liu, Jiashi Feng, Guosheng Lin, and Jianfeng Zhang. Puppeteer: Rig and animate your 3d models. *arXiv preprint arXiv:2508.10898*, 2025.

[34] Chaoyue Song, Jianfeng Zhang, Xiu Li, Fan Yang, Yiwen Chen, Zhongcong Xu, Jun Hao Liew, Xiaoyang Guo, Fayao Liu, Jiashi Feng, et al. Magicarticulate: Make your 3d models articulation-ready. In *CVPR*, pages 15998–16007, 2025.

[35] Keqiang Sun, Dor Litvak, Yunzhi Zhang, Hongsheng Li, Jiajun Wu, and Shangzhe Wu. Ponymation: Learning articulated 3d animal motions from unlabeled online videos. In *European Conference on Computer Vision*, pages 100–119. Springer, 2024.

[36] Guy Tevet, Sigal Raab, Brian Gordon, Yonatan Shafir, Daniel Cohen-Or, and Amit H Bermano. Human motion diffusion model. *arXiv preprint arXiv:2209.14916*, 2022.

[37] Truebones Motions Animation Studios. Truebones. `https://truebones.gumroad.com/`, 2022. Accessed: 2026-05-03.

[38] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *NeurIPS*, 30, 2017.

[39] Yin Wang, Mu Li, Jiapeng Liu, Zhiying Leng, Frederick WB Li, Ziyao Zhang, and Xiaohui Liang. Fg-t2m++: Llms-augmented fine-grained text driven human motion generation. *IJCV*, 133(7):4277–4293, 2025.

[40] Shangzhe Wu, Tomas Jakab, Christian Rupprecht, and Andrea Vedaldi. Dove: Learning deformable 3d objects by watching videos. *International Journal of Computer Vision*, 131(10):2623–2634, 2023.

[41] Lixing Xiao, Shunlin Lu, Huaijin Pi, Ke Fan, Liang Pan, Yueer Zhou, Ziyong Feng, Xiaowei Zhou, Sida Peng, and Jingbo Wang. Motionstreamer: Streaming motion generation via diffusion-based autoregressive model in causal latent space. In *ICCV*, pages 10086–10096, 2025.

[42] Mingxi Xu, Qi Wang, Zhengyu Wen, Phong Dao Thien, Zhengyu Li, Ning Zhang, Xiaoyu He, Wei Zhao, Kehong Gong, and Mingyuan Zhang. Necromancer: Breathing life into skeletons via bvh animation. *arXiv preprint arXiv:2602.06548*, 2026.

[43] Zhan Xu, Yang Zhou, Evangelos Kalogerakis, Chris Landreth, and Karan Singh. Rignet: Neural rigging for articulated characters. *arXiv preprint arXiv:2005.00559*, 2020.

[44] Chun-Han Yao, Wei-Chih Hung, Yuanzhen Li, Michael Rubinstein, Ming-Hsuan Yang, and Varun Jampani. Lassie: Learning articulated shapes from sparse image ensemble via 3d part discovery. *Advances in Neural Information Processing Systems*, 35:15296–15308, 2022.

[45] Zhengdi Yu, Simone Foti, Linguang Zhang, Amy Zhao, Cem Keskin, Stefanos Zafeiriou, Tolga Birdal, et al. Geometric neural distance fields for learning human motion priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2232–2242, 2026.

[46] Weihao Yuan, Yisheng He, Weichao Shen, Yuan Dong, Xiaodong Gu, Zilong Dong, Liefeng Bo, and Qixing Huang. Mogents: Motion generation based on spatial-temporal joint modeling. *NeurIPS*, 37:130739–130763, 2024.

[47] Jianrong Zhang, Yangsong Zhang, Xiaodong Cun, Yong Zhang, Hongwei Zhao, Hongtao Lu, Xi Shen, and Ying Shan. Generating human motion from textual descriptions with discrete representations. In *CVPR*, pages 14730–14740, 2023.

[48] Chongyang Zhong, Lei Hu, Zihao Zhang, and Shihong Xia. Attt2m: Text-driven human motion generation with multi-perspective attention mechanism. In *ICCV*, pages 509–519, 2023.

[49] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *CVPR*, pages 5745–5753, 2019.

[50] 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 *ECCV*, pages 126–143. Springer, 2024.

Appendix

A Implementation Details

A.1 SAMoR Stage 1: Part VQ-VAE

Architecture hyperparameters. Table 5 summarises the key architectural parameters. The encoder and decoder each consist of L=4 *ChainAwareGraphTransformerBlock* layers. Each block alternates temporal self-attention (standard multi-head attention along the T axis per joint) and spatial graph-transformer attention (along the J axis per frame) with a two-layer FFN. Temporal downsampling by $4\times$ is achieved through two stacked strided Conv1d layers (stride 2 each), producing $(T/4)$ temporal positions.

Table 5: SAMoR Stage 1 architecture and training hyperparameters.

Parameter	**Value**	**Note**
Model dimension D	512	
Encoder/decoder blocks L	4 + 4	
Attention heads	4	head dim $= 128$
FFN hidden dim	1024	
Temporal downsampling	$4\times$	two stride-2 Conv1d
Max joints $J_{\max}$	96	padded; real joints masked
Clip length T	64	frames randomly sampled
Part queries K	8	
RVQ depth R	6	
Codebook size V	512	per RVQ level, EMA updates
Quantize-dropout prob.	0.5	drops deeper RVQ levels
Name-dropout prob. p	0.3	
Max hop distance	20	clamped for bias lookup
Edge types	6	see Sec. A.2
Dropout	0.1	
Batch size	64	
Learning rate	10^{-4}	5% linear warmup
Optimizer	AdamW	β_1=0.9, β_2=0.999
Precision	bf16	
EMA decay	0.99	on codebook entries
Epochs	100	
Hardware	4× GH200 96 GB	
Training time	∼96 h	wall-clock

Training losses. The VQ-VAE is trained with the weighted sum:

$$\mathcal{L} = \lambda_{\text{pos}}\,\mathcal{L}_{\text{pos}} + \lambda_{\text{vel}}\,\mathcal{L}_{\text{vel}} + \lambda_{\text{vq}}\,\mathcal{L}_{\text{vq}} + \lambda_{\text{bone}}\,\mathcal{L}_{\text{bone}} + \lambda_{\text{smooth}}\,\mathcal{L}_{\text{smooth}} + \lambda_{\text{attn}}\,\mathcal{L}_{\text{attn}}, \tag{3}$$

where $\mathcal{L}_{\text{pos}}$ and $\mathcal{L}_{\text{vel}}$ are smooth-ℓ_1 losses on root-relative joint positions and frame-to-frame velocities; $\mathcal{L}_{\text{vq}}$ is the VQ commitment loss; $\mathcal{L}_{\text{bone}}$ is the bone-length consistency loss (squared parent-to-child distance error); $\mathcal{L}_{\text{smooth}}$ penalises temporal variation of the quantized part-token latents ($\|z_q^{t+1} - z_q^t\|^2$ averaged over time); and $\mathcal{L}_{\text{attn}}$ is the attention supervision loss. Weights are given in Table 6. A rotation reconstruction loss is added only in the HumanML3D-compatible setting.

Table 6: Loss weights used during Stage 1 training.

Loss	Weight	Description
$\mathcal{L}_{\text{pos}}$	1.0	smooth-ℓ_1 on joint positions
$\mathcal{L}_{\text{vel}}$	0.5	smooth-ℓ_1 on joint velocities
$\mathcal{L}_{\text{vq}}$	1.0	VQ commitment + entropy
$\mathcal{L}_{\text{bone}}$	2.0	bone-length consistency
$\mathcal{L}_{\text{smooth}}$	0.01	latent temporal smoothness
$\mathcal{L}_{\text{attn}}$	1.0	attention supervision

A.2 Graph Structural Biases

Each spatial self-attention layer in the graph-transformer encoder and decoder adds learned structural biases to the pre-softmax attention logits. For a query joint i and key joint j, the total bias is:

$$b(i,j) = \mathbf{q}_i^\top \mathbf{w}^{tq}_{d(i,j)} + \mathbf{k}_j^\top \mathbf{w}^{tk}_{d(i,j)} + \mathbf{q}_i^\top \mathbf{w}^{eq}_{e(i,j)} + \mathbf{k}_j^\top \mathbf{w}^{ek}_{e(i,j)} + \mathbf{q}_i^\top \mathbf{w}^{cq}_{c(i,j)} + \mathbf{k}_j^\top \mathbf{w}^{ck}_{c(i,j)}, \quad (4)$$

where $\mathbf{q}_i, \mathbf{k}_j \in \mathbb{R}^{D/H}$ are per-head query and key vectors, and $\mathbf{w}^{\cdot}_{\cdot}$ are learned embedding vectors looked up by three integer indices.

Hop-distance bias ($d(i,j)$). $d(i,j)$ is the shortest-path distance between joints i and j on the kinematic tree, computed via Floyd–Warshall and clamped to $d_{\max}$=20. Two embedding tables ($\mathbf{w}^{tq}$ and $\mathbf{w}^{tk}$, each of size $(d_{\max}{+}1) \times D$) provide query-side and key-side biases that decay with skeleton distance, encouraging joints that are close in the kinematic tree to attend more strongly to one another.

Semantic edge-type bias ($e(i,j)$). Six semantic categories capture the directed structural relationship between every pair of joints:

Type	Condition
0	$i{=}j$, non-leaf joint (self)
1	j is the direct parent of i
2	j is a direct child of i
3	i and j share the same direct parent (siblings)
4	all other pairs (no direct relation)
5	$i{=}j$, leaf joint (end-effector)

Two embedding tables ($\mathbf{w}^{eq}$, $\mathbf{w}^{ek}$, each $6 \times D$) provide query-side and key-side biases. Types 1–3 distinguish the immediate kinematic neighbourhood; type 5 specialises end-effectors, which often carry distinctive motion signatures (hands, feet, tail tips).

Same-chain binary bias ($c(i,j)$). $c(i,j) \in \{0,1\}$ indicates whether joints i and j belong to the same *kinematic chain*—a maximal linear path in the tree with no branching nodes between them. Chain membership is computed once per skeleton via DFS from the root: single-child nodes continue the current chain; branching nodes start new chains for each child. Two embedding tables ($\mathbf{w}^{cq}$, $\mathbf{w}^{ck}$, each $2 \times D$) encourage joints on the same chain (e.g., all vertebrae of a spine, all segments of a tail) to form coherent part representations.

All bias terms in Eq. (4) are summed and added to the standard scaled dot-product scores before softmax, and are shared across all heads within a block but independent across blocks. Padded joints (beyond the true skeleton size) are masked to $-\infty$ before the softmax.

A.3 Stage 2: MaskGIT Token Generator

A MaskGIT transformer is trained over the $(T/4){\times}K$ grid of SAMoR part tokens, conditioned on a text embedding and the target skeleton. The main transformer (12 layers, D=512, 8 heads) predicts level-0 RVQ code indices; a lightweight residual transformer (4 layers, same D) predicts the remaining $R{-}1$ RVQ levels conditioned on the decoded level-0 codes. Text is encoded by a frozen CLIP text encoder [31] and injected via cross-attention at every layer.

Training objective. Both the main and residual transformers are trained with cross-entropy loss computed only over masked positions (unmasked positions are excluded via `ignore_index`):

$$\mathcal{L}_{\text{S2}} = -\sum_{m \in \mathcal{M}} \log p_\theta\big(\hat{c}_m \mid c_{\setminus \mathcal{M}},\, \mathbf{y},\, \mathcal{S}\big), \quad (5)$$

where $\mathcal{M}$ is the set of masked token positions, $\hat{c}_m$ is the ground-truth code at position m, $\mathbf{y}$ is the CLIP text embedding, and $\mathcal{S}$ is the skeleton conditioning. Label smoothing ϵ=0.1 is applied.

Training details. Stage 2 is trained for 300 epochs with batch size 32, learning rate 10^{-4}, and a cosine decay schedule with linear warm-up, on 4× GH200 96 GB GPUs (∼12 h wall-clock). Tokens

are masked with a mixed strategy: each batch randomly applies either temporal-frame masking (masking all K parts in selected frames) or random-token masking (independently masking tokens across the $(T/4)\times K$ grid), with equal probability. The masking ratio is sampled from a cosine schedule $r = \cos\left(\frac{\pi}{2}u\right)$ for $u \sim \mathcal{U}[0,1]$, and a BERT-style corruption is applied: 88% of masked positions receive a special MASK token, 10% are replaced with a random code, and 2% are left unchanged.

Inference. Tokens are generated iteratively in 18 MaskGIT decoding steps. At each step, the model predicts all masked positions simultaneously; the least-confident tokens are re-masked according to a cosine unmasking schedule, and confident tokens are committed. Classifier-free guidance [17] with scale $s{=}4.0$ and temperature 1.0 is used at inference; the null condition drops both the text embedding and the skeleton conditioning.

A.4 Mesh-to-Animation Frontend

The mesh-to-animation qualitative results in the main paper and supplementary videos use Puppeteer [33] as the automatic rigging and skinning frontend. Given a raw textured mesh, Puppeteer predicts a skeleton and per-vertex skinning weights; the resulting rigged character is then passed directly to SAMoR for motion generation or transfer. Other off-the-shelf auto-rigging systems such as MagicArticulate [34] and AnyMate [8] are compatible with the same pipeline.

B Canonical Joint-Name Matching

Per-joint functional-group labels $y_j \in \{0,\dots,7\} \cup \{-1\}$ are derived automatically from raw joint name strings in two stages.

Stage 1: name cleaning. Raw names undergo a sequence of normalisation steps: (i) strip namespace and vendor prefixes (`mixamorig:`, `Bip01_`, `BN_`, etc.) applied iteratively up to three times to handle nested prefixes; (ii) convert left/right suffix conventions (`_L`, `R_`, `_left`, etc.) to a canonical `Left`/`Right` prefix; (iii) split CamelCase tokens (`LeftForeArm` $\rightarrow$ `Left Fore Arm`); (iv) replace underscores and dots with spaces and collapse whitespace.

Stage 2: canonical lookup. Cleaned names are matched against a dictionary of $\sim$130 synonym groups, each mapping surface variants to a canonical name (e.g., `upper arm`, `upperarm`, `uarm` $\rightarrow$ `left upper arm`). Unresolved names undergo progressive fallback: strip residual numeric suffixes, drop unknown leading tokens, and apply keyword-based regex patterns (e.g., `wing`, `thigh`, `toe`). Joints that remain unmatched after all fallback steps receive $y_j{=}{-1}$ and contribute no terms to the attention supervision loss.

Label coverage. Applying the pipeline above, canonical part labels are successfully assigned to 100% of joints in H3D (standardised MoCap names), 81% in Zoo (semi-standard creature rigs), and 69% in OXL (highly diverse artist-authored rigs with vendor-specific naming conventions). The remaining unlabelled joints are treated as unknown at both training and inference; name dropout during training ensures the model does not rely on label availability.

8-slot functional-group mapping. Matched canonical names are mapped to one of eight part indices via the table in Table 7. Labels are used only during training; at inference, joints are routed by the learned cross-attention.

Number of part slots. We also train SAMoR with $K{=}4$ and $K{=}12$ part queries on the full corpus. The groupings are nested with the eight groups of Table 7. $K{=}4$ merges them into head/neck, torso (lower torso, upper torso, and tail), arms, and legs, and $K{=}12$ splits each arm and leg into an upper and a lower segment at the elbow or knee. Table 8 reports reconstruction on the OXL validation set of Table 3 and paired transfer on Mixamo with the protocol of Sec. H. $K{=}12$ does not improve over $K{=}8$, and $K{=}4$ gives higher errors on both metrics, so performance saturates around $K{=}8$. We keep $K{=}8$ as a trade-off between compactness, interpretable editing, and coverage of our diverse rigs. This choice is tied to our corpus and is not a universal anatomical decomposition.

Table 7: Canonical-name-to-part mapping for the $K=8$ functional-group slots. Entries in each cell are canonical names; starred entries cover indexed variants (`tail1`–`tail5`, `left leg1`–`left leg4`, etc.).

Index	Part name	Canonical joint names
0	lower torso	hips, body
1	upper torso	spine, spine1–3, chest
2	head / neck	neck, neck1–2, head, jaw, left/right eye, nose, left/right ear, tongue, tongue2, left/right eyebrow
3	left arm	left clavicle, left shoulder, left upper arm, left elbow, left forearm, left wrist, left hand, left finger0–4, left wing1–4
4	right arm	right clavicle, right shoulder, right upper arm, right elbow, right forearm, right wrist, right hand, right finger0–4, right wing1–4
5	left leg	left upper leg, left knee, left leg, left horse link, left ankle, left foot, left toe, left toe end, left heel, left leg1–4*
6	right leg	right upper leg, right knee, right leg, right horse link, right ankle, right foot, right toe, right toe end, right heel, right leg1–4*
7	tail	tail, tail1–5*

Table 8: Number of part slots K. Reconstruction MPJPE ($\times 10^{-2}$) on the OXL validation set of Table 3 and paired transfer MPJPE ($\times 10^{-2}$, all pairs) on Mixamo (Sec. H).

K	Groups	Reconstruction	Paired transfer
4	head/neck, torso, arms, legs	3.22	5.49
8	the eight groups of Table 7	2.46	4.80
12	arms and legs split at elbow and knee	2.64	4.97

C Metric Definitions

MPJPE (normalised). For cross-topology evaluation, all positions are expressed in a bounding-box normalised coordinate frame (character scaled to unit bounding box) with the per-frame root joint subtracted. Mean Per-Joint Position Error is then:

$$\text{MPJPE} = \frac{1}{T J_{\text{real}}} \sum_{t=1}^{T} \sum_{j \in \mathcal{J}} \left\| \hat{\mathbf{p}}_j^t - \mathbf{p}_j^t \right\|_2, \tag{6}$$

where $\mathcal{J}$ indexes non-padded joints and values are reported in units of 10^{-2} (i.e., 1% of bounding-box diameter). For HumanML3D, MPJPE is computed on the 22-joint relative-pose representation in the canonical body frame (root excluded) in millimetres, following the standard MoMask evaluator.

Bone-length error. Mean absolute deviation of predicted bone lengths from ground truth, averaged over all parent–child pairs and frames:

$$\text{BoneErr} = \frac{1}{T|\mathcal{E}|} \sum_{t=1}^{T} \sum_{(j,p) \in \mathcal{E}} \left| \|\hat{\mathbf{p}}_j^t - \hat{\mathbf{p}}_p^t\|_2 - \|\mathbf{p}_j^t - \mathbf{p}_p^t\|_2 \right|, \tag{7}$$

where $\mathcal{E}$ is the set of parent–child edges and $p{=}\text{pa}(j)$. Reported in the same normalised units as MPJPE.

HumanML3D metrics. FID, R-Precision (Top-1/2/3), and MM-Dist are computed using the standard feature extractor and evaluation protocol from MoMask [12], applied to 263-D SMPL motion sequences. We follow the same evaluation script without modification.

D Baseline Adaptation Details

Both fixed-skeleton tokenizers are adapted to variable-J inputs using a unified padding scheme: skeletons are zero-padded to $J_{\max}{=}96$ joints, with padded positions masked to $-\infty$ in attention and excluded from the loss computation.

T2M-GPT (K=1 flat). T2M-GPT [47] is a single-stream VQ-VAE that encodes each frame as one global token. We adapt it to arbitrary skeletons by concatenating all J per-joint position-and-velocity

features into a single flat per-frame feature vector of dimension $J_{\max} \times 6$, then applying the original 1D-convolutional encoder unchanged. This gives one token per downsampled frame ($K=1$ style) with no joint-wise structure.

MoGenTS (K=J per-joint grid). MoGenTS [46] uses a 2D grid of tokens over time and joints. We extend it to arbitrary J by replacing the fixed SMPL-22 joint set with a variable number of joints: each joint receives its own $d=6$ position-and-velocity feature (no rotation), producing a $T \times J$ grid of per-joint tokens ($K=J$ style). The architecture (1D convolutions along each axis independently) is applied unchanged; padded joint positions receive zero features and are masked during attention and loss.

Both baselines are trained on the same unified heterogeneous corpus as SAMoR with the same data splits, loss functions (position + velocity smooth-ℓ_1), and training budget, to ensure a fair comparison.

E Joint-Name Conditioning Analysis

We expand on the two name-conditioning rows of the cross-topology ablation (*labels-only* and *drop name @ inference*) with the full training-time dropout sweep and a per-source sensitivity analysis.

Dropout-probability sweep. We train SAMoR variants with name-dropout probability $p \in \{0.0, 0.1, 0.3, 0.5, 0.7, 0.9, 1.0\}$ on the full cross-topology training corpus and evaluate each checkpoint twice: once with joint-name embeddings (with-name), and once with all embeddings replaced by zero at inference time (drop-name). Table 9 reports both columns, the absolute gap, and the relative gap. The trade-off forms a clean Pareto frontier: $p=0.0$ achieves the best with-name MPJPE (2.63×10^{-2}) but is brittle under name removal ($+80\%$ relative gap). SAMoR's default $p=0.3$ raises the with-name MPJPE only from 2.63 to 2.75 while reducing the gap from 80% to 16%, so a single checkpoint deploys on both well-named and unnamed rigs.

Table 9: Name-dropout sweep on the SAMoR validation set. All values are MPJPE in 10^{-2} normalized units (root translation excluded, consistent with the HumanML3D convention). *Gap (rel.)* is (drop-name − with-name)/with-name.

p	with-name	drop-name	gap (abs.)	gap (rel.)	Notes
0.0	**2.63**	4.74	2.11	80%	best with-name; brittle
0.1	2.67	4.27	1.60	60%	
0.3	2.75	3.19	0.44	16%	SAMoR default
0.5	2.85	3.27	0.42	15%	balanced
0.7	2.95	3.22	0.27	9%	
0.9	3.05	3.17	0.12	4%	
1.0	3.10	3.10	0	0%	labels-only; fully robust

Per-source sensitivity. The drop-name penalty is source-dependent. H3D is most name-sensitive: at $p=0$, removing names inflates H3D MPJPE from $(3.19 \to 7.96) \times 10^{-2}$ ($+150\%$), because the model exploits the well-vocabularised SMPL joint names. OXL shows an intermediate 76% jump ($(3.91 \to 6.88) \times 10^{-2}$). Zoo is most robust: only 53% ($(5.30 \to 8.13) \times 10^{-2}$), because cross-species diversity forces reliance on structural priors. At $p=1.0$, removing names no longer changes the error on any source, indicating zero name dependence. Bone error and jitter are largely insensitive to dropout, indicating that name embeddings primarily affect positional accuracy rather than skeleton consistency or temporal smoothness.

Practical implication. The *drop-name @ inference* row uses the $p=0.3$ checkpoint evaluated without names, giving the worst-case behaviour on rigs with no semantic joint labels. The *labels-only* row trains a separate $p=1.0$ checkpoint and shows the lower bound achievable with zero name dependence. In deployment we ship the $p=0.3$ checkpoint; unnamed inputs fall through to the drop-name path with the modest cost reported in the ablation table.

Masking joint names at inference. To test whether the model depends on joint names at inference, we take 500 held-out OXL validation characters for which at least half of the joints have a canonical name. At inference, we replace the name embeddings of a random fraction of the named joints with zeros, using three random masks per character for the intermediate fractions. The model is unchanged. Masking 0%, 25%, 50%, 75%, and 100% of the names gives a root-aligned reconstruction MPJPE of

3.24, 3.27, 3.30, 3.41, and 3.73 ($\times 10^{-2}$), averaged per character and then over characters. Masking half of the names raises the error by 2%, and masking all names raises it by 15%.

F Dataset Details

Dataset statistics are summarised in Table 1 of the main paper; here we give additional preprocessing details.

Shared normalization. All three sources undergo the same geometric normalization before training. Each character is centered and scaled to unit size using the mesh bounding box. The up axis is corrected to $+Z$ using foot and leg joint keyword detection on the skeleton. The facing direction is aligned to $+Y$ using hip–shoulder or foot-pair landmarks. Finally, the skeleton is translated so that the lowest foot joint lies at $Z{=}0$ (Z-floor alignment). For OXL and Zoo, the first frame of each motion clip serves as the pose anchor injected into the encoder and decoder at every graph-transformer layer; for HumanML3D-compatible experiments, the canonical SMPL rest pose is used instead.

HumanML3D. We follow the standard HumanML3D 263-D feature extraction: root trajectory, relative joint positions, 6D joint rotations, joint velocities, and binary foot-contact indicators, concatenated per frame. In addition, we augment the dataset with additional humanoid characters by fitting SMPL meshes to posed bodies, applying skeleton augmentation (random perturbations to bone lengths and proportions), and normalizing the resulting meshes to unit size with the same up-axis and facing-direction convention as OXL and Zoo. The official train/val split is used unchanged.

Truebones Zoo. Raw FBX files are processed in Blender to extract the skeleton hierarchy, per-vertex skinning weights, and motion sequences, which are exported as BVH. The shared normalization is then applied. Motions are split at the species level: 54 species for training and 11 species for validation, ensuring zero species overlap between splits. Validation species are: Ant, Buzzard, Chicken, Crocodile, Elephant, Flamingo, Lion, Ostrich, PolarBearB, Puppy, and SandMouse. At training time, $T{=}64$ frames are randomly sampled from each motion; the full motion is used at evaluation.

Objaverse-XL. Starting from the raw GLB assets, we run a Blender Python pipeline that (i) exports the skinned mesh at the first animation frame as `object.obj` (with UV and texture maps), and (ii) exports the full animation as a BVH in which all joint rotations are expressed relative to the first frame ($\mathbf{R}_t^{\text{rel}} = \mathbf{R}_0^{-1}\mathbf{R}_t$). The shared normalization is applied after export.

The resulting assets pass through a multi-stage curation pipeline:

1. **Rigging validity.** Characters whose skeleton root bone lies outside the mesh bounding box are discarded, along with a manually curated exclusion list of visually confirmed broken rigs.
2. **Inside-mesh filter.** We voxelize the mesh at 64^3 resolution and discard any character for which more than 50% of skeleton joints fall outside the mesh volume, indicating a misaligned or broken rig.
3. **Near-static filter.** Characters whose maximum per-joint rotation standard deviation across time falls below a threshold, combined with low joint displacement, are removed as effectively pose-only assets.
4. **Repetition filter.** Pairs of characters sharing identical joint counts, identical parent connections, and near-identical joint positions on five randomly sampled frames are treated as duplicates; all but one are discarded.
5. **Joint count.** Characters with fewer than 5 or more than 96 joints are excluded.
6. **Short clip filter.** Motion clips shorter than 16 frames at 20 fps are discarded.
7. **Out-of-bounds filter.** Before training, clips in which any joint position exceeds 3.0 normalised units from the origin after shared normalisation are removed, as these indicate residual normalisation failures.

After curation, the remaining characters are split at the character level (zero character overlap between train and val).

OXL character type distribution. We generate one motion caption per character by rendering a $4{\times}4$ keyframe strip (front/side views $\times$ mesh/skeleton) and prompting Gemini Flash [10] to classify

the character type and produce short, medium, and detailed captions. Characters for which Gemini returns no type label (API non-responses) are assigned the GENERIC type. Table 10 reports the resulting type distribution.

Table 10: OXL character type distribution over all 71,174 curated characters. Characters without a Gemini-assigned label are counted as GENERIC.

Type	Characters	%
HUMANOID	24,596	34.6
GENERIC	31,374	44.1
BIPED OTHER	5,866	8.2
QUADRUPED	4,592	6.5
WINGED	3,099	4.4
MULTI-LIMB	1,647	2.3
Total	71,174	100.0

G Pose Anchor Design

SAMoR injects a per-character *pose anchor* into every graph-transformer layer to condition the encoder and decoder on skeleton geometry. When a canonical rest pose is available—as in HumanML3D, where all characters share the SMPL rest pose—we use it directly. For heterogeneous cross-topology assets (OXL, Zoo), rest poses are not consistently defined across rigs: different exporters and artists produce T-poses, A-poses, or arbitrary bind poses that are not comparable across species or asset styles. Using an ill-defined rest pose as the anchor would introduce inconsistent geometry conditioning, reducing the model's ability to generalize across the diverse topologies in our corpus.

We therefore use the *first frame* of each motion clip as the pose anchor for OXL and Zoo. This is a principled choice: the first frame is always available, is free of exporter inconsistencies, and—because different characters start from different first-frame configurations—encourages the model to learn motion representations that are robust to diverse starting poses rather than overfitting to a fixed rest-pose convention. For reconstruction, the first frame belongs to the input clip and is available at test time; since MPJPE is computed in per-frame root-relative coordinates, the absolute starting pose does not inflate the reconstruction metric. At transfer or generation time the pose anchor is drawn exclusively from the *target* rig and carries no information from the source motion, so cross-topology evaluation is free of motion-cue leakage through the anchor.

Choice of anchor frame. We test whether the model depends on the first frame by replacing the anchor of every sample, for both the encoder and the decoder, with the middle, the last, or a random frame of the same 64-frame window. On the validation set of Table 3, Table 11 shows that the first frame, which is used in training, gives the lowest error, and that the other anchors raise the macro-averaged MPJPE by 8% to 12%. The model therefore does not require the first frame, as long as the encoder and the decoder receive the same anchor.

Table 11: Reconstruction MPJPE ($\times 10^{-2}$) when the anchor of both the encoder and the decoder is replaced by another frame of the same 64-frame window. SAMoR (full) on the validation set of Table 3. Windows with fewer than 16 valid frames (13 of 202) are excluded, so the first-frame values differ slightly from Table 3.

Anchor frame	OXL	H3D	Zoo	Macro
First (default)	2.51	2.28	3.62	2.80
Middle	2.60	2.67	3.78	3.01
Last	2.72	2.90	3.81	3.15
Random	2.75	2.57	3.72	3.01

H Paired Transfer on Mixamo

We use the Mixamo test set released with the skeleton-aware retargeting code of Aberman et al. [1]. It contains the same 107 clips, stored as BVH files at 60 fps, for eight characters. Aj, BigVegas,

Kaya, and SportyGranny share a standard humanoid structure and differ in proportions. Goblin_m, Mousey_m, Mremireh_m, and Vampire_m have modified spine and limb structures. Clips with the same name on two characters form a paired example.

We use the simplified skeletons defined in the evaluation code of Aberman et al. The BVH files have 60 to 88 joints, including 30 to 40 finger joints. The simplified skeletons remove the fingers and keep 23 joints for the first four characters and 28 joints for the other four, which add split spine and shoulder joints and toe and head end joints. For each character and clip, we convert the first 64 frames at the original frame rate into the SAMoR format. Positions are taken relative to the root in every frame, rotated from Y-up to Z-up, and shifted so that the lowest joint of the first frame lies at $z{=}0$. The benchmark provides no meshes, so the scale is set by the rest skeleton: positions are divided by the largest absolute coordinate of the root-centred rest skeleton. Joint-name embeddings are computed from the BVH joint names as in training. The source anchor is the first frame of the source clip, and the target anchor is the first frame of the target reference clip.

We decode the source tokens on every other character and compute the mean joint error over frames and joints against the target reference. The error is averaged over the 107 clips of each ordered pair and then over the 56 pairs. *Within* pairs keep the source and target in the same group (24 pairs), and *cross* pairs span the two groups (32 pairs). For reference, reconstructing each clip on its own character gives 2.71.

I IK-Based Mesh Animation

SAMoR outputs joint positions. For mesh animation, we recover local joint rotations with an iterative position-to-rotation IK solver (30 iterations) that starts from the rest-pose bone offsets, and we apply LBS with the character's skinning weights.

Mixamo meshes. For the eight Mixamo characters of Sec. H, we use the official skinned meshes. The meshes are skinned to the full skeletons, so this evaluation uses the full skeletons with fingers. A reference mesh sequence is obtained by driving each mesh with the original Mixamo rotations. We compare it with meshes driven by rotations that IK recovers from (i) the ground-truth joint positions, (ii) SAMoR reconstructions on the same character, and (iii) SAMoR transfers from each of the seven other characters, averaged over sources. The error is the mean vertex distance, in the same normalized units as MPJPE ($\times 10^{-2}$). Over the first 64 frames of all 107 clips, the error is 2.26 for ground-truth positions, 5.11 for reconstructions, and 6.73 for transfers. The first value isolates the cost of discarding rotations. With exact positions, IK cannot recover the rotation of each bone about its own axis, and this twist changes the skinned surface by 2.26×10^{-2}.

OXL characters. On 110 held-out OXL characters with skinned meshes (the first 64-frame window of each), IK from ground-truth positions gives a mean local-rotation error of 7.45° (median 5.89°) but a bone-direction error of only 0.50°, so almost all of the rotation error is twist about the bone axis. The mean vertex error is 2.35×10^{-2} (median 1.50×10^{-2}). With SAMoR reconstructions as IK input, the errors are 11.59° for local rotations (109 characters), 7.60° for bone directions, and 5.29×10^{-2} for vertices. Fig. 4 shows a failure case.

J Additional Qualitative Results

We now provide additional qualitative evaluations regarding the part assignment of SAMoR, followed by end-to-end mesh animation and part composition.

Part assignment visualizations. Fig. 5 shows inference-time part assignment of SAMoR's right-arm and right-leg queries on six held-out characters with no joint names provided. The model routes functionally corresponding appendages to corresponding functional slots across biological (Chicken, Buzzard, Crocodile), non-biological (Robot), insect (Fire Ant), and humanoid rigs.

End-to-end mesh animation and part composition. Fig. 6 shows two complete mesh-animation pipelines—driving an AI-generated, auto-rigged mesh by text-conditioned generation and by cross-topology motion transfer—together with a part-based motion composition example that blends the upper- and lower-body part tokens of different source motions. Video versions of all examples are on our project website.

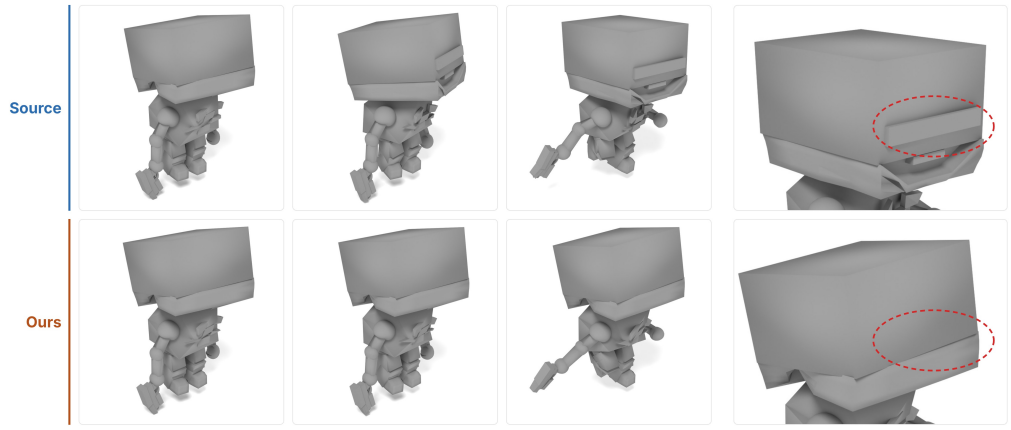


Figure 4: IK failure case on a held-out OXL character. *Source*: the mesh driven by the ground-truth joint rotations. *Ours*: joint positions reconstructed by SAMoR, converted to rotations by IK, and used to drive the same mesh. The first three columns show frames of the clip in temporal order, and the right column enlarges the head in the last frame. In the source animation, the head turns by rolling about its own bone axis. This axial twist leaves the joint positions unchanged, so joint positions alone do not determine it uniquely, and IK does not recover it. The head turn is therefore lost in our result (red circles).

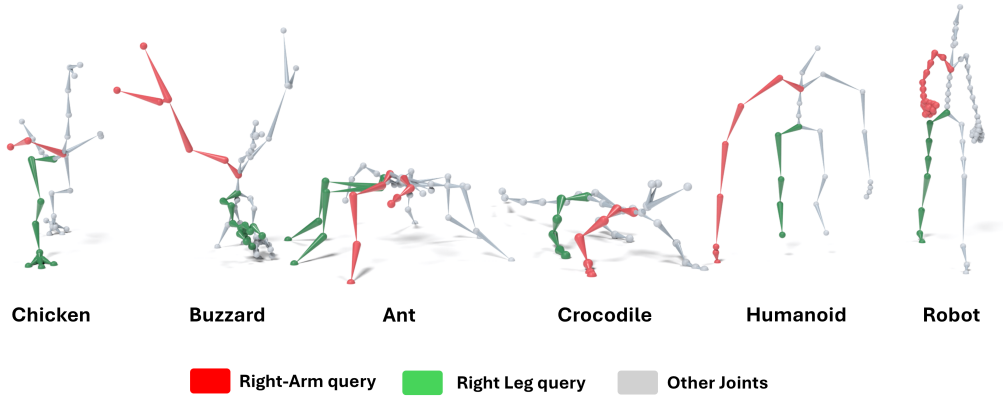


Figure 5: **Functional-query alignment across skeleton topologies.** Dominant joint assignment of SAMoR's right-arm-like (red) and right-leg-like (green) queries on six held-out characters (Chicken, Buzzard, Fire Ant, Crocodile, Humanoid, Robot) at inference time (no joint names provided). Each query localizes to functionally corresponding appendage chains—forelimbs, wings, and leading insect legs for the arm-like query; hindlimbs and walking-leg pairs for the leg-like query—showing that $\mathcal{L}_{\text{attn}}$ (Sec. 3.2) induces functional group routing that generalizes across biological, non-biological, and mechanical rigs.

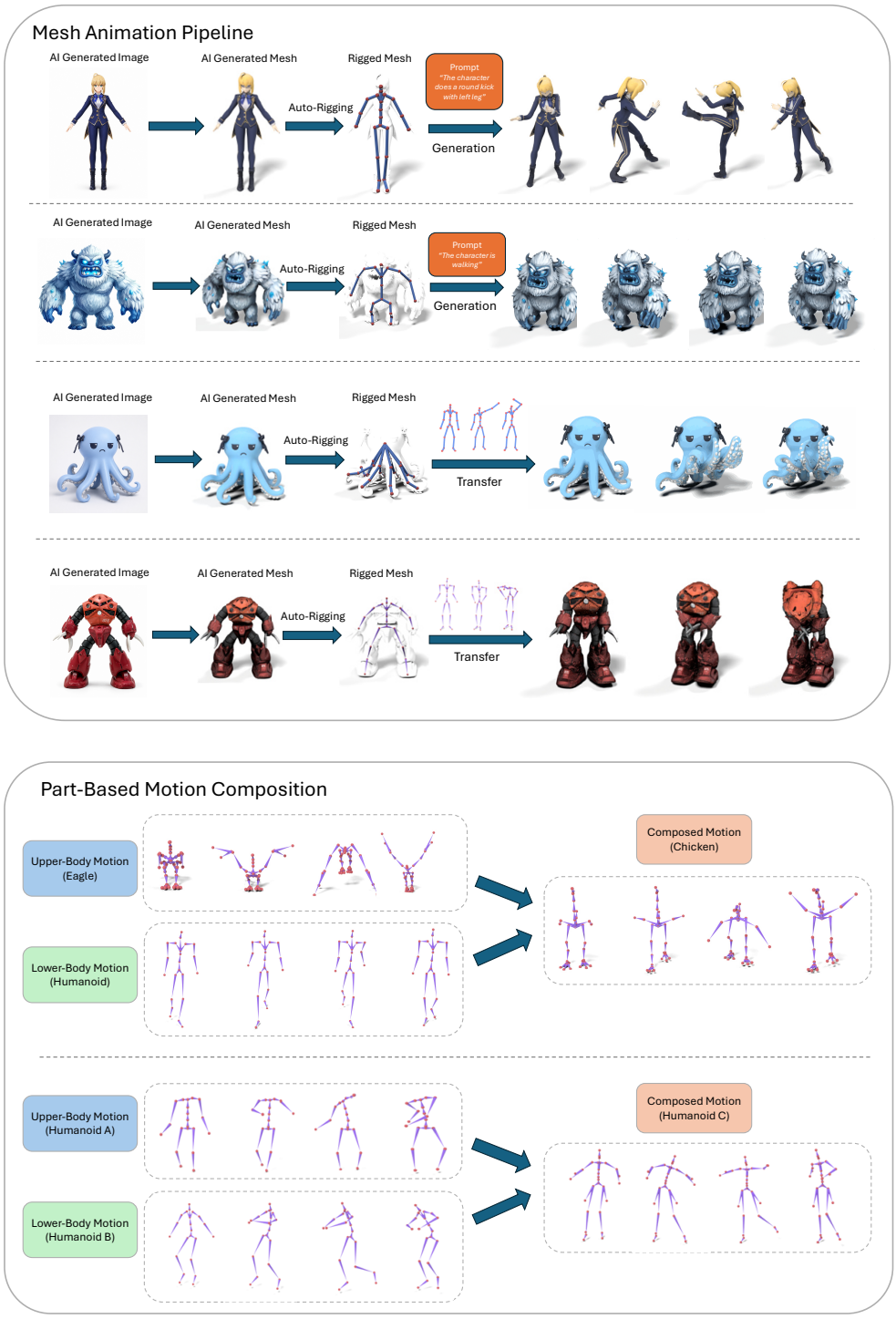


Figure 6: **End-to-end mesh animation and part-based motion composition.** *Top — Mesh Animation Pipeline:* from a single AI-generated image we reconstruct a 3D mesh, auto-rig it, and drive it with SAMoR via text-conditioned generation and cross-topology motion transfer. *Bottom — Part-Based Motion Composition:* SAMoR's per-part tokens let us compose a new motion by pairing the upper-body part tokens of one source motion with the lower-body part tokens of another, yielding a coherent blended motion. Video versions of all examples are on our project website.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and Sec. 1 clearly state three contributions: (i) the SAMoR cross-topology VQ-VAE with part-query pooling, (ii) the topology-agnostic attention supervision loss $\mathcal{L}_{\text{attn}}$, and (iii) a unified heterogeneous motion corpus. All claims are supported by quantitative experiments in Sec. 4 and ablations in Table 3.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Sec. 5 contains a dedicated "Limitations and future work" paragraph discussing the position-only representation and the resulting IK twist ambiguity for mesh animation (quantified in Sec. I), the missing global root trajectory, the absence of contact metrics, the fixed number of functional groups, and the valid-rig assumption. The scope of supported skeletons is stated in Sec. 3, and Sec. G discusses the pose-anchor design.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper makes no formal theoretical claims; all contributions are empirical (architecture, training objective, and experimental evaluation).

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Sec. A provides complete architecture hyperparameters (Table 5), all loss weights (Table 6), optimizer settings, training schedule, and Stage 2 MaskGIT parameters. Baseline adaptation procedures are described in Sec. D. The data curation pipeline is detailed in Sec. F.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: Code, the curated OXL character split, the functional-group annotations, and the captions will be released on the project page. The underlying datasets (HumanML3D, Truebones Zoo, Objaverse-XL) and the Mixamo test set of Aberman et al. [1] are publicly accessible.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Training hyperparameters and architecture details are in Sec. A. Data splits and preprocessing are described in Sec. 4 and Sec. F. Evaluation metrics are defined in Sec. C.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: Text-to-motion generation metrics (FID, R-Precision, MM-Dist) on HumanML3D are computed over multiple evaluation runs following the MoMask [12] protocol, and the mean±standard deviation is reported in Table 2. Reconstruction and cross-topology metrics (MPJPE, RT-MPJPE) are deterministic over fixed held-out test sets and require no error bars.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Sec. A and Table 5 report that Stage 1 is trained on 4× GH200 96 GB GPUs for 100 epochs (∼96 h wall-clock); Stage 2 uses the same hardware for 300 epochs (∼12 h wall-clock).

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research uses publicly licensed motion-capture and 3D-asset datasets, involves no human subjects, and raises no privacy or fairness concerns. We have reviewed the NeurIPS Code of Ethics and confirm compliance.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: SAMoR lowers the barrier to 3D character animation for non-expert creators by enabling cross-topology motion transfer and text-conditioned generation, with positive impact on accessibility in games, film, and virtual environments. A potential negative use is synthetic motion generation for deepfake character animations; however, the system requires rigged 3D assets as input and does not produce photorealistic human imagery, substantially limiting this risk.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: SAMoR is a motion representation model trained on animation data. It does not generate photorealistic images or text and does not pose a high risk for misuse; dedicated safeguards beyond standard responsible release practices are not required.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets are cited with their original publications: HumanML3D [13], Truebones Zoo [37], Objaverse-XL [7], and the Mixamo test set of Aberman et al. [1]. HumanML3D is built from AMASS [25] and HumanAct12 [14], and AMASS is used under its non-commercial research license. Objaverse-XL is licensed under ODC-By v1.0 as a whole, and each object has its own license. Truebones Zoo is a commercial royalty-free library, and we do not redistribute its motions. The Mixamo test set is used under the terms of its provider and for evaluation only.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: We introduce a curated OXL motion subset (71,174 animated characters across six broad character types) with per-clip text captions and functional-group annotations. Dataset composition, curation criteria, and annotation procedure are documented in Sec. F. The split, annotations, and captions will be released alongside the code. OXL assets are referenced by their Objaverse-XL identifiers and are not redistributed.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The paper does not involve crowdsourcing or human subjects. All evaluation is performed on pre-existing motion-capture and 3D-asset datasets.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: No human subjects are involved in this research; IRB approval is not required.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: We use Gemini Flash as an automated text-captioning component for OXL motion clips (Secs. 3.3 and F). This is a core part of the data pipeline enabling text-conditioned generation on cross-topology characters and is disclosed in the main paper.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.