RETARGETING MOTIONS TO DIVERSE SKELETONS VIA LEARNABLE FLATTENING

Anonymous authors
Paper under double-blind review

ABSTRACT

Cross-structural motion retargeting aims to transfer motion between different skeletal topologies. Despite recent progress, existing state-of-the-art models struggle with reliability in zero-shot settings, i.e. skeletons with different topologies which were unseen during training, and recent Transformer-based attempts have failed to outperform specialized geometric methods. We bridge this gap with a Transformer Autoencoder that learns a topology- and translation-invariant latent space. Our core contribution is a learnable flattening of skeletal graphs that captures both local dependencies and global structure. Unlike the standard transformer architecture, which adds positional information to token content, we integrate graph-based positional encodings multiplicatively, a design choice that follows directly from our flattening formulation. The resulting model handles diverse skeletal topologies within a single unified architecture and trains in a fully unsupervised manner, requiring no paired retargeting data. Ablation studies show, that the graph encodings, multiplicative formulation, and Transformer backbone is critical for the performance. In zero-shot evaluations, our method reduces global joint position error by $43 - 47\%$ over current benchmarks. A user study ($n = 37$), including expert animators, further ranks our approach highest in motion alignment and physical plausibility ($p < 0.05$). These results demonstrate that our model design is key to making transformer architectures effective for motion retargeting, outperforming existing approaches.

1 INTRODUCTION

Character animation and motion synthesis are longstanding challenges in computer graphics and computer vision, with broad applications in entertainment, virtual reality, biomechanics, and human-computer interaction (Bruderlin & Williams, 1995; Gleicher, 1998; Holden et al., 2016; Loper et al., 2015). Among these, particularly motion retargeting—the process of transferring motion between characters with differing skeletal topologies—is challenging. Traditional approaches often rely on manual efforts by skilled animators or require specialized algorithms tailored for specific pairs of skeletons, limiting scalability and generalization.

Deep learning approaches have recently shown promising results in motion synthesis and representation (Pavllo et al., 2019; Petrovich et al., 2021; Tevet et al., 2023; Yuan et al., 2023). However, most existing methods operate on fixed skeletal topologies, limiting their generalizability across diverse character structures. The underlying challenge lies in the representation of skeletal motion data, which inherently combines local joint rotations with global hierarchical structure. When mapped to a latent space, these representations often become tied to specific skeletal configurations, making cross-skeleton transfer difficult.

In this paper, we introduce a Transformer Autoencoder for motion retargeting that learns a skeleton topology- and translation-invariant latent space, enabling seamless motion transfer across characters with vastly different skeletal structures (Figure 1). The central architectural contribution is a learnable flattening of skeletal graphs, which reinterprets the standard joint-concatenation operation as a combination of learnable tiling and masking. This formulation allows the model to generalize to unseen skeletal topologies and directly motivates our integration of graph-based positional encodings via a multiplicative scheme, a design choice our ablations confirm is critical for retargeting quality.

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

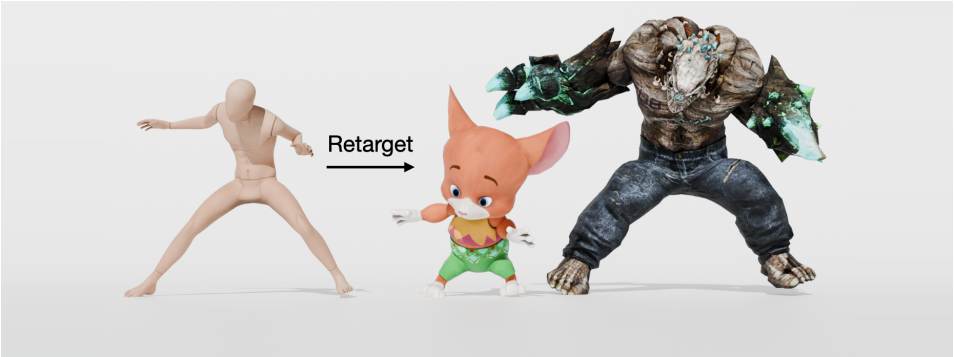


Figure 1: Our method retargets a source motion (left) to characters with vastly different skeletal topologies via learnable flattening of skeletal graphs in a fully zero-shot setting.

The model is trained end-to-end in a fully unsupervised, cycle-consistent manner, requiring no paired retargeting data, and incorporates novel data augmentation strategies that our ablations confirm are critical for zero-shot generalization. The resulting framework outperforms all existing methods both quantitatively and qualitatively, substantially advancing the state of the art in motion retargeting across unseen skeletal topologies. Our main contributions are:

- A Transformer Autoencoder that retargets motion between arbitrary humanoid skeletons — with varying numbers of joints and edges — within a single unified model, trained without paired data, or textual joint information, encoding skeletal structure purely from graph topology and rest pose geometry.
- A learnable skeletal flattening with multiplicative graph-positional encodings that yields a topology- and translation-invariant latent space, directly enabling zero-shot generalization to unseen skeletons.
- State-of-the-art retargeting performance, reducing global joint position error by 43–47% over existing benchmarks and ranking highest in motion alignment and physical plausibility in a user study with expert animators ($n = 37$, $p < 0.05$).

2 RELATED WORK

Our work builds upon and extends research in several areas, including character animation, motion retargeting and deep representation learning for skeletal motion. We review the most relevant contributions in each of these domains.

2.1 MOTION RETARGETING

Motion retargeting transfers motion between characters while preserving semantic meaning. Classical approaches formulate this as space-time optimization (Gleicher, 1998; Tak & Ko, 2005) or inverse kinematics (Lee & Shin, 1999; Choi & Ko, 2000), requiring extensive manual tuning. Recent deep learning methods are largely unsupervised due to the lack of paired datasets.

Villegas et al. (2018) use an RNN with cycle consistency and adversarial training, but perform poorly across different skeleton structures. Aberman et al. (2020) support arbitrary joint counts but need a separate model per skeleton pair. Lee et al. (2023) take a supervised approach, relying on MotionBuilder (Autodesk, 2021) to generate proxy ground truth. Liu et al. (2026) propose a transformer-based unsupervised single model but rely on textual joint name embeddings and still fail to outperform existing methods, indicating that architecture alone is insufficient without careful design. Chen et al. (2025) offer a training-free alternative via patch-based motion matching, but require explicit bone correspondences and a target motion database at inference time.

Our work is the first to make transformers effective for motion retargeting via a learnable flattening of skeletal graphs, outperforming all existing methods with a single unsupervised zero-shot model.

2.2 Deep Motion Representations

Learning compact latent motion representations has proven highly effective across tasks. Holden et al. (2015) pioneered convolutional autoencoders for motion, Aberman et al. (2019) learned a skeleton-agnostic latent space for 2D retargeting, Athanasiou et al. (2022) encode spatio-temporal motion with text for conditional generation, and Starke et al. (2022) capture motion in periodic feature embeddings.

MotionPuzzle (2022) employs cycle-consistency and reconstruction objectives on unpaired data for per-body-part style transfer, but within a fixed topology. Gat et al. (2025) integrate graph structure additively into transformer attention maps. In contrast, our multiplicative positional encoding is derived from first principles and our ablations confirm it is critical for zero-shot generalization.

Despite large-scale successes in NLP and vision, motion data remains fragmented across incompatible parametrizations. AMASS (Mahmood et al., 2019) unified datasets into SMPL (Loper et al., 2015; Pavlakos et al., 2019), and learned motion priors (Rempe et al., 2021; Chen et al., 2022; Raab et al., 2024; Yuan et al., 2023) offer unified representations, but all remain limited to fixed humanoid topologies. SAME (2023) handles varying topologies in a single model but requires ground-truth retargeting pairs.

In contrast, our framework requires neither, and learns a skeleton-agnostic pose component alongside a separate root trajectory component.

3 METHODS

In order to enable retargeting across diverse skeletal topologies, we must decouple the motion representation from the source skeleton's joint count and structure. Standard flattening ($\mathbb{R}^{J\times d} \to \mathbb{R}^{Jd}$) creates representations that scale linearly with J, preventing generalization to new skeletons. We address this by decomposing the flattening operation, revealing it is implicitly a *multiplicative positional encoding* process, and replacing its fixed operations with learned neural networks to obtain fixed-dimensional, topology-agnostic latent representations.

3.1 Data Representation

We first formalize the skeletal motion inputs used in our framework. The skeleton is defined by its rest pose $\mathcal{S} = \{p_{\text{rest}}, \mathcal{A}\}$. Here, $p_{\text{rest}} \in \mathbb{R}^{J\times 3}$ encodes the rest pose geometry based on fixed bone offsets in a canonical T-pose configuration (with the y-axis defined as up), while $\mathcal{A} \in \{0,1\}^{J\times J}$ is the adjacency matrix defining kinematic connectivity. The motion is modeled as a temporal sequence $\mathbf{X} = \{q_t, p_t, p_{t-1}, v_t, r_t\}_{t=1}^{T}$. Here, $q_t \in \mathbb{R}^{J\times 6}$ encodes joint rotations using the continuous 6D representation (Zhou et al., 2019), $p_t \in \mathbb{R}^{J\times 3}$ denotes root-centered joint positions, v_t represents joint velocities, and $r_t \in \mathbb{R}^3$ specifies the global root trajectory.

3.2 Flattening Representation Learning

Decomposing Flattening. Consider a pose on a skeleton with J joints, each represented by d features, forming a data matrix $\mathcal{X} \in \mathbb{R}^{J\times d}$. Standard flattening concatenates the rows of $\mathcal{X}$ into a single vector $\bar{q} = \text{vec}(\mathcal{X}^T) \in \mathbb{R}^{Jd}$. We can express this operation as the interaction between two matrices: a projection matrix $\mathbf{T}$ and a positional mask $\mathbf{M}$.

$$\mathbf{T} = \big[\mathbf{I}_d, \mathbf{I}_d, \ldots, \mathbf{I}_d\big], \quad \mathbf{M} = \mathbf{I}_J \otimes \mathbf{1}_d^T \tag{1}$$

The projection matrix $\mathbf{T} \in \mathbb{R}^{d\times Jd}$ broadcasts the content features of a joint across all possible output positions and $\otimes$ denotes the Kronecker product. The positional mask $\mathbf{M} \in \mathbb{R}^{J\times Jd}$ is a sparse binary matrix that "activates" the specific slot in the flattened vector corresponding to the j-th joint. The flattening and unflattening operations can then be rewritten as:

$$\bar{q} = \sum_{j=1}^{J} \Big[(\mathcal{X}_j \mathbf{T}) \odot \mathbf{M}_j\Big], \tag{2}$$

$$\mathcal{X} = ((\mathbf{1}_J \bar{q}) \odot \mathbf{M})\mathbf{T}^T, \tag{3}$$

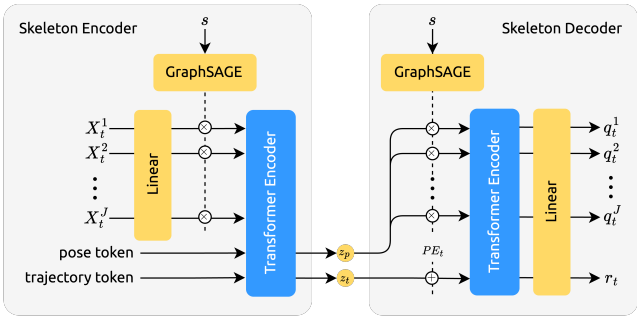


Figure 2: Schematic of the our model architecture. The Skeleton Position Encoding uses GraphSAGE to produce a spatial mask from the rest pose $\mathcal{S}$. The Encoder "flattens" skeletal data into latent pose and trajectory tokens, while the Decoder "unflattens" them to recover joint rotations.

where $\odot$ denotes the Hadamard (element-wise) product and the subscript j denotes the j-th row of the corresponding matrix.

The (un)flattening operation functions as a *structural gating mechanism*, mathematically analogous to attention, where a map $\mathbf{A}$ applies to values $\mathcal{V}$. In our case, $\mathbf{M}_j$ acts as a static attention map $\mathbf{A}$ applied *multiplicatively* to the values $\mathcal{V}$ ($\mathbf{X}_j\mathbf{T}$ or $\bar{q}$). Since $\mathbf{M}_j$ encodes joint identity via the position of its nonzero entries, this acts as a positional encoder. Unlike standard positional encoding which typically sum position and content, here the content ($\mathcal{X}_j\mathbf{T}$) is modulated by the structural position ($\mathbf{M}_j$) *via multiplication*.

Learnable Flattening. To achieve a topology-agnostic representation, we replace these fixed matrices with learned, continuous functions. Our encoder implements the learnable flattening operation:

$$z_p = \mathsf{Agg}_{j\in J}\Big[\mathsf{T}_{\text{enc}}(\mathcal{X}_j) \odot \mathsf{M}(\mathcal{S})_j\Big], \tag{4}$$

where $\mathsf{T}_{\text{enc}} : \mathbb{R}^d \to \mathbb{R}^D$ is a learned projection that maps joint features to a fixed latent dimension D, independent of J. $\mathsf{M}(\mathcal{S}) \in \mathbb{R}^{J\times D}$ is a learned positional embedding derived dynamically from the skeleton topology $\mathcal{S}$.

This formulation preserves the multiplicative interaction crucial for spatial structure but decouples the representation size from the joint count. The decoder reverses this operation ("unflattening") to reconstruct the specific skeletal motion:

$$\hat{\mathcal{X}} = \mathsf{T}_{\text{dec}}\Big(z_p \odot \mathsf{M}(\mathcal{S})\Big), \tag{5}$$

where $\mathsf{T}_{\text{dec}} : \mathbb{R}^D \to \mathbb{R}^d$ reconstructs per-joint features from the latent space.

3.3 Model Architecture

Our framework leverages a transformer-based autoencoder that implements the learned flattening and unflattening operations. It integrates three key modules: (1) a **Skeleton Position Encoding**, which learns the spatial mask $\mathbf{M}(\mathcal{S})$; (2) a **Skeleton Encoder**, which performs the learnable flattening (Eq. 4) to produce latent tokens; and (3) a **Skeleton Decoder**, which performs the unflattening (Eq. 5) to reconstruct the motion (Figure 2). For architecture details please refer to the Appendix Section A.

Skeleton Position Encoding. $\mathsf{M} : \mathcal{S} \to \mathbb{R}^{J\times D}$ generates learned spatial masks from the rest pose $\mathcal{S}$ using an undirected GraphSAGE (Hamilton et al., 2017) network, producing a spatially-aware embedding $\mathsf{M}(\mathcal{S})_j$ for topology-aware encoding and decoding.

Skeleton Encoder. Implements the *learnable flattening* in Equation 4. First a linear projection layer $\mathsf{T}_{\text{enc}} : \mathbb{R}^d \to \mathbb{R}^D$ acts as the projection operator, mapping each joint's features into a shared D-dimensional space, where it is element-wise multiplied by the learned spatial mask $\mathsf{M}(\mathcal{S})$. A

Transformer Agg then aggregates the joint features into a fixed-dimensional pose token $z_p \in \mathbb{R}^D$, while a separate trajectory token z_t encodes global translation independently of pose.

Skeleton Decoder. Implements the *learnable unflattening* in Equation 5. The spatial mask $\mathsf{M}(\mathcal{S})$ is reapplied to the pose token z_p, and a Transformer decoder combines the resulting features with the trajectory token z_t. Finally, a linear layer maps the decoded features back to the original input dimensions: 6 for joint rotations and 3 for the root trajectory. Together, the Transformer and linear layer constitute the learned decoding operator T_{dec}. We use forward kinematics to obtain world-space joint positions.

3.4 Training

In the absence of explicit ground truth, our framework uses cycle consistency (Zhu et al., 2017), an adversarial loss (Goodfellow et al., 2014), and an augmentation consistency strategy (Figure 3). Three augmentations prime the autoencoder to learn meaningful invariances and execute reliable retargeting: global translation $T(\cdot)$, bone scaling $S(\cdot)$, and skeleton augmentation $R(\cdot)$ (per-joint scaling and joint removal).

The losses are defined as follows: $\mathcal{L}_{\text{rec}}$ combines joint-position, rotation, root-trajectory, and root-child position losses (the latter introduced to preserve joints directly connected to the root). $\mathcal{L}_{\text{temp}}$ encourages temporal smoothness, $\mathcal{L}_{\text{cntct}}$ reduces foot sliding and ground penetration, $\mathcal{L}_{\text{ee}}$ preserves normalized end-effector velocities, $\mathcal{L}_{\text{latent}}$ enforces pose consistency, and $\mathcal{L}_{\text{adv}}$ encourages realistic motions. Except for the root-child position loss, these follow prior work (Aberman et al., 2020; Lee et al., 2023). Details are provided in Appendix A.3.

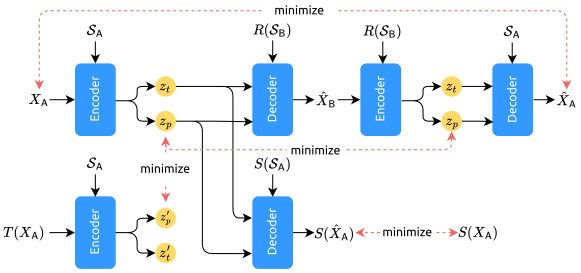


Figure 3: Schematic of the training framework.

Two complementary schemes combine these losses: *augmentation consistency* and *cycle consistency*. Given an input motion $\mathcal{X}_A$ associated with a source skeleton $\mathcal{S}_A$. Denote the encoder and decoder by $\mathcal{E}$ and $\mathcal{D}$, respectively. The augmented reconstruction $S(\hat{\mathcal{X}}_A)$ is computed as encoding the input motion $\mathcal{X}_A$ with skeleton $\mathcal{S}_A$ and then decoding into the augmented skeleton $S(\mathcal{S}_A)$. Cycle consistency is enforced by retargeting the input motion $\mathcal{X}_A$ with skeleton $\mathcal{S}_A$ to a target skeleton $R(\mathcal{S}_B)$ obtaining the retargeted $R(\hat{\mathcal{X}}_B)$ and then mapping it back to the original skeleton $\mathcal{S}_A$ obtaining $\hat{\mathcal{X}}_A$. For $A \in \{S, T, R\}$, let $\mathcal{X}^A = A(\mathcal{X})$:

$$\mathcal{L}_{\text{aug. cons.}} = \mathcal{L}_{\text{rec}}(\mathcal{X}_A^S, \hat{\mathcal{X}}_A^S) + \mathcal{L}_{\text{cntct}}(\mathcal{X}_A^S, \hat{\mathcal{X}}_A^S) + \mathcal{L}_{\text{temp}}(\mathcal{X}_A^S, \hat{\mathcal{X}}_A^S) + \mathcal{L}_{\text{ee}}(\mathcal{X}_A, \hat{\mathcal{X}}_A^S) + \mathcal{L}_{\text{latent}}(\mathcal{X}_A, \mathcal{X}_A^T)$$
$$\mathcal{L}_{\text{cyc. cons.}} = \mathcal{L}_{\text{rec}}(\mathcal{X}_A, \hat{\mathcal{X}}_A) + \mathcal{L}_{\text{cntct}}(\mathcal{X}_A, \hat{\mathcal{X}}_A) + \mathcal{L}_{\text{latent}}(\mathcal{X}_A, \hat{\mathcal{X}}_B^R) + \mathcal{L}_{\text{adv}}(\mathcal{X}_A, \hat{\mathcal{X}}_B^R)$$

The total loss is $\mathcal{L}_{\text{aug. cons.}} + \mathcal{L}_{\text{cyc. cons.}}$.

4 Experiments

We assess three capabilities: zero-shot reconstruction, latent space invariance, and unsupervised retargeting fidelity. We compare against SAME (Lee et al., 2023), which handles arbitrary topologies in a single model via a delta-based trajectory approach, Aberman et al. (2020), the standard for unsupervised retargeting with topology-specific models. and PALUM (Liu et al., 2026), an unsupervised transformer-based approach that also handles all joint topologies in a single model.

We train on LaFAN1 (Harvey et al., 2020), ACCAD (acc), TotalCapture (Trumble et al., 2017), PFNN (Holden et al., 2017), SFU (Simon Fraser University, 2011), and CMU (Carnegie Mellon University, 2003). We evaluate on Bandai-Namco (Kobayashi et al., 2023) and 100Style (Mason

et al., 2022) as zero-shot benchmarks, and Mixamo (Adobe, 2021) for intra- and cross-structure retargeting comparisons. Further details are in Appendix A.5.

4.1 RECONSTRUCTION

We evaluate zero-shot reconstruction on Mixamo, Bandai-Namco (Kobayashi et al., 2023), and 100Style (Mason et al., 2022) against SAME (Lee et al., 2023), the only prior work handling diverse unseen skeletons without retraining and publicly available code. We report Joint Position (JP) and Joint Rotation (JR) errors, Root Trajectory (RT) error, Foot Sliding (FS), and Ground Penetration (GP), following the evaluation protocol of Lee et al. (2023). Critically, we evaluate on *entire animation sequences* instead of SAME's 1-second clips, to allow quantitative assessment of long-term stability.

Table 1: Reconstruction Results on unseen testing datasets. We compare Joint Position (JP), Joint Rotation (JR), Root Trajectory (RT), Foot Sliding (FS), and Ground Penetration (GP) errors. **Bold** indicates best.

Method	**JP** [cm]	**JR** [rad]	**RT** [cm]	**FS**	**GP** [cm]
Bandai-Namco (Unseen)					
SAME	12.40	0.34	8.58	0.08	-0.20
Ours	**2.63**	**0.19**	**1.65**	**0.05**	**-0.01**
Mixamo (Unseen)					
SAME	10.07	0.31	4.47	**0.02**	**-0.02**
Ours	**4.08**	**0.22**	**0.84**	0.03	**-0.02**
100 Style (Unseen)					
SAME	104.66	0.22	101.78	**0.01**	**-0.00**
Ours	**2.14**	**0.16**	**1.85**	0.02	**-0.00**

Our method generalizes better to unseen datasets and outperforms SAME across all datasets (Table 1). On Bandai-Namco, Joint Position error drops by 78% (12.40→2.63 cm), with lower Foot Sliding (0.08→0.05) and Ground Penetration (-0.20→-0.01). Particularly, SAME requires ground contact labels at inference while we use them only during training. On 100Style, SAME suffers catastrophic drift (JP:104.66→2.14 cm, RT: 101.78→1.85 cm) due to delta-based root prediction accumulating errors over long sequences (Figure 7, Appendix C.2). Unlike SAME, we predict the global root trajectory directly like Aberman et al. (2020), eliminating drift entirely.

4.2 LATENT SPACE INVARIANCES

4.2.1 SKELETON INVARIANCE

Our latent space is designed to be topology-agnostic, ensuring that identical motions map to the same representation regardless of the skeleton. To verify this, we project latent representations of distinct Mixamo characters (Mousey, Goblin, Vampire, etc.) into a 2D PCA space, following the approach of Lee et al. (2023). The tightly overlapping clusters show that embeddings from different skeletons (represented by different colors) align closely when performing the same action (e.g., "Walking," "Idle"), confirming that our model abstracts motion semantics while discarding skeleton-specific structure (Figure 4).

4.2.2 TRANSLATION INVARIANCE

Our model disentangles motion into two distinct latent spaces — pose (z_p) and root trajectory (z_t) — with z_p designed to be invariant to global translation. To validate this, we compare pose embeddings of identical motions before and after applying large-scale global translations (10^6 cm) along the x-, z-, and xz-axes, measuring Cosine Similarity between original and translated representations (Figure 5).

Our method achieves near-perfect invariance (> 0.99), while SAME also reaches perfect invariance (1.0) as expected from its delta-based representation. In contrast, Aberman et al. (2020) shows significantly higher sensitivity, with scores dropping below 0.8 and exhibiting high variance, confirming that it entangles global position with pose, a limitation our explicit disentanglement and augmentation strategy avoids.

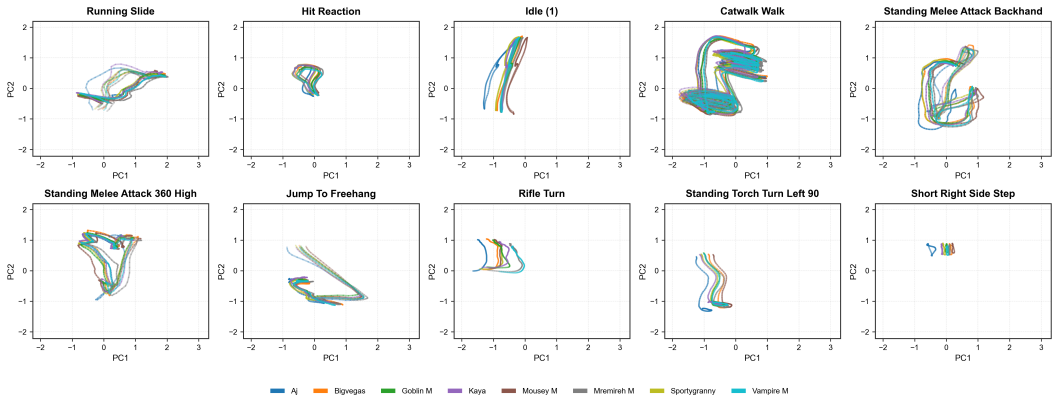


Figure 4: Skeleton invariance latent spaces of the model: 2D Principal Component Analysis projected latent space of the model for different skeletons performing semantically identical motions for different motions. The PCA space is shared across all sequences.

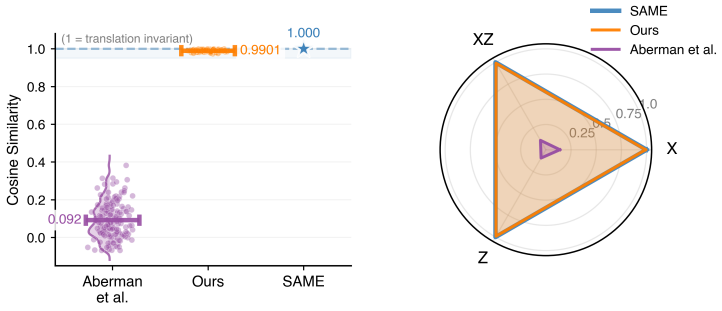


Figure 5: Translation invariance of our pose latent space. Left shows distributions of cosine similarities between latents of untranslated and translated motions. SAME is invariant by definition, so all the values are 1. Right shows the mean cosine similarities across translations in the x-, z- and xz-axes.

4.2.3 Motion Classification

In order to measure that the semantic of the motion was captured across different skeletons, we ran a motion classification task based on our pose latent space. A topology-invariant latent space should cluster semantically similar motions together regardless of the skeletons they were performed on, and therefore the classification accuracy tells something about the quality of the latent space. Following the evaluation protocol of Lee et al. (2023), we achieve 59/60 correctly classified motions compared to 57/60 reported by SAME. The only difference from their setup is that our latent space has 128 dimensions compared to 32 in SAME.

4.3 Retargeting

Motion retargeting transfers a motion sequence from one character to another with a different skeletal structure, while preserving the intent and style of the original movement. We evaluate our method on this task both qualitatively, through a user study, and quantitatively, using three complementary metrics across two structural settings.

4.3.1 Quantitative Measurement

To assess quantitative performance, we evaluate on two sub-tasks: *Intra-structure* (Intra), where source and target share identical skeletal topologies, and the more challenging *Cross-structure* (Cross), where topologies differ significantly. We use three metrics: **Global Joint Position (GJP)**

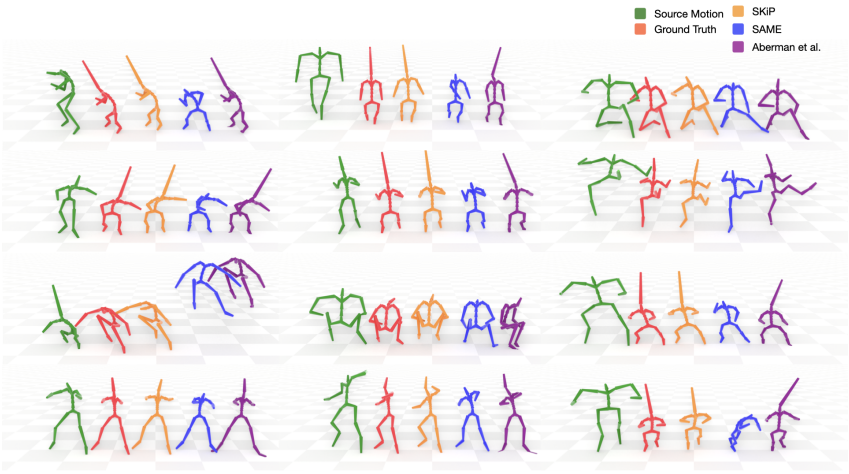


Figure 6: Retargeting across diverse skeletons: The green skeletons are the source motions, red are ground truth retargets from the Mixamo dataset, the orange are retargets from our method, blue are from SAME and purple from Aberman et al. (2020).

error (Aberman et al., 2020), the mean Euclidean distance between predicted and ground-truth joint positions in global space, normalized by character height and scaled by 10^3; **Jerk**, the third derivative of joint positions, where values closer to ground truth indicate natural dynamics; and **Procrustes Aligned Root Trajectory (PART)** error, which measures how well the predicted root trajectory preserves the original motion shape, independent of absolute alignment.

Despite not being trained on Mixamo, our method achieves state-of-the-art GJP across both Intra and Cross tasks, outperforming the next best method by at least 47% and 43% respectively (Table 2). Our Jerk score (0.72) falls below ground truth ($\approx$ 1.3), reflecting a mild smoothing effect rather than a perceptual deficiency — adding minimal noise ($\sigma = 0.005$) sufficient to produce visible jitter raises Jerk to 8.61, confirming the gap lies well below the threshold of visual perceptibility. PART scores further show that methods using global root representations (ours and Aberman et al. (2020)) preserve trajectory shape considerably better than delta-based methods (SAME).

Table 2: Animation retargeting on Mixamo. **Bold**: best; underline: second best. GT Jerk: 1.28 (Intra), 1.32 (Cross). †: not trained on Mixamo motions. ‡: not trained on Mixamo skeletons.

	Intra			Cross		
Method	GJP	Jerk	PART	GJP	Jerk	PART
Copy rotations	8.86	-	-	N/A	N/A	N/A
Villegas '18	6.24	-	-	243	-	-
Lim '19	5.72	-	-	N/A	N/A	N/A
Aberman	2.76	-	-	2.25	**1.15**	0.08
SAME†	2.91	1.87	0.18	2.47	1.94	0.17
PALUM†	2.72	-	-	5.67	-	-
Ours†‡	**1.45**	**0.72**	**0.03**	**1.28**	0.72	**0.05**

4.3.2 Qualitative Measurement

To corroborate our quantitative results, we conducted a user study in which 37 participants, including 8 professional animators, evaluated retargeted animations from our method, SAME, Aberman et al. (2020), and ground truth. Participants viewed each animation individually in random order and rated it on two criteria, alignment to the source motion and perceptual quality, using a 1 (poor) to 5 (excellent) Likert scale. Ground truth serves as an upper bound.

While our method closes the gap to ground truth, a perceptual difference remains. A two-sided Mann-Whitney U test ($p < 0.05$) indicates that our method significantly outperforms SAME and Aberman et al. (2020) in quality, and SAME in alignment (Table 3). Qualitative comparisons in Figure 6, and 9 in Appendix C.4 further show coherent retargeting across diverse skeleton topologies.

Table 3: User study. **Bold**: best; * significant vs. ours ($p < 0.05$).

Method	Align. ↑	Qual. ↑
Aberman	4.00 ± 0.05	$3.49 \pm 0.06^*$
SAME	$3.42 \pm 0.06^*$	$3.69 \pm 0.06^*$
Ours	$\mathbf{4.06 \pm 0.05}$	$\mathbf{3.86 \pm 0.05}$
Ground truth	$4.42 \pm 0.04^*$	$4.39 \pm 0.04^*$

4.4 Ablation Study

We ablate the key architectural and training design choices of our method in Table 4. Replacing the Transformer with a GAT (Veličković et al., 2018) substantially degrades reconstruction and retargeting performance, resulting in very large GJP errors ($891.96/927.38$ for Intra/Cross). The extreme magnitude of these errors is primarily driven by inaccurate global root-trajectory prediction (> 233 cm), although local motion reconstruction also deteriorates. The training curves reveal that the GAT overfits to the training skeletons, failing to generalize to the joint modeling of local pose and global trajectory required for retargeting. Removing positional encoding (PE) similarly causes the model to fail, while replacing our multiplicative PE with an additive formulation increases GJP by 42% on Intra ($1.45 \rightarrow 2.06$) and 57% on Cross ($1.28 \rightarrow 2.01$). Finally, removing our augmentations substantially degrades generalization, increasing GJP to $3.92/2.66$. Together, these results demonstrate the importance of the Transformer architecture, multiplicative structural encoding, and augmentation strategy. Ablations of the individual training losses are provided in Table 9 in Appendix C.1.

Table 4: Ablation Study on the Mixamo dataset. We evaluate the impact of removing key components on Intra- and Cross-retargeting performance. **Bold** indicates best; underline indicates second best.

	Intra			Cross		
Method	GJP	Jerk	PART	GJP	Jerk	PART
Ours Full	**1.45**	**0.72**	**0.03**	**1.28**	**0.72**	**0.05**
with GAT	891.96	30.07	0.38	927.38	31.43	0.36
w/o PE	62.98	0.01	0.57	62.98	0.01	0.60
additive PE	<u>2.06</u>	<u>0.70</u>	0.05	<u>2.01</u>	**0.72**	<u>0.06</u>
w/o aug.	3.92	0.63	<u>0.04</u>	2.66	<u>0.64</u>	**0.05**

5 Summary, Limitations, and Future Work

In this work, we introduce a unified, unsupervised Transformer Autoencoder for motion retargeting across arbitrary skeletal topologies. While prior transformer-based attempts such as PALUM fail when retargeting between characters with differing skeletal structures, our learnable skeletal graph flattening with multiplicative positional encodings is what makes transformers effective for this task. Unlike Aberman et al. (2020) and SAME, our single model requires neither topology-specific components nor paired training data, and avoids delta-based error accumulation by predicting the global root trajectory directly. Evaluated entirely in a zero-shot setting, our method outperforms all existing state-of-the-art methods both quantitatively and qualitatively.

Despite these advancements, several limitations remain. Our model assumes a T-pose rest configuration, requiring preprocessing of input motion data. Additionally, our loss functions focus on skeletal kinematics and do not account for surface geometry, which can lead to self-penetration for characters with voluminous meshes or misalignment between the skeleton and the mesh, such as shoulders dropping when the mesh is applied. Our model is also designed for human-like skeletal structures and struggles with characters that have a large number of joints from non-standard body parts, such as capes or other protruding elements. Future work could address these limitations by introducing mesh-aware constraints and extending the model to handle more diverse skeletal configurations, and explore leveraging the learned latent space for large-scale motion modelling across diverse datasets.

AI USE STATEMENT

In this work, we used generative AI tools for none of the tasks with required disclosure. We have **not used** generative AI tools for help to develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, clean and reformat dataset, support qualitative and thematic data analysis, or interpret results, and generating synthetic data sets, or assist with translation are not applicable to this work. We **used** generative AI tools for editing the research paper to improve readability and formatting references. We have reviewed all AI-assisted work. All AI-assisted edits were manually reviewed by the authors for correctness and consistency, and all generated BibTeX entries were verified against the original sources. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of our work, we have provided detailed information on our method and experimental setups. We will discuss the respective details below.

Our method and architecture In addition to the details presented in the main text (Section 3)(Sec. 3), we provided detailed descriptions of the model architecture in (Section A.1), the hyperparameter settings (Section A.2), and the loss functions and their weightings. Furthermore, we provided detailed descriptions of all data augmentation strategies used in our work (Section B), including skeleton augmentation (Section B.3), bone scaling (Section B.2), global translation (Section B.4), and rest pose augmentation (Section B.1).

Experiments In addition to the details provided in the main text (Section 4), we provided extensive additional information (Section A). Specifically, we detail the data preparation and evaluation protocols (Section A.5), and describe the training protocol and hardware setup (Section A.6).

Code Availability The complete implementation, along with pretrained models and detailed instructions for reproducing our results, is publicly available at `https://anonymous.4open.science/r/retarget-2B74/README.md`. The repository includes guidelines for setting up the environment and executing the training pipeline.

REFERENCES

Accad motion capture database. `https://accad.osu.edu/research/mocap/mocap_data.html`. Accessed: 2025-05-19.

Kfir Aberman, Rundi Wu, Dani Lischinski, Baoquan Chen, and Daniel Cohen-Or. Learning character-agnostic motion for motion retargeting in 2d. *ACM Transactions on Graphics (TOG)*, 38(4):75, 2019.

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 (TOG)*, 39(4):120, 2020.

Adobe. Mixamo animation dataset. `https://www.mixamo.com`, 2021.

Nikos Athanasiou, Mathis Petrovich, Michael J Black, and Gül Varol. Teach: Temporal action composition for 3d humans. In *Proceedings of the International Conference on 3D Vision (3DV)*. IEEE, 2022. URL `https://teach.is.tue.mpg.de`.

Autodesk. Motionbuilder - a 3d character animation software, 2021. URL `https://www.autodesk.com/products/motionbuilder/overview`.

Armin Bruderlin and Lance Williams. Motion signal processing. In *Proceedings of SIGGRAPH 1995*, pp. 97–104. ACM, 1995.

Carnegie Mellon University. Cmu graphics lab motion capture database. `http://mocap.cs.cmu.edu/`, 2003. Accessed: 2025-05-19.

Ling-Hao Chen, Yuhong Zhang, Zixin Yin, Zhiyang Dou, Xin Chen, Jingbo Wang, Taku Komura, and Lei Zhang. Motion2motion: Cross-topology motion transfer with sparse correspondence. *ACM SIGGRAPH Asia 2025 Conference Proceedings*, 2025.

Xin Chen, Zhuo Su, Lingbo Yang, Pei Cheng, Lan Xu, Bin Fu, and Gang Yu. Learning variational motion prior for video-based motion capture. *arXiv preprint arXiv:2210.15134*, 2022. URL https://arxiv.org/abs/2210.15134.

Kwang-Jin Choi and Hyeong-Seok Ko. Online motion retargetting. *Journal of Visualization and Computer Animation*, 11(5):223–235, 2000.

Inbar Gat, Sigal Raab, Guy Tevet, Yuval Reshef, Amit H. Bermano, and Daniel Cohen-Or. Anytop: Character animation diffusion with any topology. In *Proceedings of the ACM SIGGRAPH Conference*, 2025. URL https://anytop2025.github.io/Anytop-page/.

Michael Gleicher. Retargetting motion to new characters. *Proceedings of SIGGRAPH*, pp. 33–42, 1998.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 2672–2680, 2014. URL https://arxiv.org/abs/1406.2661.

William L. Hamilton, Rex Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS)*, pp. 1024–1034. Curran Associates, Inc., 2017. URL https://arxiv.org/abs/1706.02216.

Félix G. Harvey, Mike Yurick, Derek Nowrouzezahrai, and Christopher Pal. Robust motion inbetweening. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)*, 39(4), 2020.

Daniel Holden, Jun Saito, Taku Komura, and Thomas Joyce. Learning motion manifolds with convolutional autoencoders. In *SIGGRAPH Asia 2015 Technical Briefs*, pp. 1–4. ACM, 2015.

Daniel Holden, Jun Saito, and Taku Komura. A deep learning framework for character motion synthesis and editing. In *ACM Transactions on Graphics (SIGGRAPH 2016)*, volume 35, pp. 138:1–138:11, 2016.

Daniel Holden, Taku Komura, and Jun Saito. Phase-functioned neural networks for character control. *ACM Transactions on Graphics (TOG)*, 36(4):42:1–42:13, 2017.

Deok-Kyeong Jang, Soomin Park, and Sung-Hee Lee. Motion puzzle: Arbitrary motion style transfer by body part. *ACM Transactions on Graphics (TOG)*, 41(3):1–16, 2022.

Makito Kobayashi, Chen-Chieh Liao, Keito Inoue, Sentaro Yojima, and Masafumi Takahashi. Motion capture dataset for practical use of ai-based motion editing and stylization, 2023.

Jehee Lee and Sung Yong Shin. A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*, pp. 39–48. ACM Press/Addison-Wesley Publishing Co., 1999.

Sunmin Lee, Taeho Kang, Jungnam Park, Jehee Lee, and Jungdam Won. Same: Skeleton-agnostic motion embedding for character animation. In *ACM SIGGRAPH ASIA 2023 Conference Proceedings*, 2023. URL https://sunny-codes.github.io/assets/pdf/SAME.pdf.

Siqi Liu, Maoyu Wang, Bo Dai, and Cewu Lu. Palum: Part-based attention learning for unified motion retargeting. *arXiv preprint arXiv:2601.07272*, 2026.

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. Smpl: A skinned multi-person linear model. In *ACM Transactions on Graphics (TOG)*, volume 34, pp. 248. ACM, 2015. URL http://smpl.is.tue.mpg.de.

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)*, 2019. URL https://amass.is.tue.mpg.de.

Ian Mason, Sebastian Starke, and Taku Komura. Real-time style modelling of human locomotion via feature-wise transformations and local motion phases. *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, 5(1), may 2022.

Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3d hands, face, and body from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10975–10985, 2019.

Dario Pavllo, Christoph Feichtenhofer, David Grangier, and Michael Auli. 3d human pose estimation in video with temporal convolutions and semi-supervised training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7753–7762, 2019.

Maxim Petrovich, Michael J. Black, and Christoph Lassner. Action-conditioned 3d human motion synthesis with transformer vae. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 10981–10991, 2021.

Sigal Raab, Inbal Leibovitch, Guy Tevet, Moab Arar, Amit H. Bermano, and Daniel Cohen-Or. Single motion diffusion. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2302.05905.

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 (ICCV)*, pp. 11488–11499, 2021.

Simon Fraser University. Sfu motion capture dataset. http://mocap.cs.sfu.ca/, 2011. Accessed: 2025-05-19.

Sebastian Starke, Ian Mason, and Taku Komura. Deepphase: Periodic autoencoders for learning motion phase manifolds. *ACM Transactions on Graphics (TOG)*, 41(4):136:1–136:13, 2022.

Seyoon Tak and Hyeong-Seok Ko. A physically-based motion retargeting filter. *ACM Transactions on Graphics (TOG)*, 24(1):98–117, 2005. doi: 10.1145/1037957.1037963.

Guy Tevet, Sigal Raab, Brian Gordon, Yonatan Shafir, Daniel Cohen-Or, and Amit H. Bermano. Human motion diffusion model. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://arxiv.org/abs/2209.14916.

Matthew Trumble, Andrew Gilbert, Charles Malleson, Adrian Hilton, and John Collomosse. Total capture: 3d human pose estimation fusing video and inertial sensors. In *Proceedings of the British Machine Vision Conference (BMVC)*. BMVA Press, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Ruben Villegas, Jimei Yang, Duygu Ceylan, and Honglak Lee. Neural kinematic networks for unsupervised motion retargetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8639–8648, 2018.

Ye Yuan, Jiaming Song, Umar Iqbal, Arash Vahdat, and Jan Kautz. Physdiff: Physics-guided human motion diffusion model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 16010–16021, 2023. doi: 10.1109/ICCV51070.2023.01467.

Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5738–5746, 2019.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 2223–2232, 2017. URL https://arxiv.org/abs/1703.10593.

A ARCHITECTURE DETAILS

This section details our transformer autoencoder architecture, including model specifications, hyperparameter settings, loss function weighting, training protocol, and code availability.

A.1 MODEL ARCHITECTURE

Our autoencoder is composed of an encoder and a decoder. The encoder consists of a linear projection layer followed by a transformer encoder. The positional encoding in the encoder is learned using a GraphSAGE layer. Similarly, the decoder is built with a transformer encoder and a linear decoding layer, and it employs a GraphSAGE layer to learn its positional encoding. Furthermore it introduces two learnable parameters, used for the latent representation of the pose and root trajectory separately. Also for the discriminator network, which uses only the transformer encoder part, we introduce a learnable parameter, used for the classification. Table 5 summarizes these components.

Table 5: Model Architecture Specifications

Component	**Specification**
Encoder	
Linear Projection	Embedding dimension: 128
Transformer Encoder	Layers: 4, Attention Heads: 8 FeedForward: 512
Positional Encoding	GraphSAGE 2 layers
Decoder	
Transformer Encoder	Layers: 4, Attention Heads: 8 FeedForward: 512
Linear Decoding Dimensions	6 (For joints), 3 (For Trajectory)
Positional Encoding	GraphSAGE 2 layers
Adversarial Discriminator	
Linear Projection	Embedding dimension: 128
Transformer Encoder	Layers: 2, Attention Heads: 2, FeedForward: 64
Positional Encoding	GraphSAGE 2 layers
Linear Decoding Dimensions	1

A.2 HYPERPARAMETER SETTINGS

The training process employs the hyperparameters listed in Table 6. These settings were chosen based on extensive experimentation to ensure stable convergence and robust performance. It is to note, that for the adversarial discriminator optimizer, no scheduler was used.

Table 6: Hyperparameter Settings

Hyperparameter	**Value**
Batch Size	128×8
Weight Decay	1×10^{-4}
Learning Rate	1×10^{-3}
Learning Rate Scheduler	Cosine Annealing with Restarts
Optimizer	AdamW

A.3 Loss Function and Weighting

A.3.1 Reconstruction Loss

The primary goal is to accurately reconstruct a globally scaled input motion, while for the cycled back skeleton we reconstruct the unscaled input motion. This is achieved by minimizing a reconstruction loss that is composed of multiple components. First, the *position loss* minimizes the Euclidean distance between the ground truth positions p and the reconstructed positions $\hat{p}$ (6). Additionally, the *root children position loss* minimizes the Euclidean distance between the ground truth positions p and the reconstructed positions $\hat{p}$ of the children joints of the root joint(7). Next, the *rotation loss* employs a geodesic distance metric to quantify the angular discrepancy between the ground truth rotations q and the predicted rotations $\hat{q}$ (8). The *trajectory loss* minimizes the Euclidean distance between the ground truth trajectory r^t and its reconstruction $\hat{r}^t$(9).

$$\mathcal{L}_{\text{rec}}(\mathcal{X}^t, \hat{\mathcal{X}}^t) = \lambda_{\text{pos}} \frac{1}{J} \sum_{j=1}^{J} \|p_j^t - \hat{p}_j^t\| \tag{6}$$

$$+ \lambda_{\text{child}} \frac{1}{J} \sum_{j=1}^{J_{\text{child}}} \|p_j^t - \hat{p}_j^t\| \tag{7}$$

$$+ \lambda_{\text{rot}} \frac{1}{J} \sum_{j=1}^{J} d_g(q_j^t, \hat{q}_j^t) \tag{8}$$

$$+ \lambda_{\text{traj}} \|r^t - \hat{r}^t\| \tag{9}$$

A.3.2 Temporal Smoothing

Since our model reconstructs frame by frame individually, we regularize the temporal derivatives of the motion by introducing the *velocity loss*, which minimizes the Euclidean distance between the reconstructed velocity $\hat{v}_t$ and the ground truth velocity v_t (10). Moreover, we also penalize discrepancies in joint jerks by adding a *jerk loss*, which minimizes the Euclidean distance between reconstructed jerk $\hat{a}^t - \hat{a}^{t-1}$ and ground truth jerk $a^t - a^{t-1}$ (11). Here, a describes the acceleration and t_{frame} is the frame time.

$$\mathcal{L}_{\text{temp}}(\mathcal{X}^t, \hat{\mathcal{X}}^t) = \lambda_{\text{vel}} \sum_{j=1}^{J} \left\| v_j^t - \hat{v}_j^t \right\| \tag{10}$$

$$+ \frac{\lambda_{\text{jerk}}}{t_{\text{frame}}^3} \sum_{j=1}^{J} \left\| (a_j^t - a_j^{t-1}) - (\hat{a}^t - \hat{a}^{t-1}) \right\| \tag{11}$$

A.3.3 Contact Loss

One common issue with deep learning based retargeting methods is foot sliding and ground penetration. In order to tackle the foot sliding, we introduce a *contact position loss*. This minimizes the distance of the reconstructed height $\hat{y}$ to the ground. It is only applied on the joints, for which the ground truth joints are in contact with the ground (12). To generate these ground truth contact labels for training, a joint is considered to be in contact if its height falls below a threshold referred to as the *contact position*. The contact position is defined per animation by computing, for each frame, the minimum joint height and taking the 5th percentile of these values as the final threshold. We emphasize that this heuristic is strictly a data pre-processing step used to derive supervision signals; the contact labels are not required during inference.

Furthermore, we introduce a *contact velocity loss*, which forces the reconstructed velocity $\hat{v}$ of the ground contact joints to be zero, so that the feet are not sliding along the ground (13). To penalize ground penetration, we introduce a *ground penetration loss*, that penalizes joints penetrating the ground plane by enforcing upward correction toward zero penetration (14).

$$\mathcal{L}_{\text{cntct}}(\mathcal{X}^t, \hat{\mathcal{X}}^t) = \lambda_{\text{cntct_pos}} \sum_{j=1}^{J_{\text{cntct}}} \left\| \hat{y}_j^t \right\| \tag{12}$$

$$+ \lambda_{\text{cntct_vel}} \sum_{j=1}^{J_{\text{cntct}}} \left\| \hat{v}_j^t \right\| \tag{13}$$

$$+ \lambda_{\text{grnd_pen}} \sum_{j=1}^{J} \left\| \min(\hat{y}_j^t, 0) \right\| \tag{14}$$

A.3.4 End Effector Loss

Different sized skeletons should have the same normalized velocities while performing the same motion. In order to enforce that condition, we introduce an *end-effector loss* like (Aberman et al., 2020). This forces the velocities of the end effectors v normalized by its path length h to be the same between skeletons A and B. The skeletons A and B have to share the same number of end-effectors.

$$\mathcal{L}_{\text{ee}}(\mathcal{X}_A^t, \hat{\mathcal{X}}_B^t) = \lambda_{\text{ee}} \sum_{j=1}^{J_{\text{ee}}} \left\| \frac{v_{A,j}^t}{h_A} - \frac{\hat{v}_{B,j}^t}{h_B} \right\|$$

A.3.5 Latent Consistency

Stability in the latent representation is enforced by minimizing the Euclidean distance between the latent encoding z_p and its reconstructed version $\hat{z}_p$.

$$\mathcal{L}_{\text{latent}}(\mathcal{X}, \hat{\mathcal{X}}) = \lambda_{\text{zpose}} \| z_p - \hat{z}_p \|$$

A.3.6 Adversarial Loss

Since we have no access to the ground truth retargeting motion between two skeletons, we introduce an adversarial loss. This ensures more realistic looking motions, especially for the retargeted motions. In order to train with an adversarial loss, we introduce a separate discriminator network $\mathcal{D}$. The adversarial loss consists of two parts. The first part is the discriminator part, which classifies if the input is real (Equation 15) or fake (Equation 16). For that, only the weights of the discriminator network is updated.

The second part is the generator part, which tries to make the discriminator predict the auto-encoder output as real sample (Equation 17). Here, only the auto-encoder weights are updated.

$$\mathcal{L}_{\text{adv}}(\mathcal{X}^t, \hat{\mathcal{X}}^t) = \lambda_{\text{disc}} E_{i \sim \mathcal{M}_A} \left[\left\| 1 - \mathcal{D}(\mathcal{X}_i^t, \mathcal{S}_i) \right\| \right] \tag{15}$$

$$+ \lambda_{\text{disc}} E_{i \sim \mathcal{M}_B} \left[\left\| \mathcal{D}(\hat{\mathcal{X}}_i^t, \mathcal{S}_i) \right\| \right] \tag{16}$$

$$+ \lambda_{\text{gen}} E_{i \sim \mathcal{M}_B} \left[\left\| 1 - \mathcal{D}(\hat{\mathcal{X}}_i^t, \mathcal{S}_i) \right\| \right] \tag{17}$$

A.3.7 Loss Function Weighting

Our training objective balances multiple loss terms. The weighting coefficients for these losses are provided in Table 7. For example, the positional loss and the rotation loss are weighted by λ_{pos} and λ_{rot}, respectively. Notably, the corresponding weights for cycle consistency and augmentation consistency is the same. One example is, that λ_{pos} is the same in $\mathcal{L}_{\text{cyc.cons.}}$ and $\mathcal{L}_{\text{aug.cons.}}$.

Table 7: Loss Function Weighting Coefficients

Loss	Weight
λ_{pos}	10^2
λ_{child}	10^2
λ_{rot}	5
λ_{jerk}	10^{-5}
λ_{traj}	10
λ_{vel}	1
λ_{trans}	1
$\lambda_{\text{cntct_pos}}$	1
$\lambda_{\text{cntct_vel}}$	1
$\lambda_{\text{grnd_pen}}$	0.01
λ_{ee}	10
λ_{zpose}	0.1
λ_{disc}	0.1
λ_{gen}	1

A.4 Technical Details

Inference Latency. We measure per-frame latency for a $26 \rightarrow 26$ joint configuration. On CPU, our model runs at 2.46 ± 0.02 ms per frame, and on an NVIDIA A100 GPU at 0.12 ± 0.01 ms per frame.

Computational Cost. For a single frame with $26 \rightarrow 26$ joints, our model requires 1.82 MFLOPs.

A.5 Data Preparation

To train our autoencoder for per-frame motion retargeting across diverse skeletons, we leverage skeleton-invariant pose embeddings. For this purpose, we use publicly available motion capture datasets, including LaFAN1 (Harvey et al., 2020), ACCAD (acc), TotalCapture (Trumble et al., 2017), PFNN (Holden et al., 2017), the SFU dataset (Simon Fraser University, 2011), and CMU (Carnegie Mellon University, 2003). These datasets provide a rich variety of motions and skeletal configurations. SAME (2023) uses the same base datasets, except for CMU, but additionally includes approximately 13 hours of motion across $\sim$160 different skeletons synthesized via MotionBuilder (Autodesk, 2021) to generate paired ground-truth retargeting data. Aberman et al. (2020) and PALUM (2026) train directly on Mixamo, with the evaluation skeletons held out but the motions already seen during training. Importantly, our method trains entirely unsupervised, requiring neither paired retargeting data nor any Mixamo motion sequences, making our zero-shot evaluation more challenging than prior work.

Before passing the data into the autoencoder for training, we propose a data preprocessing step, which includes scaling of the skeleton $\mathcal{S}$. Assume h is the largest distance between two joints in the rest pose, which for human skeletons is usually the height. The scaling factor of the skeleton $\mathcal{S}$ is then defined as $1/h$. For consistency reasons, also the (previous-) joint positions and root trajectories are scaled by the same factor. In addition to the previous preprocessing, we introduce a feet-grounding step to fix floating-foot artifacts in animation datasets. For each animation, we compute the contact position, which defines when a joint is in contact with the ground. This is calculated by taking the minimum joint height per frame and using the 5th percentile as a robust, scale-invariant threshold. Joint heights in each frame are then translated by the negative contact position, ensuring the animation is properly grounded. This preprocessing is applied on a per-animation basis.

Furthermore, augmentation strategies such as random scaling of the joint lengths, global translations, and joint removal are employed to improve generalization and enables to achieve the demanded invariance. Unlike the preprocessing steps applied before training, these augmentations are applied dynamically during every training step

For testing and evaluation, we use the Bandai-Namco (Kobayashi et al., 2023) dataset, the 100Style (Mason et al., 2022) dataset and a subset of the Mixamo (Adobe, 2021) dataset. Only the Mixamo dataset is used for the Cross and Intra retargeting evaluations, following the evaluation protocols of previous motion retargeting methods, including Aberman et al. (2020) and SAME (2023), to ensure fair comparisons. The source-target ground-truth motion pairs in Mixamo are provided by Adobe, while the character pairings used for evaluation follow these previous methods. Following SAME, we exclude motion clips captured with objects, resulting in 95 motion clips per character instead of the 106 clips in the original evaluation set. The Mixamo evaluation consists of four characters for cross-structural retargeting (cross Mixamo dataset) and one additional character for intra-structural retargeting (intra Mixamo dataset). Our evaluation datasets comprise a diverse set of skeletons with varying joint counts and topological differences.

A.6 Training Protocol

The training protocol, including hardware and convergence criteria, is summarized in Table 8. Our experiments were conducted on high-performance GPUs to ensure efficient model training.

Table 8: Training Protocol Details

Specification	**Details**
Hardware	NVIDIA Tesla A100
Training Time	Approximately 48 hours

B Data Augmentation

In this section, we introduce two complementary data augmentation techniques designed to enrich skeletal animation data by perturbing its geometric properties while preserving the underlying kinematic structure: *skeleton augmentation* and *bone scaling augmentation*. Furthermore, we need *global translation augmentations* to enable the disentanglement of the pose and trajectory latent space. While training on the humanoid characters we only use the three data augmentation mentioned previously. However, when it comes to training on also non-humanoid characters, we also need *rest pose augmentations* to enrich the rest pose variety.

B.1 Rest Pose Augmentation

We perturb the skeletal animation's rest pose by applying random rotations to each joint while maintaining the hierarchical relationships between them. Let N denote the total number of joints in the skeleton, indexed by $i = 0, 1, \ldots, N-1$. For each joint i, let $p(i)$ represent the index of its parent, with $p(0) = -1$ indicating that the root joint has no parent. The original rotations are given by matrices $R_i \in SO(3)$, as obtained from the animation data.

For augmentation, we independently sample a set of random rotation matrices $\{Q_i\}_{i=0}^{N-1}$, where each $Q_i \in SO(3)$ is drawn from a exponential distribution with $\frac{\lambda}{2} \cdot \exp(-\lambda|x|)$ with $\lambda = 15$ from the rotation group. These matrices perturb the rest pose without compromising the skeletal hierarchy. The augmented rotation R'_i for each joint is computed as follows. For the root joint ($i = 0$), which has no parent, the augmented rotation is defined by

$$R'_0 = R_0\, Q_0^\top,$$

where $Q_0^\top$ (the transpose, equivalently the inverse, of Q_0) counteracts the random offset applied to the root. For a non-root joint ($i > 0$) with parent $p(i)$, the augmented rotation is given by

$$R'_i = Q_{p(i)}\, R_i\, Q_i^\top.$$

Here, left-multiplication by $Q_{p(i)}$ aligns the joint's coordinate system with the perturbed frame of its parent, while right-multiplication by $Q_i^\top$ compensates for the joint-specific random rotation. This formulation preserves the relative orientation between parent and child joints despite the applied perturbations.

Each joint is associated with an offset vector t_i representing its positional displacement from its parent. To maintain consistency after rotating the joints, these offsets are updated as

$$t'_i = Q_{p(i)}\, t_i,$$

for every joint i with $p(i) \neq -1$.

B.2 Bone Scaling Augmentation

In addition to reorienting the rest pose, we apply a global scaling augmentation to the entire skeletal structure. This strategy uniformly scales the joint offsets, thereby affecting the computed joint positions and the resulting motion dynamics while preserving the overall pose configuration.

Let t_i denote the original offset vector for joint i. The augmented offset $\tilde{t}_i$ is obtained by applying a global scaling factor s to every offset:

$$\tilde{t}_i = s \cdot t_i, \quad \forall i.$$

The scaling factor s is sampled probabilistically as

$$s = \begin{cases} \text{Uniform}(0.5, 1.5), & \text{with probability } 0.25, \\ 1.0, & \text{with probability } 0.75. \end{cases}$$

After scaling the offsets, the new joint positions are recalculated using forward kinematics. Denote by p_i the original position of joint i and by R_i its rotation matrix. The updated joint position $\tilde{p}_i$ is given by

$$\tilde{p}_i = f\left(\tilde{t}_i, R_i\right),$$

where f is the forward kinematics function that propagates the scaled offsets along the skeletal hierarchy. The same is also applied for the previous joint positions, which is part of the input feature. Furthermore, the root trajectory r is updated to $\tilde{r}$ via

$$\tilde{r} = s \cdot r.$$

The joint velocity v_i is then recomputed based on the difference between the positions in consecutive frames:

$$v_i = \tilde{p}_i(t) - \tilde{p}_i(t-1).$$

This recalculation ensures that the dynamic properties of the motion remain consistent with the new, scaled geometry.

B.3 Skeleton Augmentation

In order to get a more diverse skeleton dataset, we combine global scaling, local scaling and joint removal augmentations to each skeletal structure used in the cycle consistency step. Let t_i denote the original offset vector for joint i.

The global scaling procedure is done in the same way as described in subsection B.2. Assume in the following, that $\tilde{t}_i$ is the global scaling augmented offset, which is obtained by applying a global scaling factor s to every offset. In order to perform the local scaling augmentation, we first draw a scaling factor s_i for each joint in the skeleton. This is sampled independent from other joint scaling factors probabilistically as:

$$s_i = \begin{cases} \text{Uniform}(0.5, 1.5), & \text{with probability } 0.25, \\ 1.0, & \text{with probability } 0.75. \end{cases}$$

The already globally scaled offset is then scaled for each joint with its respective joint scaling value.

$$\tilde{t}_i \to s_i \cdot \tilde{t}_i, \quad \forall i.$$

In the last step, we also further augment the skeleton, by introducing random joint removal. First we need the determine the number of joints, which will be removed. This number $N_{\text{to remove}}$ is sampled probabilistically as:

$$N_{\text{to remove}} = \text{Uniform}(0, N_{\text{max num}}).$$

$N_{\text{max num}}$ refers to the maximal number of joints, which will be removed and is a hyperparameter. In our case, $N_{\text{max num}}$ is chosen to be 4. After that, we determine which joints specifically will be removed. For that, we again sample $N_{\text{to remove}}$ numbers, which are in the range of the number of joints, leaving out zero since this is referred to as the root joint. Each joint which should be removed $J_{i,\text{to remove}}$ will be probabilistically sampled in the following way:

$$J_{i,\text{to remove}} = \text{Uniform}(1, N_{\text{joints}} - 1).$$

Here we use $N_{\text{joints}} - 1$, because we start counting from zero on. After we sampled all our $J_{i,\text{to remove}}$, we only keep the unique joint number $\text{Unique}(J_{\text{to remove}})$. At the end we need to update the skeleton, which is described by its parents and offsets. Assume in the following, that C_i are all the children of joint i and p_i is the parent of joints i. Then we update the parent child relation in the following way:

$$p_j = p_i \quad \forall i \in \text{Unique}(J_{\text{to remove}}), \forall j \in C_i.$$

For the offsets, we update it in the following way:

$$\tilde{t}_j \to \tilde{t}_j + \tilde{t}_i \quad \forall i \in \text{Unique}(J_{\text{to remove}}), \forall j \in C_i.$$

After that, all joints in the set $\text{Unique}(J_{\text{to remove}})$ in the parents-child hierarchy as well as for the offsets will be removed.

B.4 Global Translation Augmentation

To further enhance robustness to variations in absolute positioning, we introduce a global translation augmentation that shifts the entire skeletal pose along the horizontal plane. This augmentation simulates variations in subject placement by adding random offsets in the x and z coordinates. Let $T = (T_x, T_z)$ denote the translation vector applied to the global pose. For each sample, the translation components T_x and T_z are independently sampled from a uniform distribution:

$$T_x \sim \text{Uniform}(a_x, b_x), \quad T_z \sim \text{Uniform}(a_z, b_z),$$

where a_x, b_x, a_z, and b_z are hyperparameters defining the translation range. After sampling T, the translation is applied uniformly to all joint positions. For each joint with original position $p_i = (x_i, y_i, z_i)$, the updated position $\tilde{p}_i$ is computed as

$$\tilde{p}_i = (x_i + T_x,\ y_i,\ z_i + T_z).$$

This augmentation is also applied to any global trajectories associated with the pose, ensuring that the spatial dynamics remain consistent with the translation.

B.5 Limitations of the Sparse Representation

Although the formulation above allows exact recovery of q from $\bar{q}$, it has two significant limitations. First, the dimensionality of $\bar{q}$ grows linearly with the number of joints J, which is suboptimal for deep learning architectures that require fixed-size inputs. Second, due to the sparsity of M and T, only a subset of the latent dimensions is actively utilized, potentially constraining the expressive capacity of the representation. These challenges motivate our design of a dense, constant-dimensional representation inspired by the operations described above.

C Further Results

C.1 Loss Ablation Studies

We ablate each loss individually to verify its contribution to reconstruction and retargeting (Tab. 9). *Structural losses:* removing $\mathcal{L}_{\text{child}}$ substantially increases GJP ($1.45 \to 3.21$ Intra, $1.28 \to 2.51$

Cross), while removing $\mathcal{L}_{\text{ee}}$ causes smaller but consistent degradations. $\mathcal{L}_{\text{cyc. cons.}}$ is essential for cross-structural transfer, increasing Cross GJP from $1.28 \rightarrow 3.25$ when removed, whereas $\mathcal{L}_{\text{adv}}$ provides smaller but consistent improvements.

removing $\mathcal{L}_{\text{jerk}}$ or $\mathcal{L}_{\text{vel}}$ degrades performance, with $\mathcal{L}_{\text{jerk}}$ notably increasing Cross GJP to 2.07 due to overly smooth but inaccurate motions.

$\mathcal{L}_{\text{cntct_vel}}$ has a stronger impact than $\mathcal{L}_{\text{cntct_pos}}$, particularly on Intra retargeting ($1.45 \rightarrow 2.26$). Removing $\mathcal{L}_{\text{cntct_pos}}$ slightly improves Intra but harms Cross performance, the more challenging and practically relevant setting, suggesting it mainly benefits cross-topology generalization. $\mathcal{L}_{\text{grnd_pen}}$ further improves overall retargeting stability and performance across both settings.

We find that our design choices provide at minimum a 11% intra, and 13% cross improvement on GJP. However, $\mathcal{L}_{\text{cyc. cons.}}$ by design only improves GJP on the cross task, and $\mathcal{L}_{\text{cntct_pos}}$ increases the GJP error (6% intra; 3% cross), however, decreases foot sliding.

Table 9: Ablation Study on the Mixamo dataset. We evaluate the impact of removing key components on Intra- and Cross-retargeting performance. **Bold** indicates best; underline indicates second best.

	Intra			Cross		
Method	GJP	Jerk	PART	GJP	Jerk	PART
Ours Full	<u>1.45</u>	0.72	**0.03**	**1.28**	0.72	**0.05**
w/o $\mathcal{L}_{\text{cyc. cons.}}$	1.47	**0.82**	**0.03**	3.25	**1.23**	<u>0.06</u>
w/o $\mathcal{L}_{\text{adv}}$.	1.65	0.67	**0.03**	1.47	0.69	**0.05**
w/o $\mathcal{L}_{\text{child}}$	3.21	0.77	0.05	2.51	0.75	0.06
w/o $\mathcal{L}_{\text{ee}}$	1.62	0.70	**0.03**	1.46	0.71	0.04
w/o $\mathcal{L}_{\text{vel}}$	1.69	0.66	**0.03**	1.45	0.68	**0.05**
w/o $\mathcal{L}_{\text{jerk}}$	1.87	0.68	<u>0.04</u>	2.07	0.68	**0.05**
w/o $\mathcal{L}_{\text{cntct_pos}}$	**1.37**	<u>0.81</u>	<u>0.04</u>	<u>1.32</u>	<u>0.76</u>	**0.05**
w/o $\mathcal{L}_{\text{cntct_vel}}$	2.26	0.68	<u>0.04</u>	1.74	0.71	**0.05**
w/o $\mathcal{L}_{\text{grnd_pen}}$	1.70	0.72	<u>0.04</u>	1.51	0.72	**0.05**

C.2 TRAJECTORY ERROR ACCUMULATION

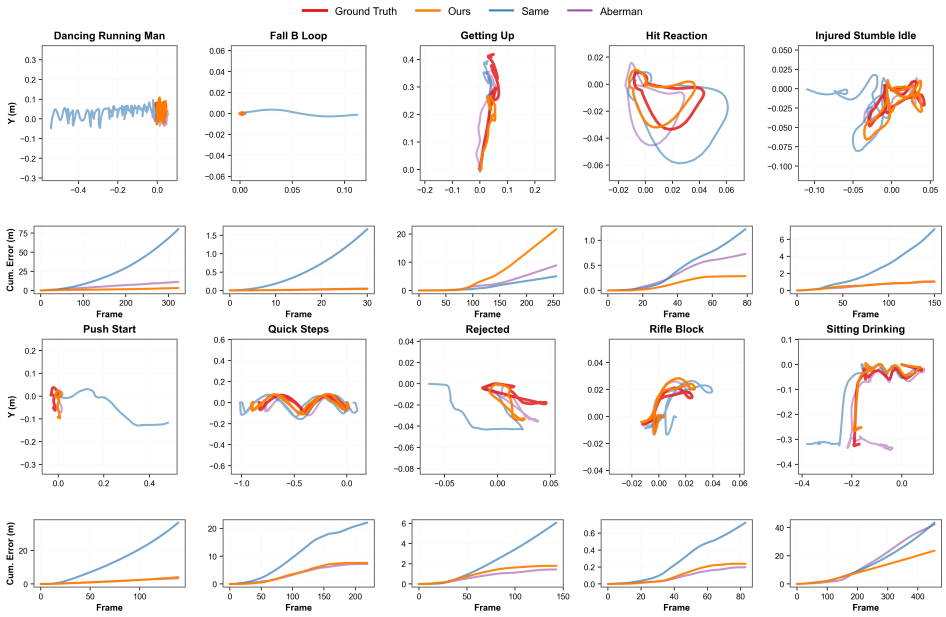


Figure 7: Error accumulation when using the delta representation for trajectories:. In contrast, predicting the global position (Ours, Aberman et al. (2020)) does not result in as much error accumulation (SAME). It shows the global root trajectory projected on the xz plane (top) and the xz root trajectory error with increasing number of frames (bottom) for three different motions. The global root trajectory prediction method used by us and Aberman et al. (2020) does not suffer from error accumulation, unlike SAME who predicts only the difference between two frames.

C.3 SKELETON INVARIANCE

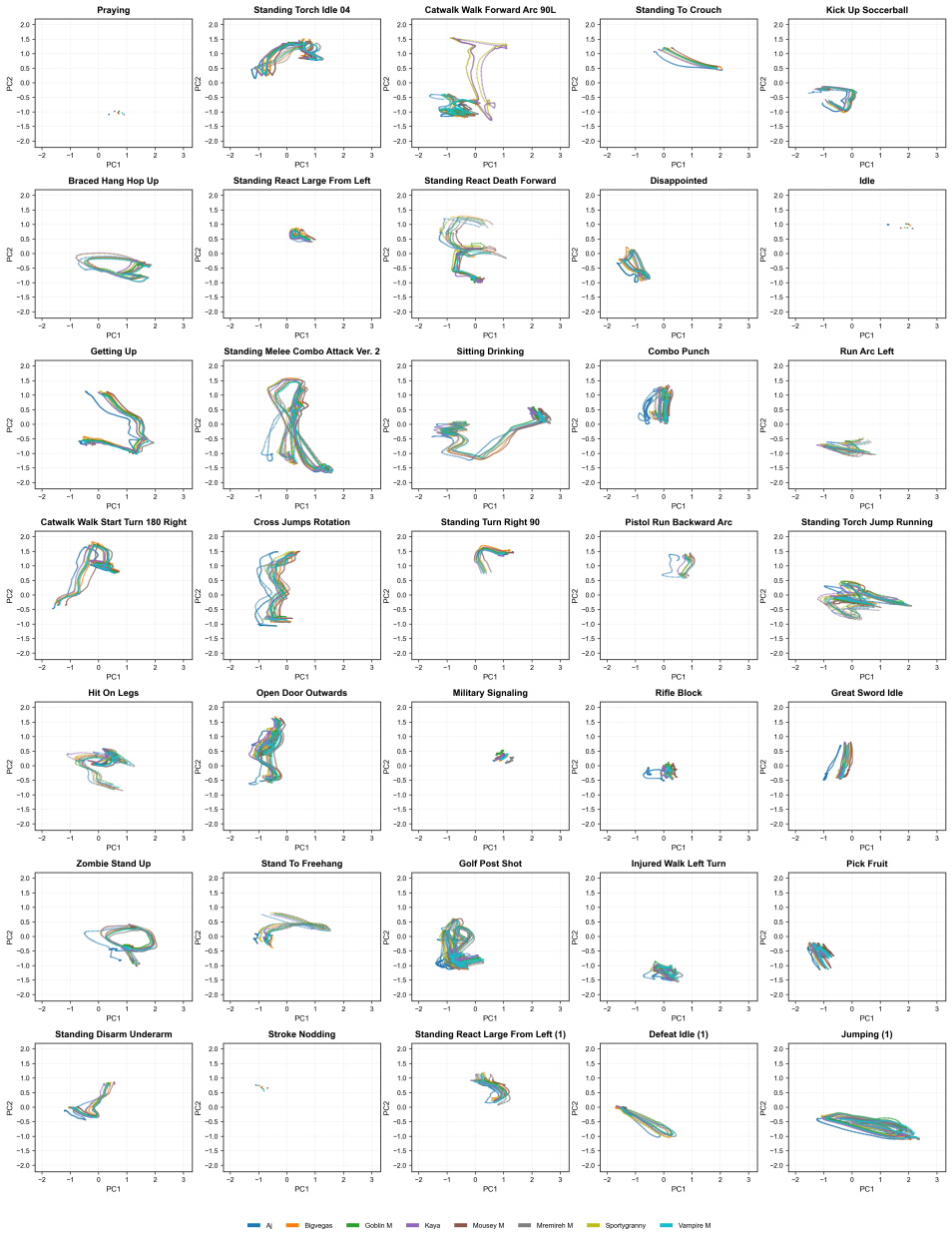


Figure 8: Additional skeleton invariance latent spaces of the model: 2D Principal Component Analysis projected latent space of the model for different skeletons performing semantically identical motions for different motions. The PCA space is shared across all sequences.

C.4 RETARGETING

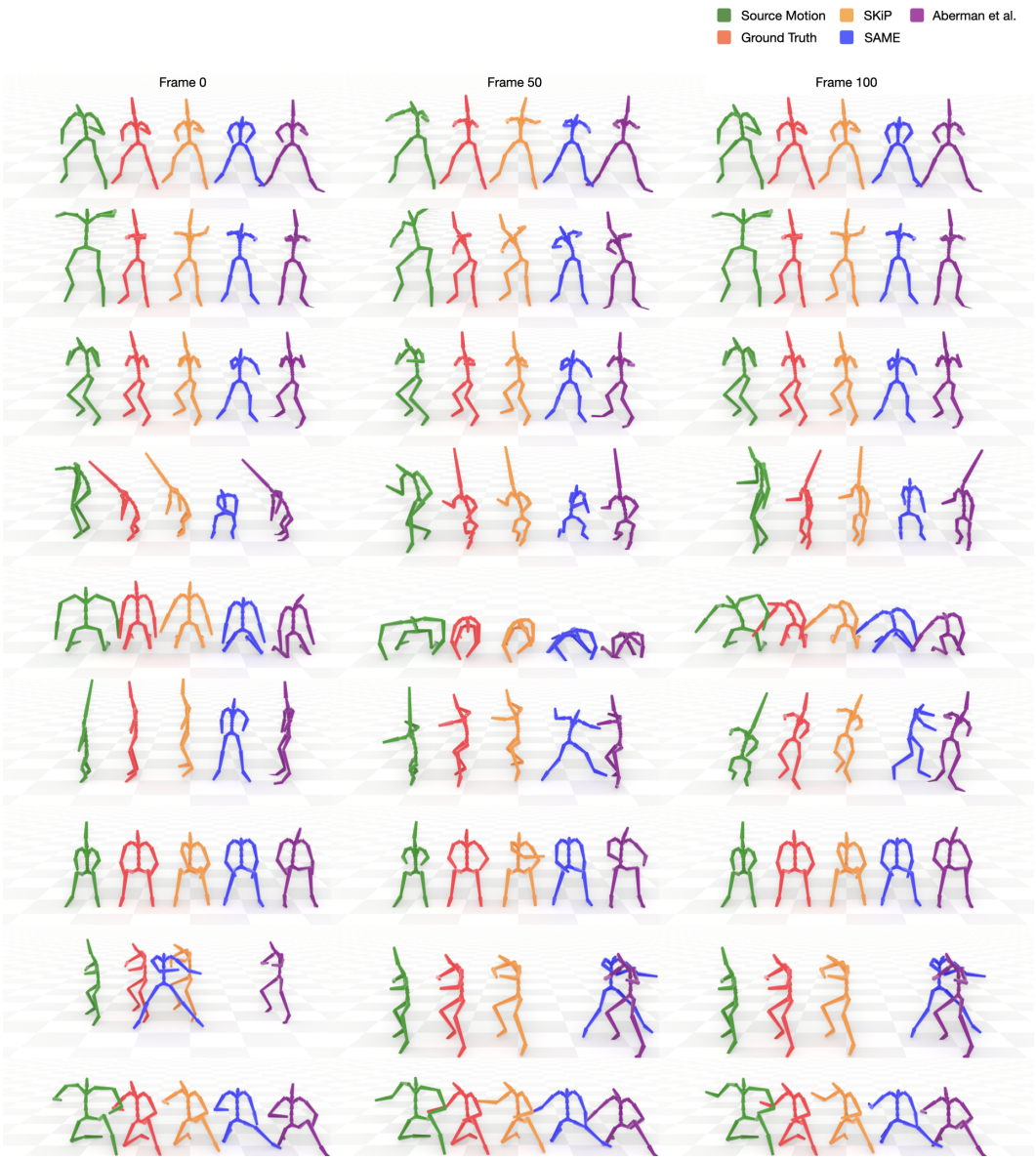


Figure 9: Additional examples of retargeting across diverse skeletons: The green skeletons are the source motions, red are ground truth retargets from the Mixamo dataset, the orange are retargets from our method, blue are from SAME and purple from Aberman et al. (2020). Each row represents the same motion, while each column frames 0, 50 and 100 from that clip.