BRANCHTOKENS: BRANCH-CENTRIC TOKENIZATION AND TEST-TIME AUGMENTATION FOR SKELETON GENERATION

Anonymous authors
Paper under double-blind review

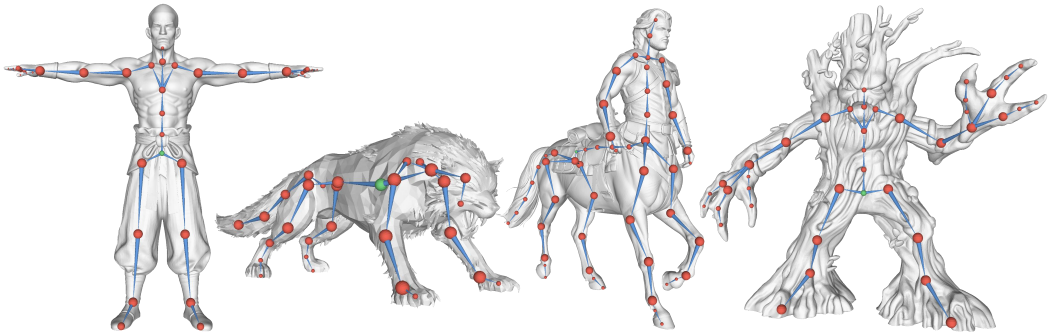


Figure 1: BranchTokens predicts skeletons accurately from in-the-wild meshes, including humans, animals, and non-existent fantasy creatures. See supplementary materials for interactive visualization.

ABSTRACT

Automatic skeleton generation involves predicting both joint positions and skeletal connectivity. However, existing approaches struggle to encode branch structures into token sequences and do not use test-time computation effectively. We address these choices within **BranchTokens**, a unified autoregressive framework. First, we introduce branch-centric tokenization, a branch-aware representation that places structurally related elements next to each other and encodes connectivity directly in the sequence. Compared with standard BFS-style serialization, this representation yields more compact sequences. Second, we introduce view-augmented generation, a test-time augmentation procedure that applies axis-aligned rotations to the input mesh, maps all predictions back to a common frame, and selects the final skeleton based on mesh coverage and consistency among predictions from different views. Experiments show that our method achieves better skeleton prediction accuracy than state-of-the-art methods. In particular, our method reduces the CD-J2B error by 16.9% on the Articulation-XL2.0 dataset compared to the strongest baseline, Auto-Connect. Qualitative results on in-the-wild meshes further demonstrate generalization across diverse inputs.

1 INTRODUCTION

Creating the skeleton of a given static mesh is a crucial step for character animation. Traditionally, this process is done manually by skilled rigging artists with domain knowledge of human and animal anatomy. With the advancement in deep learning (Vaswani et al., 2017; Brown et al., 2020) and the availability of large rigged-mesh datasets (Xu et al., 2019; Song et al., 2025a; Xu et al., 2020; Deng et al., 2025), automatic skeleton generation (Guo et al., 2025a; Zhang et al., 2025; Song et al., 2025a;b; Liu et al., 2025; Zhang et al., 2026; Deng et al., 2025) has substantially improved. Recent state-of-the-art (SOTA) skeleton generation methods (Zhang et al., 2025; Song et al., 2025a; Guo et al., 2025a) are inspired by language models (Touvron et al., 2023; Brown et al., 2020). The main idea is to represent a skeleton as a sequence of tokens and autoregressively decode the token sequence with Transformers (Zhang et al., 2022) given an encoded feature vector of the input mesh (Zhao et al., 2023; Yu et al., 2022). Despite this progress, some design choices could still be improved.

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.

First, the quality of these autoregressive models heavily depends on how the skeleton is tokenized. Auto-Connect (Guo et al., 2025a) introduced a unified vocabulary that encodes both joint positions and connectivity. This enables standard autoregressive generation without a separate connectivity module. However, a standard breadth-first search (BFS) traversal is used to generate the sequence from the skeleton, which treats all nodes identically. In other words, the tokenization does not explicitly capture the different structural roles of nodes in the skeleton. Second, existing works (Zhang et al., 2025; 2026) use *generic* test-time scaling methods to improve performance at the cost of computation, *e.g.*, increasing the beam width during decoding. In this work, we address these two shortcomings.

Intuitively, a good tokenization should compactly represent both joint positions and connectivity by keeping structurally related joints close together in the token sequence. Classical work on skeleton representation distinguishes junction nodes (degree > 2), leaf nodes (degree $= 1$), and degree-2 connection nodes that form branches between them (Cornea et al., 2024; Felzenszwalb & Huttenlocher, 2005; Torsello & Hancock, 2006; Siddiqi et al., 1999; Bai et al., 2009; Bai & Latecki, 2008). Inspired by these works, we propose **branch-centric tokenization**, where the serialization first visits the junction and leaf nodes to identify the branch structure. Then, the connection nodes are placed back between the nodes representing the two ends of a branch. In other words, tokens from the same branch are placed adjacent in the sequence.

Next, we introduce a test-time augmentation (TTA) approach to improve test-time performance that is complementary to the existing generic test-time scaling. The proposed TTA approach is based on the observation that meshes in existing datasets, *e.g.*, Articulation-XL2.0 (Song et al., 2025a), do not have a consistent upright direction/axis (Zhao et al., 2026). Commonly, 3D assets from different sources may appear in different axis-aligned orientations. To address this orientation issue, we perform *view augmentation* on the mesh to generate one skeleton for each rotated view. We then introduce a *coverage-consensus aggregation* to select the final generation based on the coverage of the mesh geometry and agreement among the other predictions.

Empirically, we combine these two improvements into a single framework, **BranchTokens**, by training a standard autoregressive next-token prediction model (Zhang et al., 2022) for skeleton generation; other design choices are kept the same as the baseline SOTA of Auto-Connect. Across Articulation-XL2.0, ModelsResource, and the more challenging diverse-pose split, our model achieves the best results on all reported metrics. Specifically, relative to Auto-Connect on Articulation-XL2.0, our model reduces CD-J2J, CD-J2B, and CD-B2B by 8.1%, 16.9%, and 14.0%, respectively.

Our contributions are as follows:

- We propose branch-centric tokenization, a branch-aware skeleton representation that encodes connectivity directly and places structurally related elements close together in the token sequence.
- We introduce view-augmented generation, a test-time augmentation strategy for skeleton generation that applies axis-aligned input rotations and selects the final prediction based on mesh coverage and cross-view consistency.
- Extensive experiments show that our method achieves state-of-the-art skeleton prediction accuracy on both in-dataset and cross-dataset evaluations, and generalizes well to in-the-wild meshes.

2 Related Work

Automatic skeleton generation has made progress from geometric and topology-driven methods to data-driven template-based and template-free models. Early systems embedded a predefined skeleton into a mesh with geometric heuristics (Baran & Popović, 2007), while curve-skeleton and medial-axis methods extracted topology-aware structures from point clouds or volumetric shape representations (Tagliasacchi et al., 2009; Yan et al., 2016; 2018). Later template-based methods improved robustness for constrained domains such as humanoids by fitting, transferring, or deforming a predefined rig and learning articulation priors from data (Feng et al., 2015; Li et al., 2021; Ma & Zhang, 2023; Chu et al., 2025).

In contrast, template-free approaches predict joints and connectivity directly from geometry, using volumetric networks, graph neural networks, or motion-aware priors to handle more diverse categories and topologies (Xu et al., 2019; 2020; 2022). Other recent works (Guo et al., 2025b; Sun et al., 2025a) go beyond fixed templates but are only demonstrated on characters, which limits their applicability to more diverse categories.

The availability of large-scale rigging datasets, such as Anymate, Articulation-XL2.0, and Rig-XL (Deng et al., 2025; Song et al., 2025a; Zhang et al., 2025), has further accelerated recent progress in automatic skeleton generation. More recent methods increasingly cast skeleton generation as structured prediction over a serialized tree. A common pattern is to encode the input shape and autoregressively decode a variable-length skeleton sequence, which better handles varying numbers of joints and bones and their dependencies (Liu et al., 2025; Song et al., 2025b; Zhang et al., 2025; Song et al., 2025a; Guo et al., 2025a; Sun et al., 2026; Qin et al., 2026; Sun et al., 2025b).

Within this trend, methods still differ in how they represent and predict connectivity. MagicArticulate (Song et al., 2025b) studies spatial and hierarchical orderings for serialized bone sequences. UniRig (Zhang et al., 2025) and TokenRig (Zhang et al., 2026) decompose the skeleton into branch-like subsequences, autoregressively predict quantized coordinates for each branch, and reconstruct cross-branch parent-child relations by proximity during detokenization. Anymate (Deng et al., 2025) predicts connectivity in a separate stage after joint prediction. RigAnything (Liu et al., 2025) and Puppeteer (Song et al., 2025a) both use autoregressive joint sequences with order-dependent parent indices, which couple the representation of a joint to the chosen traversal.

Most related to this work is Auto-Connect (Guo et al., 2025a), which uses a unified connectivity-preserving tokenization to model topology and coordinates in a single vocabulary, and further improves topological accuracy with a DPO-based post-training stage. Our method is similar to Auto-Connect in model architecture but with a different tokenization design. Also, unlike Auto-Connect, our method does not use post-training and instead studies test-time augmentation.

Test-Time Augmentation (TTA) and scaling. TTA improves prediction by aggregating outputs from multiple transformed versions of the same test input. In image classification, early TTA methods used heuristic multi-crop, flip, and multi-scale evaluation to improve accuracy (Krizhevsky et al., 2012; Szegedy et al., 2015; Simonyan & Zisserman, 2014). Later work made TTA more adaptive by learning which test-time views to apply or how to aggregate them. For example, Lyzhov et al. (2020) learn a test-time augmentation policy, Kim et al. (2020) predict useful transformations for each test instance, and Shanmugam et al. (2021) replace uniform averaging with learned aggregation weights. However, these methods focus on image classification and do not directly transfer to structured skeleton generation, where predictions may differ in both topology and correspondence; it remains unclear how to aggregate these predictions. Note that, in prior skeleton generation work, test-time scaling mainly relies on beam search for autoregressive decoding (Wu et al., 2016; Zhang et al., 2025; 2026). In contrast, our method adopts TTA for test-time scaling by generating multiple hypotheses from transformed inputs and aggregating them with a task-specific strategy.

3 Background

To establish a common notation, we review the task of skeleton generation and the prior work, Auto-Connect (Guo et al., 2025a). We then provide an overview of the TTA framework.

Task formulation. Given an input mesh $\mathcal{M}$, the goal of skeleton generation is to build a model f_θ that generates the corresponding skeleton $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ represented as a rooted tree. Here, $\mathcal{V}$ is the set of nodes denoting the skeleton joints, and $\mathcal{E}$ is the set of edges between two nodes indicating the parent-child relationship in a tree (Zhang et al., 2025). Each skeleton joint $v \in \mathcal{V}$ corresponds to a 3D coordinate of $j^v = (j_x^v, j_y^v, j_z^v)$ in the normalized unit cube space of $[-1, 1]^3$.

Recent works (Guo et al., 2025a) formulate the model f_θ as a sequence generation task using Transformers (Vaswani et al., 2017) parametrized by θ. The main innovation is how to convert the skeleton (a tree) into a sequence of tokens $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_T)$, which we now describe in detail.

Connectivity-preserving tokenization. The main challenge to represent a skeleton $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ as tokens is to preserve the connectivity information of the tree structure. Tokens should not only encode the coordinates of the joints v, but also the parent-child relationships between joints. To do so, Auto-Connect (Guo et al., 2025a) discretizes the joint coordinates using n-bit quantization, where each coordinate is mapped into a discrete index, *i.e.*,

$$Q_n(a) = \left\lfloor (a+1) \cdot 2^{n-1} \right\rfloor . \tag{1}$$

Recall that each joint v lies within the normalized cube space $[-1, 1]^3$, so the quantization maps each coordinate into an integer index in $\mathcal{C}_{\texttt{coord}} \triangleq \{0, 1, \ldots, 2^n - 1\}$. $a = 1$ is clamped to $2^n - 1$.

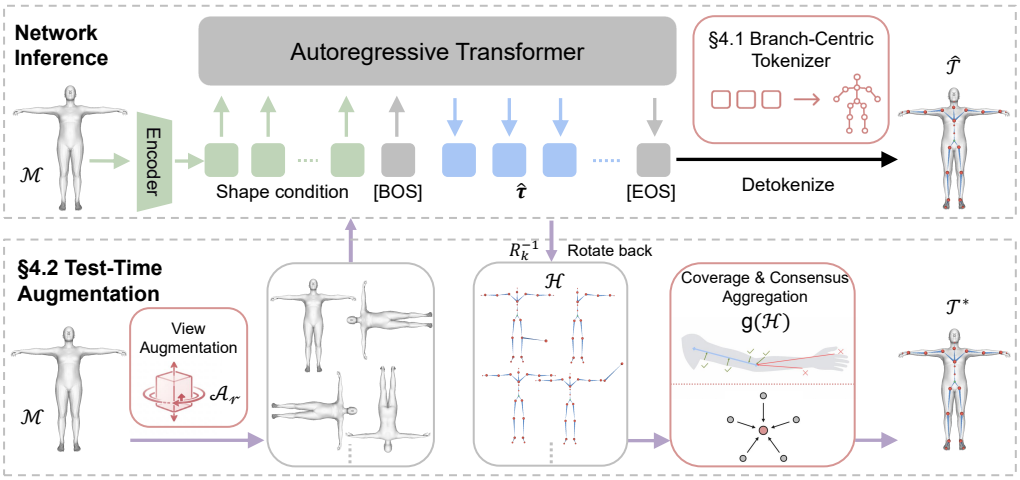


Figure 2: Overview of BranchTokens. **Top: network inference with branch-centric tokenization.** Following the standard mesh-conditioned autoregressive formulation reviewed in Sec. 3, the input mesh $\mathcal{M}$ is first encoded into shape-conditioned features, and the autoregressive transformer predicts a token sequence from `[BOS]` to `[EOS]`. In Sec. 4.1, we design branch-centric tokenization (BCT), which organizes the skeleton by branch structure and is used to detokenize the predicted sequence $\widehat{\boldsymbol{\tau}}$ into the skeleton tree $\widehat{\mathcal{T}}$. **Bottom: test-time augmentation.** In Sec. 4.2, we apply the rotation set $\mathcal{A}_r$ to the input mesh, decode one skeleton hypothesis for each rotated view, map all predictions back to the canonical frame with the inverse rotations R_k^{-1}, and form the hypothesis pool $\mathcal{H}$. Coverage-consensus aggregation $g(\mathcal{H})$ then selects the final prediction $\mathcal{T}^*$ by favoring hypotheses that both better cover the mesh geometry and agree with the other candidates.

To capture the tree structure, a set of four special structure tokens $\mathcal{C}_{\texttt{struct}}$ is introduced to encode parent-child relationships: [BOS] (beginning of sequence), [EOS] (end of sequence), [E1] (end of a child sequence), and [E2] (end of a level), *i.e.*,

$$\mathcal{C}_{\text{struct}} \triangleq \{[\text{BOS}], [\text{EOS}], [\text{E1}], [\text{E2}]\}. \tag{2}$$

Auto-Connect (Guo et al., 2025a) then determines the ordering of these tokens by traversing the skeleton in a Breadth-First Search (BFS) order, processing the tree level by level.

With the sequence of tokens defined, a Transformer model f_θ is trained to predict the token sequence $\widehat{\boldsymbol{\tau}} = f_\theta(\mathcal{M})$ given the input mesh $\mathcal{M}$. We refer readers to the original Auto-Connect paper (Guo et al., 2025a) for further details on the training procedure, *e.g.*, losses and DPO post-training (which we do not need).

Test-Time Augmentation (TTA). As the name suggests, TTA is a family of techniques that augment a test input and aggregate the model's predictions across the augmented variants to improve performance. More formally, let x be the test input, $\mathcal{A} = \{a_1, a_2, \ldots, a_K\}$ denote a set of transformations, and f_θ be the model. TTA outputs an aggregated prediction $y^\star$ by applying the model to each transformed input $a_k(x)$ and aggregating the predictions with an aggregation function g. In theory, any function that is invariant to the permutation in the index k could work. For tasks such as image classification, a common choice for g is the average function, *i.e.*,

$$y^\star = \frac{1}{K} \sum_{k=1}^{K} f_\theta(a_k(x)). \tag{3}$$

In practice, the choice of the transformation set $\mathcal{A}$ and the aggregation function g is defined based on the specific task at hand.

4 Method

We build upon Auto-Connect (Guo et al., 2025a) and improve it in two directions: **(a)** Auto-Connect relies on BFS-based connectivity-preserving tokenization, as reviewed in Sec. 3. However, this

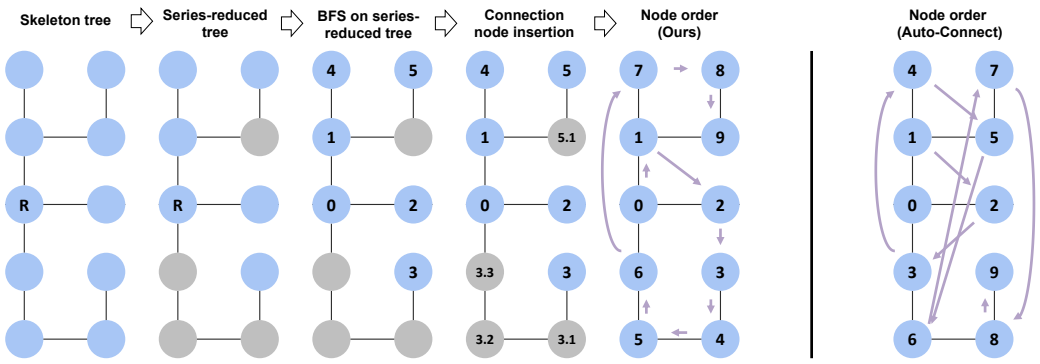


Figure 3: Comparison of node ordering with Auto-Connect. **Left: illustration of branch-centric tokenization (BCT).** Starting from the input skeleton tree, we first construct its series-reduced tree and run BFS on this reduced tree to order the branches. We then reinsert the smoothed connection nodes along each visited branch, which yields the final node order used by BCT. For clarity, we omit delimiter insertion in this figure. We use purple arrows to indicate the node traversal order. **Right: the node order of Auto-Connect.** As indicated by the arrows, Auto-Connect interleaves nodes from one branch with nodes from other branches in the traversal sequence. In contrast, our branch-level ordering keeps nodes from the same branch adjacent in the serialized sequence.

tokenization treats all nodes equally during its traversal, which does not utilize additional node properties that could be beneficial in representing tree topology; **(b)** Auto-Connect handles the ambiguity of z-axis alignment only at training time through random rotation augmentation, which may be insufficient to address the diverse axis-aligned orientations in the data. In this work, we propose *branch-centric tokenization* (Sec. 4.1), which focuses on viewing a skeleton as a sequence of branches (a.k.a. chains in the graph literature) rather than a sequence of individual joints. We then introduce a *test-time augmentation* (Sec. 4.2) that augments along different axis alignments and then aggregates to make a final prediction. Together, these two components form our framework, **BranchTokens**. The overall pipeline is illustrated in Fig. 2.

4.1 Branch-Centric Tokenization (BCT)

Our proposed tokenization is inspired by classic works on skeleton graphs (Bai & Latecki, 2008; Bai et al., 2009), which define several different node types: a *junction node* is a node with more than two adjacent nodes, a *connection node* is a node with two adjacent nodes, and a *leaf (end) node* is a node with only one adjacent node. A *skeleton branch* (series/chain) is defined as a path connecting two non-connection nodes, with all intermediate nodes, if any, being connection nodes. In a skeleton, junction and leaf nodes determine the branching structure, whereas connection nodes mainly describe the within-branch structure.

Existing works using a BFS traversal on the tree do not consider the branching structure, and **may split nodes from the same branch into separate parts** of the traversal sequence. See illustration in Fig. 3 (right). This ordering requires the model to remember branch information throughout a longer sequence, which we view as an undesirable property of the representation.

To address this issue, we propose branch-centric tokenization (BCT), which focuses on ensuring the nodes within a branch remain in a chain after the tokenization. As illustrated in Fig. 3, BCT involves four stages: (a) tree smoothing, (b) BFS on the series-reduced tree, (c) connection node insertion, and (d) delimiter insertion.

Tree smoothing. To build a branch-centric representation, we first need to identify all the non-connection nodes. This is done by "smoothing" the tree, *i.e.*, replacing the two edges incident to a connection node with a single edge joining its two neighbors and removing that node. We smooth the skeleton tree repeatedly until no vertices of degree two remain, to obtain the series-reduced tree (Gross et al., 2018) with vertices

$$\mathcal{V}_{\mathtt{sr}} \triangleq \{v \in \mathcal{V} \mid \deg(v) \neq 2\} \cup \{v^{\text{root}}\}. \tag{4}$$

Here, we always include the root in the set of nodes. This yields the rooted series-reduced tree $\mathcal{T}_{\mathtt{sr}} = (\mathcal{V}_{\mathtt{sr}}, \mathcal{E}_{\mathtt{sr}})$, whose edges $\mathcal{E}_{\mathtt{sr}}$ correspond to *branches* in the original skeleton tree.

BFS on series-reduced tree and connection node insertion. We serialize $\mathcal{T}_{\mathtt{sr}}$ in breadth-first order starting from v^{root}, which determines an ordering of branches. To obtain a deterministic sibling order, for the children of each series-reduced node p, we use a *distance-aware traversal (DAT)* rule and sort them in ascending order of Euclidean distance to p, instead of using random shuffling or spatial sequence ordering (Song et al., 2025b). This rotation-invariant rule yields a more consistent serialization order under rotations. For each visited series-reduced edge (p, v), we first emit the node v and then insert the intermediate connection nodes on the corresponding branch in the original tree along the path from v back to p, yielding the ordered subsequence $\mathcal{S}_{v \to p}$. Each branch forms a contiguous subsequence in the final token sequence.

Delimiter insertion. Lastly, we use the same `[E1]` and `[E2]`, as reviewed in Sec. 3, to mark the end of a child sequence and the end of a BFS level. We additionally introduce `[E3]` to terminate each local branch after all nodes in $\mathcal{S}_{v \to p}$ have been emitted. This extra delimiter makes branch boundaries explicit in the token sequence and helps distinguish starting a new branch from continuing the current one. We provide the full algorithm in Appendix Sec. B.

Benefits of BCT. Besides the fact that nodes within a branch maintain their closeness in the tokenized sequence, we also observe that BCT leads to a more compact representation. Fig. 4 shows the empirical distribution of sequence lengths for our tokenization and Auto-Connect's tokenization. We observe that our tokenization produces shorter sequences for the large majority of samples, which should be beneficial, as shorter sequences are typically easier to model (Khandelwal et al., 2018; Press et al., 2021; Li et al., 2022).

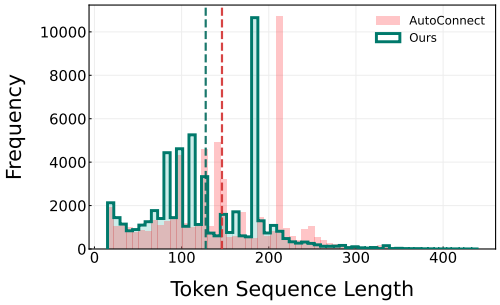


Figure 4: Token sequence length distributions on Articulation-XL2.0. Our branch-centric tokenization on average produces shorter token sequences than Auto-Connect, with mean lengths of 127.6 vs. 146.3 tokens (indicated by the dashed lines). Auto-Connect produces longer token sequences than ours in 94.4% of samples.

4.2 Test-Time Augmentation for Skeleton Generation

Meshes at test time can appear in different axis-aligned orientations rather than a single canonical upright frame, as in datasets such as Articulation-XL2.0 and practical 3D asset collections. This creates alignment ambiguity for autoregressive skeleton generation. An intuitive idea is to apply test-time augmentation over multiple rotated views. However, standard test-time augmentation, *e.g.*, for image classification, does not directly apply. Predictions from different views may differ in both topology and joint correspondence; it is *unclear* how to "aggregate" these predictions.

We introduce a coverage-consensus aggregation rule that selects the final prediction using two criteria: mesh coverage and cross-view agreement.

View-augmented generation. We define our augmentation set $\mathcal{A}_r = \{a_1, \ldots, a_6\}$ as a discrete set of six rotations, each represented by a rotation matrix $R_k \in \mathrm{SO}(3)$ such that

$$R_k \mathbf{u}_k = \mathbf{e}_z, \qquad \mathbf{u}_k \in \{\pm\mathbf{e}_x, \pm\mathbf{e}_y, \pm\mathbf{e}_z\},$$

where $\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z$ are the canonical coordinate axes. That is, $\mathcal{A}_r$ contains the identity rotation and five additional shortest-arc rotations that map each axial direction to the canonical z-axis. We center this set on the z-axis because many 3D assets use z as an upright direction. For an input mesh $\mathcal{M}$, the model independently predicts a skeleton for each augmented view $a_k(\mathcal{M})$. We then map the k-th prediction back to the canonical frame by applying the inverse rotation R_k^{-1}, yielding the prediction $\widehat{\mathcal{T}}_k = (\widehat{\mathcal{V}}_k, \widehat{\mathcal{E}}_k)$ and its joint set $\widehat{\mathcal{J}}_k = \{j^v\}_{v \in \widehat{\mathcal{V}}_k}$. We denote the set of predictions as $\mathcal{H} = \{\widehat{\mathcal{T}}_1, \ldots, \widehat{\mathcal{T}}_6\}$ in a shared canonical coordinate system.

Table 1: **Quantitative comparison with baselines across three test sets.** We report results on Articulation-XL2.0, ModelsResource, and the diverse-pose split using CD-J2J, CD-J2B, and CD-B2B. Our method achieves the best results among directly comparable methods. **Bold** numbers denote the best result among directly comparable methods in each column. *Gray results are reported from TokenRig's paper and are included for reference only, as TokenRig uses additional training data, including the ModelsResource training set.

Method	Articulation-XL2.0			ModelsResource			Diverse-pose		
	J2J ↓	J2B ↓	B2B ↓	J2J ↓	J2B ↓	B2B ↓	J2J ↓	J2B ↓	B2B ↓
Pinocchio (Baran & Popović, 2007)	8.324	6.612	5.485	6.852	4.824	4.089	7.967	6.411	5.149
RigNet (Xu et al., 2020)	7.618	6.076	5.279	7.223	5.987	4.329	7.751	6.392	5.713
UniRig (Zhang et al., 2025)	3.305	2.611	2.180	3.964	3.021	2.570	3.252	2.569	2.077
MagicArti. (Song et al., 2025b)	3.264	2.503	2.123	4.114	3.137	2.693	4.376	3.456	2.955
Puppeteer (Song et al., 2025a)	3.109	2.370	1.983	3.766	2.804	2.405	2.514	1.986	1.598
Auto-Connect (Guo et al., 2025a)	2.572	2.030	1.682	3.735	2.816	2.362	-	-	-
TokenRig (Zhang et al., 2026)*	2.485	1.599	1.463	2.893	2.012	1.547	2.317	1.540	1.446
Ours w/o TTA	2.438	1.770	1.517	3.547	2.632	2.115	2.088	1.557	1.338
Ours	**2.364**	**1.687**	**1.447**	**3.445**	**2.513**	**2.016**	**2.052**	**1.521**	**1.296**

Coverage-consensus aggregation. Given the set of predictions $\mathcal{H}$, we select a single skeleton $\mathcal{T}^*$ by minimizing a score that considers both coverage and joint consensus:

$$\mathcal{T}^* = g(\mathcal{H}) \triangleq \arg\min_{\widehat{\mathcal{T}}_k \in \mathcal{H}} \Big(S_{\text{cov}}(\widehat{\mathcal{T}}_k, \mathcal{M}) + \lambda_{\texttt{joint}} \cdot S_{\text{joint}}(\widehat{\mathcal{T}}_k, \mathcal{H}) \Big). \tag{5}$$

Observe that g in Eq. (5) is invariant to the permutation of the augmentation index k given that the minimum is unique.

In more detail, the coverage score S_{cov} measures how well a predicted skeleton covers the mesh geometry, with lower values indicating better coverage. Given $M = 500$ uniformly sampled surface vertices $\mathcal{V}_{\mathcal{M}} \subset \mathcal{M}$, it is defined as:

$$S_{\text{cov}}(\widehat{\mathcal{T}}_k, \mathcal{M}) = \frac{1}{M} \sum_{\mathbf{p} \in \mathcal{V}_{\mathcal{M}}} -\exp\Big(-\alpha_r \min_{e \in \widehat{\mathcal{E}}_k} \mathcal{D}(\mathbf{p}, e) \Big), \tag{6}$$

where α_r dictates the spatial decay rate. $\mathcal{D}(\mathbf{p}, e)$ is the orthogonal projection distance from a surface vertex $\mathbf{p}$ to a predicted bone $e = (u, v) \in \widehat{\mathcal{E}}_k$, which is:

$$\mathcal{D}(\mathbf{p}, e) = \min_{\gamma \in [0,1]} \big\| \mathbf{p} - \big(j^u + \gamma (j^v - j^u) \big) \big\|_2. \tag{7}$$

The joint-consensus score S_{joint} measures how well one prediction agrees with the others in joint space. For each hypothesis $\widehat{\mathcal{T}}_k$, we compute pairwise Sinkhorn distances (Villani et al., 2009) between its joint set $\widehat{\mathcal{J}}_k$ and the joint sets of the remaining predictions:

$$S_{\text{joint}}(\widehat{\mathcal{T}}_k, \mathcal{H}) = \sum_{\widehat{\mathcal{T}}_i \in \mathcal{H}, \widehat{\mathcal{T}}_k \neq \widehat{\mathcal{T}}_i} \mathcal{D}_{\texttt{sinkhorn}}(\widehat{\mathcal{J}}_k, \widehat{\mathcal{J}}_i). \tag{8}$$

This term favors the predictions that are, on average, closer to the other predictions.

5 Experiments

In Sec. 5.1, we compare our method with prior works. We then ablate the tokenization design and the view-augmented generation strategy in Sec. 5.2. Additional experiments and remaining test-time augmentation ablations are provided in Appendix Sec. A, C, D, and E.

Dataset. For a fair comparison with previous work (Song et al., 2025a), we train our model on the Articulation-XL2.0 dataset, which contains $57k$ samples including the diverse-pose training split. For evaluation, we report results on the Articulation-XL2.0 test set, the diverse-pose test split from Articulation-XL2.0, and the ModelsResource test set (Xu et al., 2019).

Metrics. Following standard metrics in prior work (Song et al., 2025a), we report Chamfer Distance between joints (CD-J2J), Chamfer Distance between joints and bones (CD-J2B), and Chamfer

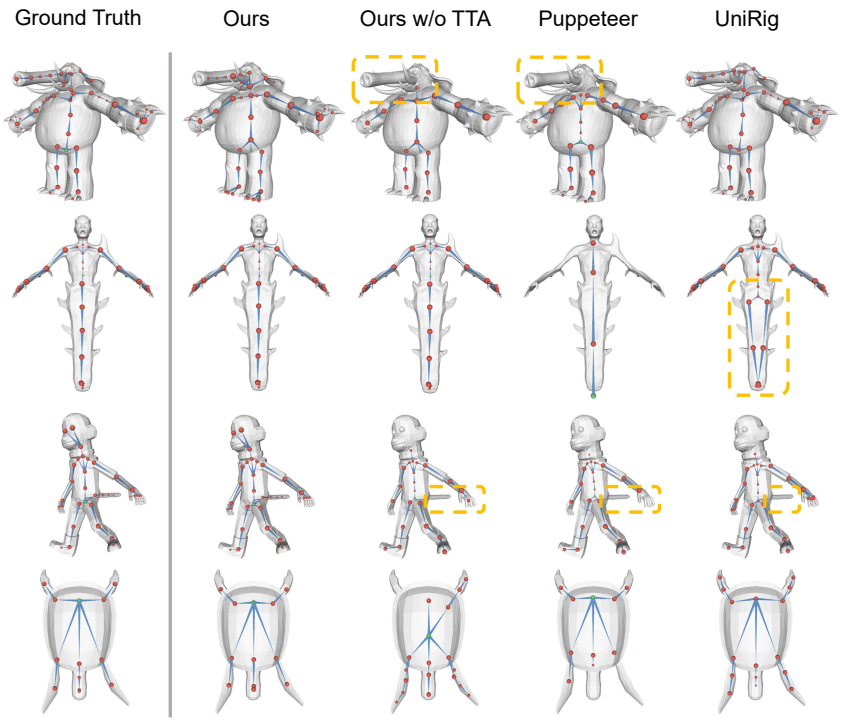


Figure 5: **Qualitative comparison on challenging examples from Articulation-XL2.0.** We compare our method, our method without TTA, Puppeteer (Song et al., 2025a), and UniRig (Zhang et al., 2025) against the ground-truth skeletons. Our method better matches the target connectivity and joint positions, especially around challenging part attachments and elongated structures, while TTA further improves over the variant without TTA. The dashed boxes highlight failure regions.

Distance between bones (CD-B2B). CD-J2J measures the proximity of predicted joints to the ground truth. CD-B2B and CD-J2B capture both joint-level and bone-level alignment.

Baselines. We compare our method against Pinocchio (Baran & Popović, 2007), RigNet (Xu et al., 2020), UniRig (Zhang et al., 2025), MagicArticulate (Song et al., 2025b), Puppeteer (Song et al., 2025a), and Auto-Connect (Guo et al., 2025a). For completeness, we include the results reported by TokenRig (Zhang et al., 2026) as a reference. This comparison is not strictly controlled and may favor TokenRig, since TokenRig is trained on Articulation-XL2.0, VRoid Hub, and ModelsResource. In contrast, our model is trained only on Articulation-XL2.0, following the evaluation protocol of RigNet, UniRig, MagicArticulate, and Puppeteer.

Implementation details. For the network architecture, we match Auto-Connect by using a Michelangelo-based mesh encoder (Zhao et al., 2023), an OPT-style autoregressive decoder (Zhang et al., 2022), and level embeddings for each token. We train our model with Adam for 1000 epochs, using a base learning rate of 1×10^{-4} and a one-cycle learning rate schedule (Smith & Topin, 2019). The global batch size is 96. We use 8-bit quantization, *i.e.*, $n = 8$. For training-time data augmentation, we follow previous work (Zhang et al., 2025; Guo et al., 2025a) and include random rotation, random translation, random scaling, and bone perturbation. For test-time augmentation, we set $\alpha_r = 15$ and $\lambda_{\texttt{joint}} = 1.0$. Training takes approximately 57 hours on four NVIDIA H100 GPUs.

5.1 Results

Tab. 1 reports quantitative comparisons with prior methods on three test sets. Even without TTA, our model already outperforms all directly comparable baselines/methods. Adding TTA further improves all reported metrics, with consistent gains across both the in-domain Articulation-XL2.0 benchmark and the cross-dataset ModelsResource benchmark. We do not report results for Auto-Connect on the diverse-pose split because no official pretrained checkpoint is available.

Table 3: Ablation on test-time augmentation on Articulation-XL2.0. We compare two ways to generate candidate skeletons under a fixed six-hypothesis budget: greedy decoding with $\mathcal{A}_r$ (Ours) and beam search under $\mathcal{A}_{\text{id}}$. We then compare different selection rules over the resulting candidates. The best results are shown in **bold**.

Aug.	Sampling	# Hypo.	Aggregation	J2J ↓	J2B ↓	B2B ↓
$\mathcal{A}_{\text{id}}$	Greedy	1	-	2.438	1.770	1.517
$\mathcal{A}_r$	Greedy	6	Likelihood	2.593	1.901	1.631
$\mathcal{A}_r$	Greedy	6	Length-normalized likelihood	2.474	1.809	1.563
$\mathcal{A}_r$	Greedy	6	Ours	**2.364**	**1.687**	**1.447**
$\mathcal{A}_{\text{id}}$	Beam search	6	Likelihood	2.381	1.739	1.504
$\mathcal{A}_{\text{id}}$	Beam search	6	Length-normalized likelihood	2.377	1.735	1.501
$\mathcal{A}_{\text{id}}$	Beam search	6	Ours	2.375	1.735	1.501

Fig. 5 shows qualitative comparisons with baselines. Our method produces skeletons that better follow the target part layout and local connectivity across diverse shapes, including challenging cases with elongated structures. Comparing our full model with the variant without TTA shows that TTA improves the local structural accuracy of the predictions. In the supplementary website, we provide additional comparisons on in-the-wild meshes.

5.2 Ablations

Ablation on tokenization. Tab. 2 isolates the two components of our tokenization design: branch-centric tokenization (BCT) and distance-aware traversal (DAT). Introducing BCT alone already improves all three metrics on both Articulation-XL2.0 and diverse-pose. This suggests that separating global branch structure from local connection nodes yields a more effective serialization than a naive BFS traversal. Adding DAT further improves performance on top of BCT and gives the best overall results, indicating that a more stable child ordering provides an additional benefit beyond the tokenization structure itself.

Table 2: Ablation on tokenization design choices.

BCT	DAT	Articulation-XL2.0			Diverse-pose		
		J2J	J2B	B2B	J2J	J2B	B2B
		2.543	1.864	1.582	2.216	1.695	1.471
✓		2.479	1.798	1.528	2.145	1.585	1.367
✓	✓	**2.438**	**1.770**	**1.517**	**2.088**	**1.557**	**1.338**

Ablation on TTA. We view TTA for autoregressive skeleton generation as a two-stage procedure: candidate generation followed by final selection. Our method implements these stages with view-augmented generation and coverage-consensus aggregation, respectively. We ablate the design choices of the two stages separately. For candidate generation, we compare our six-view rotation set $\mathcal{A}_r$ against beam search under the identity transformation $\mathcal{A}_{\text{id}}$. For the selection, we compare our coverage-consensus aggregation against two common strategies: sequence likelihood and length-normalized sequence likelihood (Wu et al., 2016). Notably, the combination of beam search and length-normalized likelihood selection corresponds to the inference strategy used in prior work (Zhang et al., 2025). As shown in Tab. 3, coverage-consensus aggregation consistently outperforms likelihood baselines, and combining it with the view augmentation yields the best overall result.

6 Conclusion

We presented **BranchTokens**, a skeleton generation framework built around two design choices: branch-centric tokenization and view-augmented generation. Branch-centric tokenization reorganizes the serialized skeleton by branch structure and, together with distance-aware traversal, yields a more compact and effective representation than standard BFS-style serialization used in previous work. At inference time, view-augmented generation and coverage-consensus aggregation reduce errors caused by axis alignment ambiguity without changing the training objective or network backbone. Experiments show that these design choices consistently improve both joint accuracy and bone alignment over prior methods. Our ablations further support that the gains come from both the tokenization design and the proposed inference procedure.

AI USE STATEMENT

In this work, we used generative AI tools for:

- Implementing methods.
- Providing feedback on research methodology.
- Proposing hypotheses.
- Generating synthetic data sets for evaluation.

We have not used generative AI tools for

- Cleaning and reformatting datasets.
- Interpreting results.

The following are not applicable to this work:

- Providing critical ingredients for proving mathematical claims.
- Formulating mathematical claims.
- Assisting with translation.
- Developing theoretical/conceptual frameworks.
- Assisting in writing proofs.

Additionally, we used generative AI tools for editing software code, analyzing existing literature, and drafting parts of a research paper. We have reviewed all AI-assisted work. We have read all LLM-searched literature and reviewed the code for correctness. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Xiang Bai and Longin Jan Latecki. Path similarity skeleton graph matching. *PAMI*, 2008. 2, 5

Xiang Bai, Xinggang Wang, Longin Jan Latecki, Wenyu Liu, and Zhuowen Tu. Active skeleton for non-rigid object detection. In *Proc. ICCV*, 2009. 2, 5

Ilya Baran and Jovan Popović. Automatic rigging and animation of 3D characters. *TOG*, 2007. 2, 7, 8, 17, 18

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Proc. NeurIPS*, 2020. 1

Zedong Chu, Feng Xiong, Meiduo Liu, Jinzhi Zhang, Mingqi Shao, Zhaoxu Sun, Di Wang, and Mu Xu. HumanRig: Learning automatic rigging for humanoid character in a large scale dataset. In *Proc. CVPR*, 2025. 2

Nicu D Cornea, Deborah Silver, and Patrick Min. Curve-skeleton properties, applications, and algorithms. *TVCG*, 2024. 2

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. *Advances in Neural Information Processing Systems*, 2023. 17

Yufan Deng, Yuhao Zhang, Chen Geng, Shangzhe Wu, and Jiajun Wu. AnyMate: A dataset and baselines for learning 3D object rigging. In *Proc. SIGGRAPH*, 2025. 1, 3, 17, 18

Pedro F Felzenszwalb and Daniel P Huttenlocher. Pictorial structures for object recognition. *IJCV*, 2005. 2

Andrew Feng, Dan Casas, and Ari Shapiro. Avatar reshaping and automatic rigging using a deformable model. In *Proc. MIG*, 2015. 2

Joseph Gallian. *Contemporary abstract algebra*. Chapman and Hall/CRC, 2021. 16

Jonathan L Gross, Jay Yellen, and Mark Anderson. *Graph theory and its applications*. Chapman and Hall/CRC, 2018. 5

Jingfeng Guo, Jian Liu, Jinnan Chen, Shiwei Mao, Changrong Hu, Puhua Jiang, Junlin Yu, Jing Xu, Qi Liu, Lixin Xu, et al. Auto-Connect: Connectivity-preserving RigFormer with direct preference optimization. *arXiv preprint arXiv:2506.11430*, 2025a. 1, 2, 3, 4, 7, 8, 20

Zhiyang Guo, Jinxu Xiang, Kai Ma, Wengang Zhou, Houqiang Li, and Ran Zhang. Make-It-Animatable: An efficient framework for authoring animation-ready 3D characters. In *Proc. CVPR*, 2025b. 2

Urvashi Khandelwal, He He, Peng Qi, and Dan Jurafsky. Sharp nearby, fuzzy far away: How neural language models use context. In *Proc. ACL*, 2018. 6

Ildoo Kim, Younghoon Kim, and Sungwoong Kim. Learning loss for test-time augmentation. In *Proc. NeurIPS*, 2020. 3

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Proc. NeurIPS*, 2012. 3

Alexander K Lew, Tan Zhi-Xuan, Gabriel Grand, and Vikash K Mansinghka. Sequential monte carlo steering of large language models using probabilistic programs. *arXiv preprint arXiv:2306.03081*, 2023. 18

Conglong Li, Minjia Zhang, and Yuxiong He. The stability-efficiency dilemma: Investigating sequence length warmup for training gpt models. In *Proc. NeurIPS*, 2022. 6

Peizhuo Li, Kfir Aberman, Rana Hanocka, Libin Liu, Olga Sorkine-Hornung, and Baoquan Chen. Learning skeletal articulations with neural blend shapes. *TOG*, 2021. 2

Isabella Liu, Zhan Xu, Wang Yifan, Hao Tan, Zexiang Xu, Xiaolong Wang, Hao Su, and Zifan Shi. RigAnything: Template-free autoregressive rigging for diverse 3D assets. *TOG*, 2025. 1, 3

Alexander Lyzhov, Yuliya Molchanova, Arsenii Ashukha, Dmitry Molchanov, and Dmitry Vetrov. Greedy policy search: A simple baseline for learnable test-time augmentation. In *Proc. UAI*, 2020. 3

Jing Ma and Dongliang Zhang. TARig: Adaptive template-aware neural rigging for humanoid characters. *Comput. Graph.*, 2023. 2

Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005*, 2017. 14

Ofir Press, Noah A Smith, and Mike Lewis. Shortformer: Better language modeling using shorter inputs. In *Proc. ACL-IJCNLP*, 2021. 6

Zihan Qin, Mingze Sun, Yifan Mao, Jialei Xu, Jingfeng Guo, Changrong Hu, Wenbo Zhao, Junjun Jiang, and Xianming Liu. Vip-rig: Visual-prompted controllable rigging. *arXiv preprint arXiv:2607.27982*, 2026. 3

Divya Shanmugam, Davis Blalock, Guha Balakrishnan, and John Guttag. Better aggregation in test-time augmentation. In *Proc. ICCV*, 2021. 3

Kaleem Siddiqi, Ali Shokoufandeh, Sven J Dickinson, and Steven W Zucker. Shock graphs and shape matching. *IJCV*, 1999. 2

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 3

Leslie N Smith and Nicholay Topin. Super-convergence: Very fast training of neural networks using large learning rates. In *Proc. SPIE*, 2019. 8

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*, 2025a. 1, 2, 3, 7, 8, 14

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 *Proc. CVPR*, 2025b. 1, 3, 6, 7, 8

Mingze Sun, Junhao Chen, Junting Dong, Yurun Chen, Xinyu Jiang, Shiwei Mao, Puhua Jiang, Jingbo Wang, Bo Dai, and Ruqi Huang. Drive: Diffusion-based rigging empowers generation of versatile and expressive characters. In *Proc. CVPR*, 2025a. 2

Mingze Sun, Shiwei Mao, Keyi Chen, Yurun Chen, Shunlin Lu, Jingbo Wang, Junting Dong, and Ruqi Huang. Armo: Autoregressive rigging for multi-category objects. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7721–7730. IEEE, 2025b. 3

Mingze Sun, Cheng Zeng, Jiansong Pei, Junhao Chen, Chaoyue Song, Shaohui Wang, Tianyuan Chang, Bin Huang, Zijiao Zeng, and Ruqi Huang. Animator-centric skeleton generation on objects with fine-grained details. *arXiv preprint arXiv:2604.20539*, 2026. 3

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proc. CVPR*, 2015. 3

Andrea Tagliasacchi, Hao Zhang, and Daniel Cohen-Or. Curve skeleton extraction from incomplete point cloud. In *Proc. SIGGRAPH*, 2009. 2

Andrea Torsello and Edwin R Hancock. Learning shape-classes using a mixture of tree-unions. *PAMI*, 2006. 2

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 1

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. NeurIPS*, 2017. 1, 3

Cédric Villani et al. *Optimal transport: old and new*, volume 338. Springer, 2009. 7

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016. 3, 9

Zhan Xu, Yang Zhou, Evangelos Kalogerakis, and Karan Singh. Predicting animation skeletons for 3D articulated models via volumetric nets. In *Proc. 3DV*, 2019. 1, 2, 7

Zhan Xu, Yang Zhou, Evangelos Kalogerakis, Chris Landreth, and Karan Singh. RigNet: Neural rigging for articulated characters. *TOG*, 2020. 1, 2, 7, 8, 17, 18

Zhan Xu, Yang Zhou, Li Yi, and Evangelos Kalogerakis. MoRig: Motion-aware rigging of character meshes from point clouds. In *Proc. SIGGRAPH Asia*, 2022. 2

Yajie Yan, Kyle Sykes, Erin Chambers, David Letscher, and Tao Ju. Erosion thickness on medial axes of 3d shapes. *TOG*, 2016. 2

Yajie Yan, David Letscher, and Tao Ju. Voxel cores: Efficient, robust, and provably good approximation of 3D medial axes. *TOG*, 2018. 2

Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-BERT: Pre-training 3D point cloud transformers with masked point modeling. In *Proc. CVPR*, 2022. 1

Jia-Peng Zhang, Cheng-Feng Pu, Meng-Hao Guo, Yan-Pei Cao, and Shi-Min Hu. One model to rig them all: Diverse skeleton rigging with UniRig. *TOG*, 2025. 1, 2, 3, 7, 8, 9, 14, 17, 18

Jia-peng Zhang, Cheng-Feng Pu, Meng-Hao Guo, Yan-Pei Cao, and Shi-Min Hu. Skin Tokens: A learned compact representation for unified autoregressive rigging. *arXiv preprint arXiv:2602.04805*, 2026. 1, 2, 3, 7, 8, 14, 20

Kaizhong Zhang and Dennis Shasha. Simple fast algorithms for the editing distance between trees and related problems. *SIAM journal on computing*, 18(6):1245–1262, 1989. 15

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. OPT: Open pre-trained transformer language models, 2022. 1, 2, 8

Ruisi Zhao, Haoren Zheng, Zongxin Yang, Hehe Fan, and Yi Yang. Stroke3D: Lifting 2D strokes into rigged 3D model via latent diffusion models. *arXiv preprint arXiv:2602.09713*, 2026. 2, 18

Zibo Zhao, Wen Liu, Xin Chen, Xianfang Zeng, Rui Wang, Pei Cheng, BIN FU, Tao Chen, Gang Yu, and Shenghua Gao. Michelangelo: Conditional 3D shape generation based on shape-image-text aligned latent representation. In *Proc. NeurIPS*, 2023. 1, 8

A Additional In-the-Wild Mesh Comparisons

In addition to the representative examples shown in the main paper, we provide a supplementary website with more qualitative comparisons on in-the-wild meshes against three strong baselines: TokenRig (Zhang et al., 2026), Puppeteer (Song et al., 2025a), and UniRig (Zhang et al., 2025). The website is included in the submission zip and contains side-by-side visualizations across diverse shapes and poses, allowing closer inspection of local connectivity, part layout, and failure cases. We present these results in website form because the number of examples is too large to fit comfortably in the PDF supplementary material. Fig. 6 shows a preview of the website.

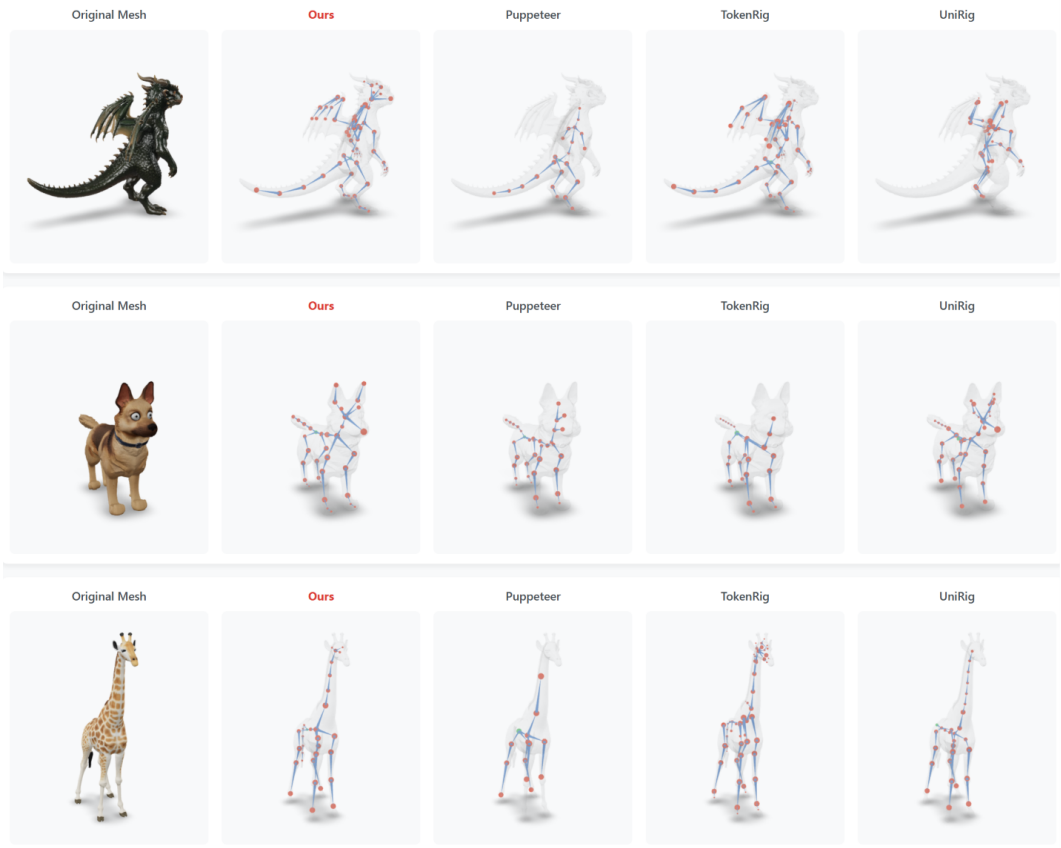


Figure 6: **Preview of the supplementary website.** Each row shows an in-the-wild mesh alongside the predicted skeletons from our method, Puppeteer (Song et al., 2025a), TokenRig (Zhang et al., 2026), and UniRig (Zhang et al., 2025). Our method generates complete skeletal structures covering the head, tail, and body without introducing redundant bones. The full website contains additional examples spanning diverse shapes, including fantasy creatures, animals, and humans.

B Branch-Centric Tokenization Algorithm

Algorithm 1 summarizes BCT. We highlight in green the steps that differ from the tokenization method proposed by Auto-Connect.

C Evaluation with Structural Metrics

We further evaluate the structural similarity of the predicted skeletons using Graph2Vec distance (Narayanan et al., 2017) and tree edit distance (TED). We train the Graph2Vec model on skeletons from the Articulation-XL2.0 training set and compute the cosine distance between the

Algorithm 1 Branch-Centric Tokenization (BCT)

Require: Skeleton tree $\mathcal{T} = (\mathcal{V}, \mathcal{E})$ with root v^{root}, quantization bit-width n
Ensure: Serialized token sequence $\boldsymbol{\tau}$
  Define APPENDCOORD(v) to append $(Q_n(j_x^v), Q_n(j_y^v), Q_n(j_z^v))$ to $\boldsymbol{\tau}$
  Construct the series-reduced tree $\mathcal{T}_{\texttt{sr}}$ from (4).
  Initialize $\boldsymbol{\tau} \leftarrow$ [BOS]
  APPENDCOORD(v^{root});
  Append [E2] to $\boldsymbol{\tau}$
  $Q \leftarrow$ Queue$([v^{\text{root}}])$
  while Q is not empty **do**
    $L \leftarrow$ length(Q)
    for $i = 1$ **to** L **do**
      $p \leftarrow Q$.dequeue()
      Let $\mathcal{V}_{\texttt{child}}$ be the children of p in $\mathcal{T}_{\texttt{sr}}$, sorted by DAT in ascending Euclidean distance to p
      for all $v \in \mathcal{V}_{\texttt{child}}(p)$ **do**
        APPENDCOORD(v)
        Let $\mathcal{S}_{v \to p}$ be the connection nodes on the path from v to p in $\mathcal{T}$, ordered from v toward p
        for all $c \in \mathcal{S}_{v \to p}$ **do**
          APPENDCOORD(c)
        end for
        Append [E3] to $\boldsymbol{\tau}$
        Q.push(v)
      end for
      Append [E1] to $\boldsymbol{\tau}$
    end for
    Append [E2] to $\boldsymbol{\tau}$
  end while
  Append [EOS] to $\boldsymbol{\tau}$
  return $\boldsymbol{\tau}$

Table 4: **Quantitative comparison of skeleton prediction methods on Articulation-XL2.0.** We report TED and Graph2Vec distance on the Articulation-XL2.0 test set. **Bold** numbers denote the best result in each column.

Method	TED ↓	Graph2Vec distance ↓
MagicArti.	6.837	0.254
Puppeteer	6.736	0.245
TokenRig	7.120	0.235
Ours	**5.984**	**0.222**

embeddings of each predicted skeleton and its ground-truth counterpart. For TED, we order the children of each node by the size of their corresponding subtrees and compute the distance using the Zhang–Shasha algorithm (Zhang & Shasha, 1989). As shown in Tab. 4, our model achieves the lowest TED and Graph2Vec distance among all compared methods.

D TEST-TIME AUGMENTATION ABLATIONS

We provide additional ablations of the test-time augmentation strategy used in our method. We focus on four questions. First, does a larger axis-aligned rotation group help? Second, is coverage-consensus aggregation more effective than likelihood baselines under common stochastic sampling? Third, which augmentation choices are effective? Finally, how sensitive is coverage-consensus aggregation to its hyperparameters?

Table 5: Ablation on a larger rotation group for view-augmented generation on Articulation-XL2.0. We compare the six-view rotation set $\mathcal{A}_r$ used in the main paper with the larger 24-view cube rotation set $\mathcal{A}_{\text{cube}}$. The larger set gives a small accuracy gain at the cost of four times more decoded hypotheses.

Aug.	Sampling	# Hypo.	J2J ↓	J2B ↓	B2B ↓
$\mathcal{A}_r$	Greedy	6	2.364	1.687	1.447
$\mathcal{A}_{\text{cube}}$	Greedy	24	**2.339**	**1.673**	**1.438**

Table 6: Comparison of selection strategies for stochastic samples without view-augmented generation on Articulation-XL2.0. The first row reports greedy decoding under $\mathcal{A}_r$ for reference. For stochastic samples under $\mathcal{A}_{\text{id}}$, coverage-consensus aggregation outperforms likelihood-based selection.

Aug.	Sampling	# Hypo.	Aggregation	J2J ↓	J2B ↓	B2B ↓
$\mathcal{A}_r$	Greedy	6	Coverage-consensus aggregation	**2.364**	**1.687**	**1.447**
$\mathcal{A}_{\text{id}}$	Stochastic	6	Likelihood	2.550	1.884	1.612
$\mathcal{A}_{\text{id}}$	Stochastic	6	Length-normalized likelihood	2.441	1.779	1.524
$\mathcal{A}_{\text{id}}$	Stochastic	6	Coverage-consensus aggregation	2.408	1.734	1.487

D.1 Alternative Rotation Set

We evaluate a larger 24-element cube rotation group $\mathcal{A}_{\text{cube}}$, which enumerates all right-handed axis-aligned rotations (Gallian, 2021) for view-augmented generation. Compared with the six-view set $\mathcal{A}_r$ used in the main paper, this larger set does not rely on prior knowledge of common upright orientations.

Results. Tab. 5 shows that expanding the rotation set from $\mathcal{A}_r$ to $\mathcal{A}_{\text{cube}}$ further improves performance, but only modestly, while requiring four times as many decoded hypotheses. We therefore use the smaller set $\mathcal{A}_r$ in the main method because it provides a better accuracy-efficiency trade-off.

D.2 Alternative Sampling Method

To supplement Tab. 3, we study a second common way to generate candidate skeletons: stochastic sampling under the identity transformation $\mathcal{A}_{\text{id}}$. We then compare coverage-consensus aggregation against standard likelihood-based selection rules on the resulting samples. As shown in Tab. 6, coverage-consensus aggregation outperforms both likelihood-based baselines in this setting. For reference, the table also reports view-augmented greedy decoding under $\mathcal{A}_r$, which achieves the best overall performance.

D.3 Augmentation Choice

We next fix the decoding strategy to greedy decoding and study which augmentations are useful for view-augmented generation. Throughout this experiment, final selection is performed using our coverage-consensus aggregation.

Augmentations. We compare three augmentation choices: the six-view rotation set $\mathcal{A}_r$ defined in Sec. 4.2, random translation jitter, and random scaling. We evaluate each augmentation individually and apply it globally to both the mesh and the skeleton to preserve equivariance. For random translation jitter, we first normalize each sample into $[-1.0, 1.0]^3$ and apply a deterministic margin scale of 0.9. We then sample an axis-wise shift from $[-0.1, 0.1]$ so that the translated sample remains within the valid spatial range. For random scaling, we sample an isotropic scale factor from $[0.5, 1.0]$.

Results. Table 7 shows that under coverage-consensus aggregation, the rotation set $\mathcal{A}_r$ is the strongest option. In contrast, random translation jitter and random scaling substantially degrade performance, with random scaling causing the largest drop.

Table 7: Additional ablation on test-time augmentation. Each augmentation is evaluated individually using our coverage-consensus aggregation.

Augmentation	J2J ↓	J2B ↓	B2B ↓
Rotation set ($\mathcal{A}_r$)	**2.364**	**1.687**	**1.447**
Random translation	4.752	3.128	2.202
Random scaling	8.291	5.324	3.113

Table 8: Hyperparameter sensitivity of coverage-consensus aggregation on Articulation-XL2.0. We report CD-J2J over the spatial decay rate α_r and the joint-consensus weight $\lambda_{\texttt{joint}}$ under the 6-hypothesis rotation TTA setting with $\mathcal{A}_r$. Bold indicates the best result.

α_r	Joint-consensus weight $\lambda_{\texttt{joint}}$					
	0.00	0.25	0.50	1.00	2.00	4.00
5.0	2.516	2.402	2.395	2.366	2.364	2.381
10.0	2.481	2.393	2.377	2.364	2.367	2.381
15.0	2.478	2.378	2.371	2.364	2.368	2.385
20.0	2.472	2.377	2.372	**2.363**	2.374	2.383
25.0	2.469	2.375	2.374	2.364	2.377	2.387

D.4 Hyperparameter Sensitivity of Coverage-Consensus Aggregation

We finally study the sensitivity of coverage-consensus aggregation to its two main hyperparameters: the spatial decay rate α_r and the joint-consensus weight $\lambda_{\texttt{joint}}$.

Results. Table 8 shows that the joint-consensus term is important. Using only the coverage term gives consistently worse results across all values of α_r. In contrast, introducing a nonzero joint-consensus weight yields a clear improvement. Once $\lambda_{\texttt{joint}}$ is set to a moderate value, performance is fairly stable. We use $\alpha_r = 15.0$ and $\lambda_{\texttt{joint}} = 1.0$ in the main experiments.

E Additional Experiment on the Anymate Dataset

We additionally train our model on the Anymate dataset (Deng et al., 2025). Following Auto-Connect, we assume that the skeleton forms a singly connected tree, and therefore exclude training samples with self-loops or multiple connected components, resulting in a clean training set of $205k$ samples. For evaluation, we use the original Anymate test set to ensure a fair comparison. Following the original paper, we report Chamfer Distance (CD) and Earth Mover's Distance (EMD), where EMD measures the optimal transport cost between predicted joints and ground-truth joints. We compare against Pinocchio (Baran & Popović, 2007), RigNet (Xu et al., 2020), Anymate (Deng et al., 2025), and UniRig (Zhang et al., 2025). Among them, Anymate directly predicts joint positions without autoregressive generation. UniRig is retrained on Anymate using the official codebase for a fair comparison.

For our method, all hyperparameters are the same as in the main text, except that we train for only 300 epochs because of the larger dataset size. Training takes approximately 61 hours. We do not directly evaluate the model trained on Articulation-XL2.0, because Anymate and Articulation-XL2.0 are both sourced from Objaverse-XL (Deitke et al., 2023) and may therefore have test-set overlap. As shown in Tab. 9, our method achieves the lowest CD and EMD among the compared methods.

F Inference Time Comparison

In Tab. 10, we compare inference time with prior autoregressive methods. Our model without test-time augmentation (TTA) is faster than UniRig, Puppeteer, and TokenRig. With TTA, inference time increases as expected because we decode six rotated views, but the overall runtime remains

Table 9: Quantitative results on the Anymate dataset (Deng et al., 2025). Our method achieves the best CD and EMD.

Method	CD ↓	EMD ↓
Pinocchio (Baran & Popović, 2007)	0.198	0.659
RigNet (Xu et al., 2020)	0.089	0.127
Anymate (Deng et al., 2025)	0.077	0.098
UniRig (Zhang et al., 2025)	0.096	0.135
Ours	**0.038**	**0.046**

Table 10: Inference time comparison on Articulation-XL2.0 test set. Measured in seconds.

Method	UniRig	Puppeteer	TokenRig	Ours w/o TTA	Ours
Inference time	1.8	1.3	7.5	**1.0**	5.8

moderate. For all baselines, we use the official code, exclude data processing time, and run inference on a single NVIDIA H100 GPU.

G Qualitative Application: Fixed-Template Skeleton Generation with a Mixamo-Style Hierarchy

We include this example as a qualitative application of fixed-template constrained decoding under a predefined 22-joint Mixamo-style hierarchy. In this setting, the structural part of the sequence is fixed by the template, and the decoder predicts only the joint coordinates.

The structural symbols in this section follow the same token convention used in the paper. Specifically, `[E3]` terminates one local branch, `[E1]` terminates the child sequence of one parent, and `[E2]` terminates one BFS level. Let $\langle C \rangle$ denote one coordinate triplet. Under the fixed Mixamo-style topology, the structural sequence is uniquely determined by the level-wise BCT expansion order, while the model fills only the coordinate slots under the corresponding constrained support. For readability, we display the static BCT token template induced by the fixed Mixamo-style hierarchy in level-wise order in Fig. 7.

This fixed BCT token template makes the decoder a fixed-template constrained generator rather than a dynamic topology generator. At structural positions, the controller forces the unique valid structural token, while at coordinate positions it masks out all structural tokens and allows the model to predict only coordinate bins. For this application, we combine the constrained support with Sequential Monte Carlo (SMC) steering (Lew et al., 2023): each weighted decoding hypothesis follows the same template controller, samples from the masked distribution, and is resampled at structural boundaries such as branch and level terminators. The final prediction is selected as the highest-weight hypothesis after `[EOS]`. The joint identities and parent relations are defined by the fixed Mixamo-style hierarchy in Tab. 11, while the token sequence follows the corresponding level-wise BCT expansion order.

As a qualitative demonstration, we apply our model, pre-trained on Articulation-XL2.0, to in-the-wild human meshes. As shown in Fig. 8, the predicted skeletons preserve the fixed Mixamo topology across diverse poses while adapting the joint locations to each mesh.

H Limitations

Our method has two main limitations. First, our view augmentation design is motivated by the fact that meshes are not consistently canonicalized to a shared upright orientation. We follow the same evaluation protocol as prior baselines, which operate on meshes in their original orientations. Stroke3D (Zhao et al., 2026) highlighted this issue and attempted to address it by aligning meshes to a consistent upright direction. However, its upright-detection system is not publicly available, so we could not evaluate our method and prior baselines under a setting where all meshes are canonicalized to a consistent z-upright orientation. If a reliable orientation-canonicalization method becomes

Static BCT Token Template

[BOS] ⟨C⟩	– *Root: Hip*
[E2]	– *End Level 0*
Level 1	
⟨C⟩⟨C⟩⟨C⟩	– *Spine branch (Spine02, Spine01, Waist)*
[E3]	
⟨C⟩⟨C⟩⟨C⟩⟨C⟩	– *Left leg branch (L_Toe0, L_Foot, L_Calf, L_Thigh)*
[E3]	
⟨C⟩⟨C⟩⟨C⟩⟨C⟩	– *Right leg branch (R_Toe0, R_Foot, R_Calf, R_Thigh)*
[E3]	
[E1]	– *End Hip's child sequence*
[E2]	– *End Level 1*
Level 2	
⟨C⟩⟨C⟩	– *Neck branch (Head, NeckTwist01)*
[E3]	
⟨C⟩⟨C⟩⟨C⟩⟨C⟩	– *Left arm branch (L_Hand, L_Forearm, L_Upperarm, L_Clavicle)*
[E3]	
⟨C⟩⟨C⟩⟨C⟩⟨C⟩	– *Right arm branch (R_Hand, R_Forearm, R_Upperarm, R_Clavicle)*
[E3]	
[E1]	– *End Spine02's child sequence*
[E1]	– *End L_Toe0's child sequence*
[E1]	– *End R_Toe0's child sequence*
[E2]	– *End Level 2*
Level 3	
[E1]	– *End Head's child sequence*
[E1]	– *End L_Hand's child sequence*
[E1]	– *End R_Hand's child sequence*
[E2]	– *End Level 3*
[EOS]	

Figure 7: Static BCT token template induced by the fixed 22-joint Mixamo-style hierarchy.

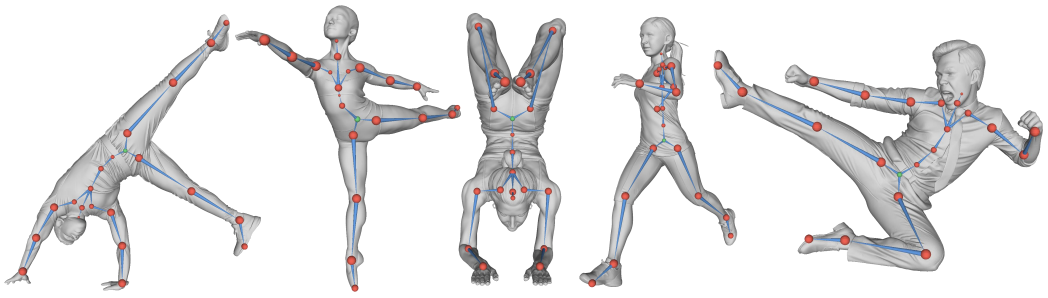


Figure 8: Qualitative application of fixed-template constrained decoding under a 22-joint Mixamo-style template. We apply our model, pre-trained on Articulation-XL2.0, to in-the-wild human meshes generated from diverse prompts. The predicted skeletons preserve the fixed topology across diverse poses while adapting the joint locations to each input shape.

available in the future, the need for the view augmentation component of our TTA design may be reduced. That said, Tab. 6 shows that our coverage-consensus aggregation remains effective even when the hypotheses are not generated by view augmentation, which suggests that it is still useful beyond this particular assumption.

Table 11: Joint indexing under the fixed Mixamo-style hierarchy.

Index	Joint name	Parent joint
0	Hip	None
1	Waist	Hip
2	Spine01	Waist
3	Spine02	Spine01
4	NeckTwist01	Spine02
5	Head	NeckTwist01
6	L_Clavicle	Spine02
7	L_Upperarm	L_Clavicle
8	L_Forearm	L_Upperarm
9	L_Hand	L_Forearm
10	R_Clavicle	Spine02
11	R_Upperarm	R_Clavicle
12	R_Forearm	R_Upperarm
13	R_Hand	R_Forearm
14	L_Thigh	Hip
15	L_Calf	L_Thigh
16	L_Foot	L_Calf
17	L_Toe0	L_Foot
18	R_Thigh	Hip
19	R_Calf	R_Thigh
20	R_Foot	R_Calf
21	R_Toe0	R_Foot

Second, we do not study post-training in this paper. Prior work, including TokenRig (Zhang et al., 2026) and Auto-Connect (Guo et al., 2025a), has shown that post-training can be effective for improving skeleton generation. We intentionally exclude it here because our goal is to study the effects of branch-centric tokenization and test-time augmentation in a clean, controlled setting. An important future direction is to combine our method with post-training and test whether the gains are complementary.