Puppeteer: Rig and Animate Your 3D Models

Chaoyue Song[1,2], Xiu Li[2], Fan Yang[1], Zhongcong Xu[2], Jiacheng Wei[1], Fayao Liu[3], Jiashi Feng[2], Guosheng Lin[†1], Jianfeng Zhang[†2]
[1]Nanyang Technological University [2]ByteDance Seed
[3]Institute for Infocomm Research, A*STAR
https://chaoyuesong.github.io/Puppeteer

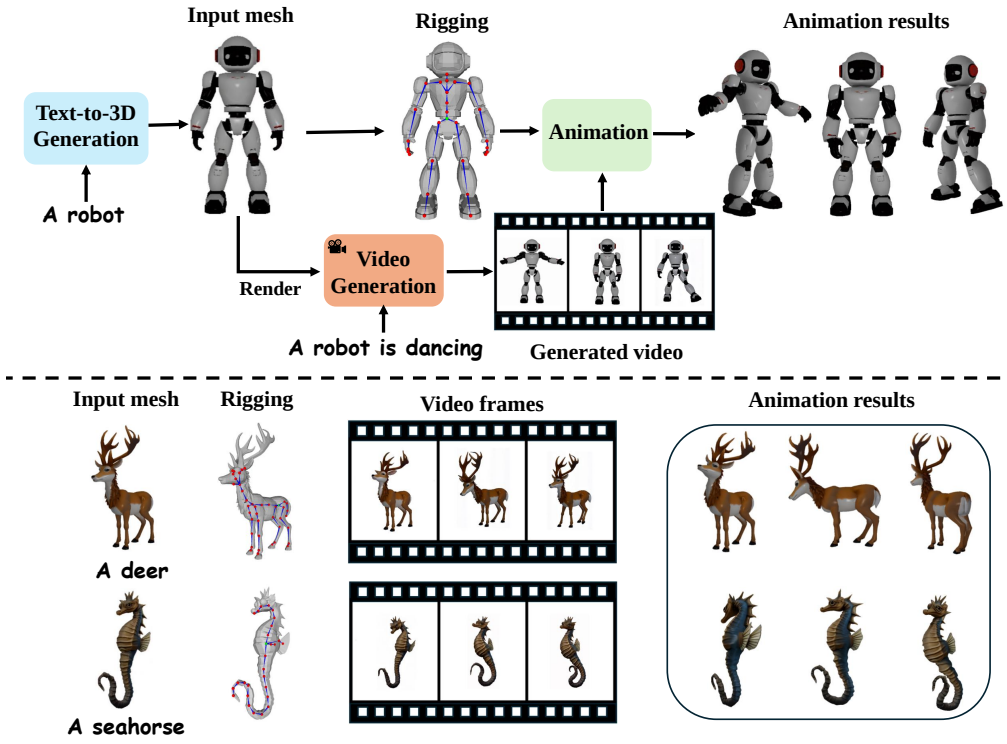


Figure 1: Given a 3D model, we apply automatic rigging to create a skeleton structure with skinning weights. The input mesh is then rendered as input for video generation models [1, 34]. Finally, we produce animations guided by the generated videos. The input 3D models are generated by [73].

Abstract

Modern interactive applications increasingly demand dynamic 3D content, yet the transformation of static 3D models into animated assets constitutes a significant bottleneck in content creation pipelines. While recent advances in generative AI have revolutionized static 3D model creation, rigging and animation continue to depend heavily on expert intervention. We present **Puppeteer**, a comprehensive framework that addresses both automatic rigging and animation for diverse 3D objects. Our system first predicts plausible skeletal structures via an auto-regressive transformer that introduces a joint-based tokenization strategy for compact representation and a hierarchical ordering methodology with stochastic perturbation that enhances bidirectional learning capabilities. It then infers skinning weights via an attention-based

[†] Corresponding authors. Email: chaoyue002@e.ntu.edu.sg.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

architecture incorporating topology-aware joint attention that explicitly encodes inter-joint relationships based on skeletal graph distances. Finally, we complement these rigging advances with a differentiable optimization-based animation pipeline that generates stable, high-fidelity animations while being computationally more efficient than existing approaches. Extensive evaluations across multiple benchmarks demonstrate that our method significantly outperforms state-of-the-art techniques in both skeletal prediction accuracy and skinning quality. The system robustly processes diverse 3D content, ranging from professionally designed game assets to AI-generated shapes, producing temporally coherent animations that eliminate the jittering issues common in existing methods.

1 Introduction

From AAA games and animated films to VR/AR experiences and robotic simulations, modern interactive media demands dynamic 3D content. While recent generative AI advances have accelerated the creation of high-fidelity 3D models with intricate geometry and textures, these assets remain predominantly static. Transforming static 3D models into animated versions requires two expert-driven processes: rigging (skeleton setup and skinning weight assignment) and animation. This manual, time-intensive workflow now constitutes a significant impediment to the efficiency of modern content creation pipelines.

Research communities have invested considerable effort in automating the rigging process. Early template-based techniques such as Pinocchio [6] fit predefined skeletal structures to input meshes, achieving satisfactory results on specific categories but failing to generalize to arbitrary shapes. Template-free algorithms [25, 3, 7, 42, 71] extract skeletal structures directly from geometric properties but frequently produce excessively dense or topologically incompatible joint configurations unsuitable for practical animation workflows. Deep learning approaches have substantially advanced the field: RigNet [86] pioneered direct skeleton and skinning weight prediction from input shapes using graph neural networks, while MagicArticulate [67] reformulated skeleton generation as an auto-regressive problem and introduced a large-scale dataset with detailed rigging annotations. Despite these innovations, significant challenges persist: RigNet struggles with complex mesh topologies due to its reliance on carefully crafted features and restrictive orientation requirements. MagicArticulate suffers from computational inefficiency during inference and limited generalization in its functional diffusion process for skinning weight prediction. Critically, both approaches address only the rigging stage of the pipeline, leaving the equally challenging animation process as a separate manual task that requires substantial expertise.

In this work, we present **Puppeteer**, a comprehensive framework that integrates automatic rigging and animation into a unified pipeline. To address the data scarcity and limited pose diversity in existing datasets, we expand the Articulation-XL dataset [67] to 59.4k rigged models, including a carefully curated subset of 11.4k diverse pose examples that enhance generalization to varied pose inputs. This expanded dataset serves as the foundation for our learning-based approach. To overcome the limitations of existing rigging approaches in handling diverse shapes and complex topologies, our system introduces key improvements to both fundamental rigging components. For skeleton generation, we employ auto-regressive transformers featuring joint-based tokenization and hierarchical sequence ordering with randomization, creating more compact representations while generating structurally coherent skeletons free from template dependencies. For skinning weight prediction, we propose an attention-based architecture incorporating topology-aware joint attention that explicitly encodes skeletal graph structure, achieving robust weight prediction with enhanced generalization and computational efficiency. Beyond rigging, we address the automatic animation challenge that previous methods have largely overlooked. We introduce a differentiable optimization-based method that requires no neural network parameters yet produces stable, high-quality animations by combining our generated rigging with reference video guidance easily obtained from off-the-shelf video generation models. Our unified framework enables full automation from static meshes to animated assets, transforming the labor-intensive manual workflow into an efficient, accessible pipeline for diverse 3D content creation.

Extensive evaluations demonstrate the effectiveness of our approach across both rigging and animation tasks. For rigging, experiments on the expanded Articulation-XL2.0 dataset and ModelsResource benchmark [76, 85] show significant improvements over state-of-the-art methods in skeleton accuracy and skinning weight quality. The robustness of our approach is further validated through successful

application to diverse 3D content—from professionally designed game assets to AI-synthesized geometries. For animation, direct comparisons against recent 4D generation techniques [77, 59] show that our optimization-based approach produces more temporally consistent and visually faithful results while maintaining computational efficiency. Notably, our method eliminates the jittering artifacts commonly seen in learning-based approaches during complex motion sequences. The clean and stable animation results also highlight the reliability of our automatically generated rigs.

In summary, our work advances automated 3D model rigging and animation through four key contributions: (1) An expanded large-scale articulation dataset with 59.4k rigged models including a diverse-pose subset; (2) A novel auto-regressive skeleton generation approach featuring efficient joint-based tokenization and hierarchical sequence ordering with randomization strategies; (3) An attention-based architecture for skinning weight prediction incorporating topology-aware joint attention; and (4) A differentiable optimization-based animation method that produces stable, high-quality animation for diverse object categories without requiring extensive computational resources or manual effort.

2 Related works

We discuss the related works in automatic rigging here, while related works in 3D animation can be found in the appendix.

Skeleton generation. Skeleton generation methods for 3D models fall into two main groups. The first leverages templates or additional inputs. Pinocchio [6] pioneered template-fitting for automatic skeleton extraction, while Li et al. [37] employed deep learning for human joint estimation with a given skeleton template. Some recent works [13, 24, 69] continue this line for humanoid skeleton generation. A significant limitation of these approaches is their inability to generalize effectively to diverse object categories. Other methods in this group require additional inputs such as point cloud sequences [87], mesh sequences [14, 30], manual annotations [26], or video data [88, 81, 66, 98, 90, 65, 68, 40]. The second group works without templates or annotations. Traditional approaches [3, 7, 25, 71, 42] extract curve skeletons and often produce overly dense joints unsuitable for animation. Modern deep learning methods like Xu et al. [85] and RigNet [86] learn directly from limited datasets containing fewer than 3,000 rigged models. Despite their innovations, these methods depend extensively on carefully crafted features and impose restrictive assumptions regarding shape orientation, substantially constraining their effectiveness when confronted with complex mesh topologies.

With the exponential growth of 3D datasets [15, 16] and the success of auto-regressive approaches in 3D generation [62, 11, 12, 72], the field has seen significant advances in skeleton generation. MagicArticulate [67] pioneered the formulation of skeleton generation as an auto-regressive problem and introduced Articulation-XL, a large-scale 3D dataset with rigging information. Several recent works [45, 100] have also successfully incorporated auto-regressive transformer architectures for skeleton generation, further validating this approach. In our work, we substantially expand the Articulation-XL dataset from 33k to 59.4k rigged models, including a diverse pose subset containing 11.4k examples. We leverage auto-regressive transformers for skeleton generation, introducing two key innovations: an efficient tokenization method for skeletal structures and a hierarchical sequence ordering strategy with randomization that enhances bidirectional learning capabilities.

Skinning weight prediction. Following skeleton generation, automatic rigging requires skinning weights prediction to establish joint influence on mesh vertices. Traditional geometric approaches [18, 28, 19, 6] assign weights based on vertex-joint distances—a method that proves inadequate for complex topologies. Learning-based approaches [46, 86, 54, 55] consistently integrate graph neural networks (GNN) with geometric distance cues for skinning weight prediction. However, these GNN-based methodologies face significant limitations in scalability and struggle to generalize effectively across 3D data with diverse spatial orientations. MagicArticulate [67] formulates skinning weight prediction as a functional diffusion problem [97], but suffers from slow inference and limited generalization. We instead introduce an attention-based network that strategically incorporates skeleton graph distances, enabling more robust skinning weight prediction with substantially enhanced generalization across diverse object categories. Concurrent works [100, 17] similarly leverage cross-attention between surface points and bones to learn skinning weights.

3 Automatic rigging

Our automatic rigging framework features two sequential modules. First, we deploy an auto-regressive transformer to infer a structurally valid skeleton from a raw 3D mesh (Section 3.2). Subsequently,

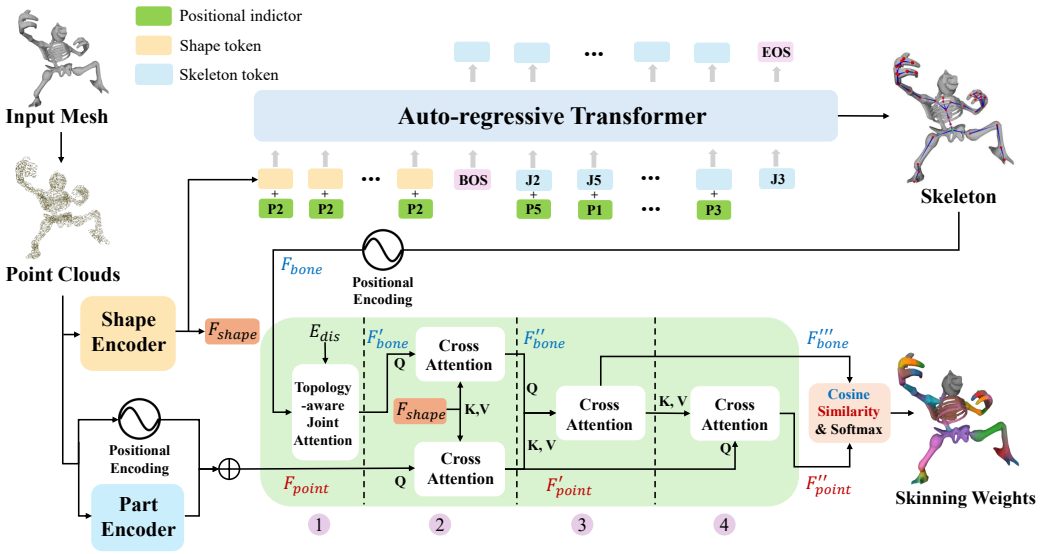


Figure 2: **Overview of our automatic rigging pipeline.** Given a 3D mesh, we first sample point clouds with normals, then generate a skeleton using an auto-regressive transformer. The point clouds and skeleton are processed through an attention-based network with four key operations: (1) bone feature enhancement via topology-aware joint attention, (2) global context integration through cross-attention with shape latents, (3) bone-point interaction via cross-attention, and (4) point feature refinement. Finally, cosine similarity and softmax normalization produce the skinning weights.

this skeleton and the original mesh are processed by an attention-based architecture to predict precise per-vertex skinning weights (Section 3.3). To facilitate large-scale learning, we introduce Articulation-XL2.0 (Section 3.1), a comprehensive dataset comprising 59.4k 3D models with high-quality rigging.

3.1 Dataset: Articulation-XL2.0

We present Articulation-XL2.0, an expanded version of Articulation-XL proposed in [67]. Our dataset incorporates multiple geometric data types from Objaverse-XL [15, 16] previously excluded, while maintaining the same data filtering process. We further improve quality by eliminating unskinned vertices and conducting manual validation, yielding over 48k high-quality rigged 3D models.

Recognizing that models in our primary dataset are predominantly in rest pose configurations, thus limiting generalization capacity to novel articulations, we have constructed a diverse-pose subset. By identifying the intersection between high-quality animation data from Diffusion4D [41] and our rigged model corpus, we extract 7.3k deformed meshes with corresponding rigging information from animation frames exhibiting maximal deviation from rest pose configurations. To counterbalance the predominance of humanoid morphologies in this subset, we supplement with 4.1k models generated using SMALR [107, 108] with parameterizations derived from 41 distinct animal scans and randomized valid poses. The resulting 11.4k diverse-pose dataset significantly enhances performance on unseen poses, as validated in our experiments. **We have released Articulation-XL2.0, a comprehensive collection of 59.4k high-quality rigged models, to facilitate future research.** Dataset statistics and examples are provided in the appendix.

3.2 Auto-regressive skeleton generation

We formulate skeleton generation as a shape-conditioned sequence modeling problem. Given an input mesh $\mathcal{M}$, we employ an auto-regressive framework (Figure 2 top) to predict a skeleton $\mathcal{S}$ consisting of 3D joint positions $\mathbf{J} \in \mathbb{R}^{j\times 3}$ and topological bone connections $\mathbf{B} \in \mathbb{N}^{b\times 2}$ defined by joint indices. Our framework consists of three key components: *joint-based skeleton tokenization*, *hierarchical sequence ordering with randomization*, and *shape-conditioned auto-regressive generation*. Together, these components enable accurate, efficient skeleton generation across varied object structures without relying on predefined templates.

Joint-based skeleton tokenization. In [67], skeletons are encoded as bone-based sequences: each of the b bones contributes 6 tokens (the 3D coordinates of its two endpoints), yielding a total sequence length of $6b$ and redundantly repeating joint positions across multiple connected bones. Inspired by

[45], we develop a joint-based tokenization strategy that represents each of the j joints by its 3D coordinates and parent index, producing a sequence of length $4j$. Since a tree-structured skeleton satisfies $j = b + 1$, this yields $4j < 6b$ whenever $j > 3$, making the joint-based representation more compact. Unlike [45], which projects joint positions into high-dimensional feature spaces via MLPs, we discretize normalized joint coordinates into a 128^3 grid and append the parent index, producing discretized token sequences that serve as input to our auto-regressive transformer. In practice, we assign the root joint a parent index of 0 and offset all other parent indices by $+1$ (subtracting 1 during detokenization).

Sequence ordering. While our joint-based tokenization provides a compact representation, the sequential ordering of tokens significantly affects skeletal coherence and model performance. For skeleton modeling with joint positions and parent indices, tokens can be sequenced using either spatial ordering (ascending z-y-x coordinates, as in [67]) or hierarchical ordering (breadth-first traversal of the skeletal tree structure). Our experiments demonstrate that spatial ordering frequently produces disconnected skeletons, as child joints generated before their parents create invalid parent references (see Section 5.5 and appendix for comparisons). We therefore adopt hierarchical ordering, applying spatial sorting only among joints at the same hierarchical level.

Additionally, inspired by [92], we enhance bidirectional learning capability through sequence randomization. We group the 4 tokens of each joint together and randomly shuffle these groups, incorporating target-aware positional indicators $\mathbf{P} = [\mathbf{p}_0, \mathbf{p}_1, ..., \mathbf{p}_{j-1}]$ to guide the generation process. Specifically, all tokens within a joint group share a positional indicator signaling which joint will be generated next. To identify the first joint group, we additionally incorporate positional indicators into shape tokens $\mathbf{T}_{shape}$ that precede skeleton tokens $\mathbf{T}_{skel}$:

$$\mathbf{T} = [\mathbf{T}_{shape}, \mathbf{T}_{skel}] + \mathbf{P} = [\mathbf{T}_{shape} + \mathbf{p}_0, \mathbf{T}^0_{skel} + \mathbf{p}_1, ..., \mathbf{T}^{j-2}_{skel} + \mathbf{p}_{j-1}, \mathbf{T}^{j-1}_{skel}]. \tag{1}$$

Shape-conditioned auto-regressive generation. With our tokenization strategy and sequence ordering established, we now describe the auto-regressive generation process. We sample 8,192 points with normals from the input mesh as shape conditioning and encode them using a pre-trained shape encoder [105]. This fixed-length shape token sequence $\mathbf{T}_{shape}$ precedes the transformer's skeleton sequence, with $<\text{bos}>$ and $<\text{eos}>$ tokens marking skeleton boundaries (omitted in Equation (1)). We adopt OPT-350M [103] as our decoder-only transformer architecture, training with cross-entropy loss for next-token prediction:

$$\mathcal{L}_{pred} = \text{CE}(\mathbf{T}, \hat{\mathbf{T}}), \tag{2}$$

where $\mathbf{T}$ and $\hat{\mathbf{T}}$ represent ground truth and predicted token sequences. During inference, generation begins with shape tokens and *sequential* positional indicators, proceeding auto-regressively until producing the $<\text{eos}>$ token, followed by detokenization to recover the complete skeleton.

3.3 Attention-based skinning weight prediction

In this section, we present an attention-based network for predicting per-vertex skinning weights that determine how the mesh deforms in response to skeleton articulation.

Network architecture. The network architecture is illustrated at the bottom of Figure 2. Our pipeline begins by sampling n points with normals from the input mesh. These points are processed through positional encoding and a part encoder from PartField [47] to obtain part-aware point embeddings $\mathbf{F}_{point} \in \mathbb{R}^{n \times d}$ that combine spatial information with part features. We incorporate part-aware features because parts and bones exhibit strong anatomical correspondence, providing valuable structural guidance for skinning weight prediction. In parallel, we construct bone-based coordinates $\in \mathbb{R}^{j \times 6}$ by concatenating each joint's parent position with its own position—for the root joint, its position is duplicated to fill both coordinate slots. These bone coordinates similarly undergo positional encoding to produce bone embeddings $\mathbf{F}_{bone} \in \mathbb{R}^{j \times d}$. Additionally, we feed the sampled points with normals into a pre-trained shape encoder [105] to extract global shape latents $\mathbf{F}_{shape} \in \mathbb{R}^{257 \times d}$.

The architecture then performs a series of attention operations [78]:(1) Bone feature enhancement. We first apply self-attention using the topology-aware joint attention on the bone embedding to obtain enhanced bone features $\mathbf{F}'_{bone}$. (2) Global context integration. Cross-attention is performed between global shape latents (as context) and both point and bone features, generating updated features $\mathbf{F}'_{point}$ and $\mathbf{F}''_{bone}$. (3) Bone-point interaction. Cross-attention uses the updated bone features as queries and

$\mathbf{F}'_{point}$ as keys/values to produce refined bone features $\mathbf{F}'''_{bone}$. (4) Point feature refinement. Final cross-attention between refined bone features $\mathbf{F}'''_{bone}$ (as context) and point features $\mathbf{F}'_{point}$ produces the final point features $\mathbf{F}''_{point}$. Finally, the network computes cosine similarity scores and applies softmax normalization to produce skinning weights:

$$\mathbf{W} = \text{softmax}\left(\alpha \frac{\mathbf{F}''_{point}\mathbf{F}'''^{\top}_{bone}}{\left\|\mathbf{F}''_{point}\right\| \left\|\mathbf{F}'''_{bone}\right\|}\right). \tag{3}$$

where α is a learnable scaling parameter. We optimize the network using cross-entropy loss during training.

Topology-aware joint attention. While the basic architecture effectively predicts weight, explicitly modeling the skeletal structure significantly enhances performance. Our ablation studies demonstrate that using bone-based coordinates $\in \mathbb{R}^{j\times 6}$ rather than joint coordinates $\in \mathbb{R}^{j\times 3}$ substantially improves performance (see Section 5.5), highlighting the importance of inter-joint relationships within the skeletal structure.

To further leverage topological structure, we propose Topology-aware Joint Attention (TAJA), which augments standard self-attention with relative positional encodings derived from skeletal graph distances. To implement TAJA, we first compute a graph distance matrix $\mathbf{D} \in \mathbb{R}^{j\times j}$ from the skeletal structure, then transform these distances into continuous embeddings through quantization and projection operations, yielding position embeddings $\mathbf{E}_{dis} \in \mathbb{R}^{j\times j\times h}$, where h is the number of attention heads. The attention mechanism is then modified as:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{E}_{dis}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}} + \lambda \mathbf{E}_{dis}\right)\mathbf{V}, \tag{4}$$

where λ is a learnable scaling parameter. This approach explicitly incorporates inter-joint topological relationships, improving the network's capacity to understand skeletal structure and generate more accurate skinning weights.

4 Video-guided 3D animation

With the generated skeleton and skinning weights, we transform static meshes into animation-ready assets. This section presents our optimization-based approach for automatically animating rigged 3D models with video guidance.

Animation pipeline. Our animation process begins by rendering the rigged mesh as the initial frame $\mathbf{I}_0$. Using this as a conditioning image, we leverage recent text-to-video generation models [34, 1] that can maintain object identity while creating plausible motion sequences. With a text prompt describing the desired animation, these models generate a video sequence $V = \{\mathbf{I}_0, \mathbf{I}_1, ..., \mathbf{I}_{n-1}\}$ comprising n frames. Given this reference video sequence V, we jointly optimize per-frame joint rotations and global root motion of the 3D mesh to align the resulting animation with the generated video sequence.

Differentiable optimization framework. For each frame $i \in \{1, 2, ..., n-1\}$ excluding the first frame, we optimize both root motion parameters $(\mathbf{Q}^i_{root}, \mathbf{T}^i_{root})$ and joint-specific rotations $Q^i_{joint} = \{\mathbf{Q}^i_0, \mathbf{Q}^i_1, ..., \mathbf{Q}^i_{j-1}\}$, where $\mathbf{Q} \in \mathbb{R}^4$ represents rotation as a unit quaternion and $\mathbf{T} \in \mathbb{R}^3$ denotes translation. For the first frame (rest pose), we initialize all transformations with identity quaternions and zero translations, which remain fixed during optimization. All subsequent frames are similarly initialized before optimization begins. Our optimization process incorporates rendering losses, tracking losses, and regularization terms:

$$\mathcal{L} = \underbrace{(\mathcal{L}_{rgb} + \mathcal{L}_{mask} + \mathcal{L}_{flow} + \mathcal{L}_{depth})}_{\text{rendering losses}} + \underbrace{(\mathcal{L}_{joint\_track} + \mathcal{L}_{vertex\_track})}_{\text{tracking losses}} + \mathcal{L}_{reg}. \tag{5}$$

For rendering losses, we utilize differentiable rendering via Pytorch3D [57] to generate predicted frames $\mathbf{I}'_i$ and compute RGB, mask, optical flow, and depth discrepancies between these predictions and the corresponding reference video frames. The optical flow and depth for video frames are extracted using off-the-shelf methods [10, 53]. The tracking losses incorporate a 2D joint tracking term and a 2D vertex tracking term that leverage Cotracker3 [35] to trace selected points throughout the video sequence. We project our optimized 3D joints and deformed mesh vertices into 2D space and minimize their distance to the corresponding tracked 2D keypoints. To address occlusion challenges,

Table 1: **Quantitative comparison of skeleton generation.** We evaluate each method on three benchmarks using CD-J2J, CD-J2B, and CD-B2B—all reported in units of 10^{-2}. Lower values indicate better alignment. * denotes models trained on Articulation-XL2.0 including the diverse-pose subset; unmarked models were trained without it. Bold and underlined numbers denote the best and second-best results, respectively.

Method	Articulation-XL2.0			ModelsResource			Diverse-pose		
	J2J ↓	J2B ↓	B2B ↓	J2J ↓	J2B ↓	B2B ↓	J2J ↓	J2B ↓	B2B ↓
Pinocchio	8.324	6.612	5.485	6.852	4.824	4.089	7.967	6.411	5.149
RigNet	7.618	6.076	5.279	7.223	5.987	4.329	7.751	6.392	5.713
MagicArti.	3.172	2.419	2.050	4.129	3.149	2.705	4.525	3.602	3.084
UniRig	3.305	2.611	2.180	3.964	3.021	2.570	<u>3.252</u>	<u>2.569</u>	2.077
Ours	<u>3.062</u>	<u>2.342</u>	<u>1.963</u>	<u>3.843</u>	<u>2.876</u>	<u>2.465</u>	3.276	2.597	<u>2.074</u>
Ours*	**3.047**	**2.337**	**1.952**	**3.785**	**2.847**	**2.430**	**2.483**	**1.922**	**1.600**

we implement visibility detection mechanisms for both joints and vertices. For joints, we define visibility based on ray-mesh intersection: *a joint is considered visible if the ray projected from the camera to the joint intersects the mesh surface exactly once*. We employ the ray_mesh_intersect function from libigl [29] to compute these joint visibility masks. For vertex visibility, we leverage the rasterization output from Pytorch3D to determine visible surface points. These visibility masks, derived from the first frame, ensure that our tracking losses are applied consistently throughout the sequence based on initial visibility, preventing optimization artifacts from elements that are occluded in the reference pose. We further incorporate regularization terms that enforce frame-to-frame motion smoothness. Complete mathematical formulations of all loss components are provided in the appendix.

5 Experiments

5.1 Experimental setup

Datasets. We train our models on the Articulation-XL2.0 dataset introduced in Section 3.1, which contains over 48k high-quality samples from Objaverse-XL [15, 16] as the main set and 11.4k samples from the diverse-pose subset. For model training, we utilize over 46k samples from the main subset and 10.9k from the diverse-pose subset. For evaluation, we employ three distinct test sets: Articulation-XL2.0-test (2k data from the main set), ModelsResource-test [76, 86] (270 upright, front-facing models with no overlap with Articulation-XL2.0, enabling assessment of cross-dataset generalization), and a 500-mesh portion of the diverse-pose subset specifically selected to evaluate model performance under varied poses.

Implementation details. To enhance robustness and generalization capabilities, we apply geometric data augmentations (scaling, shifting, rotation transformations) and pose augmentation—articulating the training samples with their ground truth skeleton and skinning weights to simulate diverse poses. Further implementation details are provided in the appendix.

5.2 Skeleton generation results

Baselines and metrics. We include four comparison methods as baselines: Pinocchio [6], which fits predefined skeleton templates to input meshes. RigNet [86], a learning-based model that employs graph convolutions to infer joint locations. MagicArticulate [67], an auto-regressive framework for skeleton generation, and the concurrent method UniRig [100], which similarly uses an auto-regressive transformer approach. All methods are evaluated on Articulation-XL2.0 and ModelsResource test sets, as well as our diverse-pose subset. We evaluate skeleton generation quality using three Chamfer Distance–based metrics from [85, 86]: CD-J2J (joint-to-joint), CD-J2B (joint-to-bone) and CD-B2B (bone-to-bone). These metrics measure the spatial alignment between generated and ground truth skeletons, where lower values indicate better performance.

Comparison results. Qualitative results are shown in Figure 3 for all three benchmarks. RigNet consistently produces invalid skeletons—its graph-convolutional model fails to converge well when trained on our large-scale dataset with highly varied orientations. UniRig presents missing and misaligned skeletons, such as missing bones on the turtle limbs and squirrel tail and misaligned skeletons on human hands, as marked in yellow circles. MagicArticulate matches reference skeletons

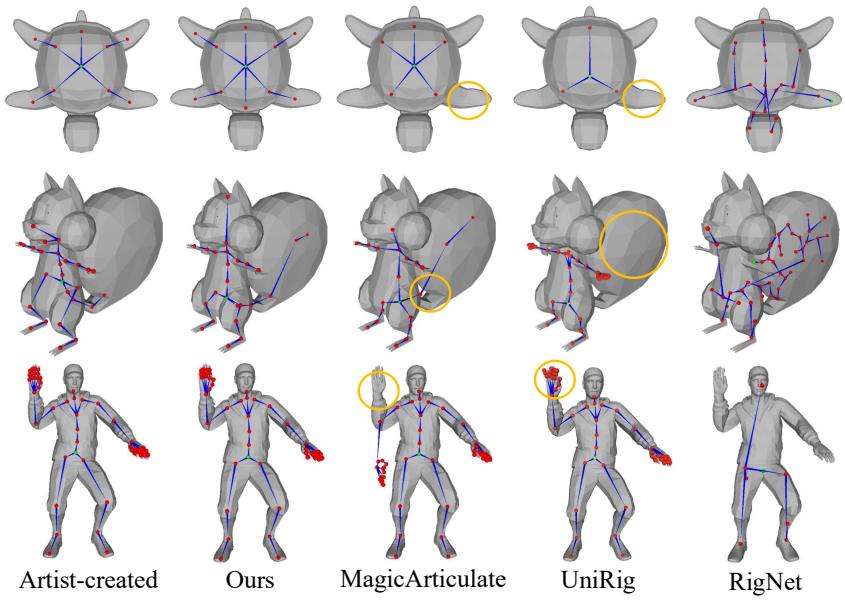


Figure 3: **Qualitative skeleton generation results**. The data is from Articulation-XL2.0, ModelsResource, and the diverse-pose subset from top to bottom.

closely on Articulation-XL2.0 and ModelsResource, but exhibits errors in fine details (e.g., missing bones in turtle limbs, incorrect squirrel tail–body junctions) and degrades on the diverse-pose subset, since it was trained only on predominantly rest-pose data without pose augmentation. In contrast, our method yields accurate, structurally correct skeletons across three benchmarks. Importantly, our generated skeletons can even correct omissions in artist-created skeletons, such as a missing turtle head–body connection. Table 1 reports quantitative metrics, where we consistently outperform all baselines on every dataset and metric. Notably, incorporating the diverse-pose subset during training leads to marked improvements on the diverse-pose benchmark.

5.3 Skinning weight prediction results

Baselines and metrics. We compare our method for skinning weight prediction against three baselines: Geodesic Voxel Binding (GVB) [18], a geometry-based technique available in Autodesk Maya [27], RigNet [86], and MagicArticulate [67]. We also evaluate these three methods on Articulation-XL2.0 and ModelsResource test sets, as well as our diverse-pose subset. Skinning weight quality is evaluated using three metrics: precision, recall, and L1-norm error. Precision is the fraction of predicted weights $> 1\mathrm{e}{-4}$ that are correct, and recall is the fraction of true weights $> 1\mathrm{e}{-4}$ we recover. The L1-norm error reports the average absolute deviation between predicted and ground truth weights over all vertices. Deformation error results are provided in the appendix.

Comparison results. Figure 4 visualizes each method's predicted skinning weights alongside their L1 error maps. Our method produces more accurate weight distributions with substantially lower errors across all benchmarks. RigNet exhibits large errors on all examples, while MagicArticulate's functional diffusion performs well on Articulation-XL2.0 and the diverse-pose subset but degrades on ModelsResource, revealing limited cross-dataset generalization. Quantitative results in Table 2 confirm these observations, with our method outperforming all baselines on every metric and dataset. Moreover, our approach runs faster—achieving per-example inference speeds that are $1.75\times$, $45\times$, and $59\times$ those of RigNet, MagicArticulate, and GVB, respectively (see appendix for details).

5.4 3D animation results

Baselines. We compare our animation results with L4GM [59] for video-to-4D generation and MotionDreamer [77] for 3D mesh animation. To ensure a fair evaluation, L4GM is given the same input videos and its multi-view synthesis for the first frame is replaced with ground-truth renderings of the input 3D model. MotionDreamer receives the input 3D model along with the same text prompts used for video generation. In Figure 5, some of its outputs appear untextured because its watertight mesh conversion breaks the UV mappings.

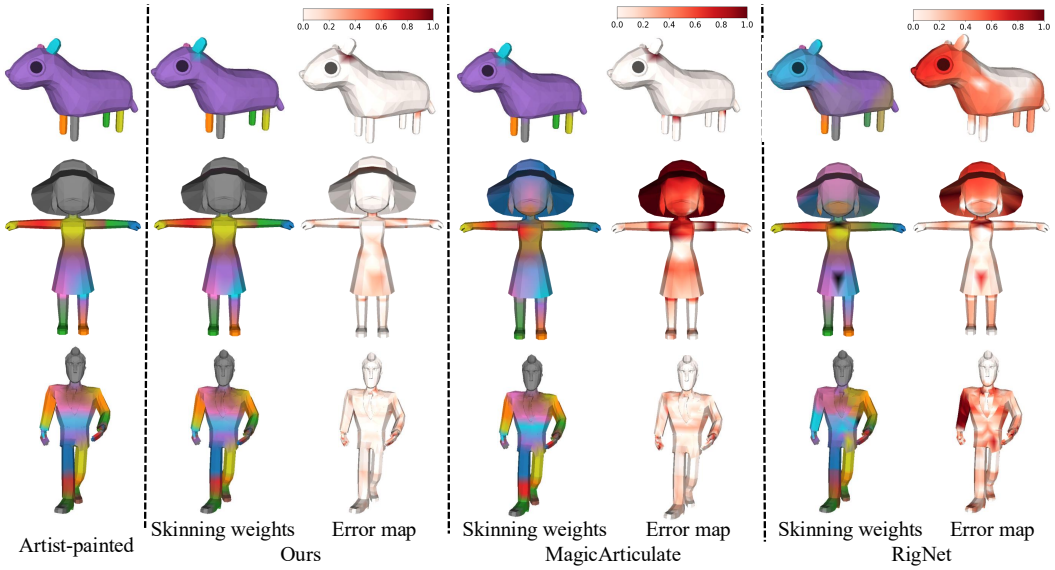


Figure 4: **Qualitative skinning weight prediction results**. The data is from Articulation-XL2.0, ModelsResource, and the diverse-pose subset from top to bottom. Each example shows the predicted weight visualization alongside its L1 error map. Additional results are provided in the appendix.

Table 2: **Quantitative comparison of skinning weight prediction.** We evaluate our approach against GVB, RigNet, and MagicArticulate. For Precision (Prec.) and Recall (Rec.), higher values indicate better accuracy and coverage. For average L1-norm error (L1), lower is better. Here, * denotes models trained on Articulation-XL2.0 with the diverse-pose subset.

Method	Articulation-XL2.0			ModelsResource			Diverse-pose		
	Prec. ↑	Rec. ↑	L1 ↓	Prec. ↑	Rec. ↑	L1 ↓	Prec. ↑	Rec. ↑	L1 ↓
GVB	72.9%	65.5%	0.745	69.3%	79.2%	0.687	75.2%	64.9%	0.786
RigNet	73.7%	66.1%	0.729	65.7%	80.2%	0.707	74.7%	65.4%	0.746
MagicArti.	74.6%	71.3%	0.451	68.1%	80.7%	0.642	74.9%	68.4%	0.479
Ours	87.6%	**74.0%**	0.335	79.7%	**81.6%**	0.443	83.6%	72.2%	0.405
Ours*	**87.9%**	73.8%	**0.333**	**79.8%**	81.5%	**0.442**	**86.4%**	**72.8%**	**0.353**

Comparison results. As shown in Figure 5, we present our generated skeletons and the corresponding video-guided animations. The shapes with skeletons represent the rest poses. Although L4GM's reference views are well aligned with the source video, it repeatedly produces geometric distortions (red highlights), even when provided with ground truth multi-view renderings. MotionDreamer's animations are subtle and can introduce unintended deformations in rigid parts (e.g., the humanoid torso). By contrast, our approach produces accurate, artifact-free animations using fully generated rigging.

5.5 Ablation studies

In this section, we present ablation studies on both skeleton generation and skinning weight prediction. All models are trained on Articulation-XL2.0 without the diverse-pose subset.

Ablation studies on skeleton generation. We ablate four components—pose augmentation, order randomization, tokenization scheme, and skeleton ordering strategy—to measure their effects on skeleton generation (see Table 3). Removing pose augmentation degrades performance across all benchmarks, especially on the diverse-pose test. Disabling order randomization similarly reduces accuracy. Bone-based tokenization matches our method's quality but requires 12 extra training hours and is 1.6× slower at inference. Finally, replacing hierarchical ordering with spatial ordering preserves CD-J2J and CD-J2B but markedly increases CD-B2B error and often produces disconnected skeletons; see the appendix for visualization comparisons.

Ablation studies on skinning weight prediction. We evaluate four key components of our skinning weight prediction framework (see Table 4). First, replacing bone embeddings with joint embeddings

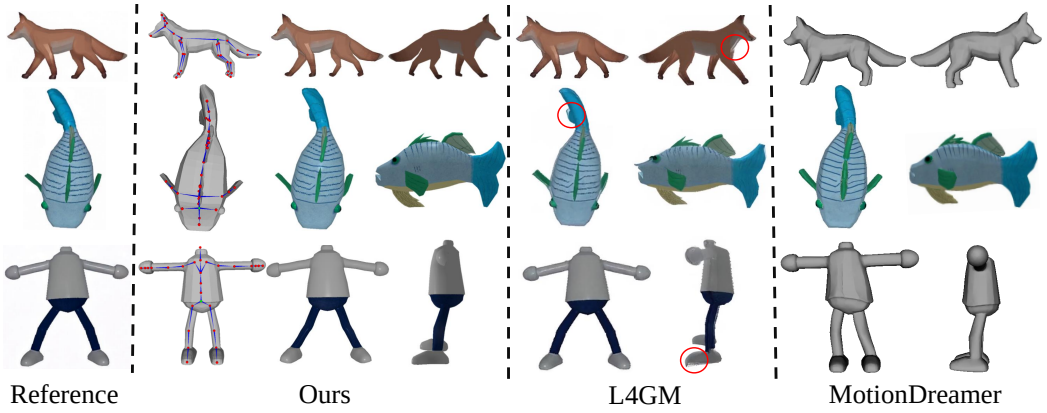


Figure 5: **Comparison of animation results.** We present our generated skeletons and corresponding video-guided animations. **The shapes with skeletons represent rest poses.** While L4GM [59] aligns its reference views closely with the input video, it consistently exhibits distortions (highlighted in red). MotionDreamer's [77] animations are subtle and can introduce unintended deformations in rigid parts (e.g., the humanoid torso). In contrast, our method delivers accurate, artifact-free animations using fully generated rigging. Videos are included in the project page.

Table 3: **Ablation studies on skeleton generation.**

Method	Articulation-XL2.0			ModelsResource			Diverse-pose		
	J2J ↓	J2B ↓	B2B ↓	J2J ↓	J2B ↓	B2B ↓	J2J ↓	J2B ↓	B2B ↓
w/o pose aug.	3.131	2.451	2.223	3.994	3.141	2.843	4.886	4.029	3.629
w/o random	3.166	2.431	2.057	3.902	3.006	2.695	3.356	2.631	2.201
Bone token	3.014	2.309	1.939	3.865	2.940	2.524	3.269	**2.518**	2.087
Spatial order	**2.982**	**2.298**	2.068	3.868	2.961	2.641	**3.210**	2.570	2.295
Ours	3.033	2.300	**1.923**	**3.841**	**2.881**	**2.475**	3.212	2.542	**2.027**

increases the average L1-norm error by 4.0% across all three benchmarks, demonstrating the importance of explicitly modeling bone information. Second, replacing Topology-aware Joint Attention (TAJA) with standard self-attention leads to performance degradation across all benchmarks, highlighting the value of modeling topological relationships between joints. Third, removing part-aware features results in consistent performance drops, confirming their contribution to accurate weight prediction. Finally, eliminating pose augmentation during training increases the L1-norm error on the diverse-pose subset by 9.6%, demonstrating that pose variation is essential for generalization to novel poses. These findings confirm that each component is crucial to our model's overall accuracy.

Table 4: **Ablation studies on skinning weight prediction.**

Method	Articulation-XL2.0			ModelsResource			Diverse-pose		
	Prec. ↑	Rec. ↑	L1 ↓	Prec. ↑	Rec. ↑	L1 ↓	Prec. ↑	Rec. ↑	L1 ↓
Joint embed	87.1%	73.8%	0.346	79.2%	80.8%	0.458	82.8%	72.2%	0.427
w/o TAJA	86.6%	**74.2%**	0.348	79.2%	81.4%	0.450	82.8%	**72.5%**	0.414
w/o part feat.	87.4%	73.8%	0.338	79.1%	81.0%	0.451	83.2%	72.1%	0.414
w/o pose aug.	**88.0%**	73.3%	0.337	79.3%	80.7%	0.449	82.8%	70.1%	0.444
Ours	87.6%	74.0%	**0.335**	**79.7%**	**81.6%**	**0.443**	**83.6%**	72.2%	**0.405**

6 Conclusion

In this work, we introduce Puppeteer, a unified rigging-and-animation pipeline built on a dataset with 59.4k high-quality rigged models. Puppeteer first generates skeletons with an autoregressive transformer that uses joint-based tokenization and hierarchical ordering with randomization to capture skeletal structures. An attention-based network with topology-aware features then predicts skinning weights, followed by an efficient optimization module that produces stable, high-quality animations at low computational cost. Across multiple benchmarks, Puppeteer outperforms state-of-the-art methods in skeleton fidelity, skinning accuracy, and animation smoothness.

Acknowledgements

This research is supported by the MoE AcRF Tier 2 grant (MOE-T2EP20223-0001) and the MoE AcRF Tier 1 grant (RG14/22).

References

[1] K. AI. Kling ai, 2025. URL https://klingai.com/.

[2] T. AI. Tripo 3d, 2023. URL https://www.tripo3d.ai/.

[3] O. K.-C. Au, C.-L. Tai, H.-K. Chu, D. Cohen-Or, and T.-Y. Lee. Skeleton extraction by mesh contraction. *ACM transactions on graphics (TOG)*, 27(3):1–10, 2008.

[4] S. Bahmani, X. Liu, W. Yifan, I. Skorokhodov, V. Rong, Z. Liu, X. Liu, J. J. Park, S. Tulyakov, G. Wetzstein, A. Tagliasacchi, and D. B. Lindell. Tc4d: Trajectory-conditioned text-to-4d generation. *arXiv*, 2024.

[5] S. Bahmani, I. Skorokhodov, V. Rong, G. Wetzstein, L. Guibas, P. Wonka, S. Tulyakov, J. J. Park, A. Tagliasacchi, and D. B. Lindell. 4d-fy: Text-to-4d generation using hybrid score distillation sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7996–8006, 2024.

[6] I. Baran and J. Popović. Automatic rigging and animation of 3d characters. *ACM Transactions on graphics (TOG)*, 26(3):72–es, 2007.

[7] J. Cao, A. Tagliasacchi, M. Olson, H. Zhang, and Z. Su. Point cloud skeletons via laplacian based contraction. In *2010 Shape Modeling International Conference*, pages 187–197. IEEE, 2010.

[8] C. Chen, S. Huang, X. Chen, G. Chen, X. Han, K. Zhang, and M. Gong. Ct4d: Consistent text-to-4d generation with animatable meshes. *arXiv preprint arXiv:2408.08342*, 2024.

[9] J. Chen, B. Zhang, X. Tang, and P. Wonka. V2m4: 4d mesh animation reconstruction from a single monocular video. *arXiv preprint arXiv:2503.09631*, 2025.

[10] S. Chen, H. Guo, S. Zhu, F. Zhang, Z. Huang, J. Feng, and B. Kang. Video depth anything: Consistent depth estimation for super-long videos. *arXiv:2501.12375*, 2025.

[11] Y. Chen, T. He, D. Huang, W. Ye, S. Chen, J. Tang, X. Chen, Z. Cai, L. Yang, G. Yu, et al. Meshanything: Artist-created mesh generation with autoregressive transformers. *arXiv preprint arXiv:2406.10163*, 2024.

[12] Y. Chen, Y. Wang, Y. Luo, Z. Wang, Z. Chen, J. Zhu, C. Zhang, and G. Lin. Meshanything v2: Artist-created mesh generation with adjacent mesh tokenization. *arXiv preprint arXiv:2408.02555*, 2024.

[13] Z. Chu, F. Xiong, M. Liu, J. Zhang, M. Shao, Z. Sun, D. Wang, and M. Xu. Humanrig: Learning automatic rigging for humanoid character in a large scale dataset, 2024. URL https://arxiv.org/abs/2412.02317.

[14] E. De Aguiar, C. Theobalt, S. Thrun, and H.-P. Seidel. Automatic conversion of mesh animations into skeleton-based animations. In *Computer Graphics Forum*, volume 27, pages 389–397. Wiley Online Library, 2008.

[15] M. Deitke, D. Schwenk, J. Salvador, L. Weihs, O. Michel, E. VanderBilt, L. Schmidt, K. Ehsani, A. Kembhavi, and A. Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153, 2023.

[16] M. Deitke, R. Liu, M. Wallingford, H. Ngo, O. Michel, A. Kusupati, A. Fan, C. Laforte, V. Voleti, S. Y. Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems*, 36, 2024.

[17] Y. Deng, Y. Zhang, C. Geng, S. Wu, and J. Wu. Anymate: A dataset and baselines for learning 3d object rigging. In *SIGGRAPH*, 2025.

[18] O. Dionne and M. de Lasa. Geodesic voxel binding for production character meshes. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 173–180, 2013.

[19] A. Dodik, V. Sitzmann, J. Solomon, and O. Stein. Robust biharmonic skinning using geometric fields. *arXiv preprint arXiv:2406.00238*, 2024.

[20] Z. Fu, J. Wei, W. Shen, C. Song, X. Yang, F. Liu, X. Yang, and G. Lin. Sync4d: Video guided controllable dynamics for physics-based 4d generation. *arXiv preprint arXiv:2405.16849*, 2024.

[21] Q. Gao, Q. Xu, Z. Cao, B. Mildenhall, W. Ma, L. Chen, D. Tang, and U. Neumann. Gaussianflow: Splatting gaussian dynamics for 4d content creation. 2024.

[22] I. Gat, S. Raab, G. Tevet, Y. Reshef, A. H. Bermano, and D. Cohen-Or. Anytop: Character animation diffusion with any topology. *arXiv preprint arXiv:2502.17327*, 2025.

[23] C. Guo, S. Zou, X. Zuo, S. Wang, W. Ji, X. Li, and L. Cheng. Generating diverse and natural 3d human motions from text. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5152–5161, 2022.

[24] Z. Guo, J. Xiang, K. Ma, W. Zhou, H. Li, and R. Zhang. Make-it-animatable: An efficient framework for authoring animation-ready 3d characters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[25] H. Huang, S. Wu, D. Cohen-Or, M. Gong, H. Zhang, G. Li, and B. Chen. L1-medial skeleton of point cloud. *ACM Trans. Graph.*, 32(4):65–1, 2013.

[26] A. Inc. Mixamo. URL https://www.mixamo.com/.

[27] A. Inc. Autodesk maya, 2024. URL https://www.autodesk.com/products/maya/overview. Version 2024.

[28] A. Jacobson, I. Baran, J. Popovic, and O. Sorkine. Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph.*, 30(4):78, 2011.

[29] A. Jacobson, D. Panozzo, et al. libigl: A simple C++ geometry processing library, 2018. https://libigl.github.io/.

[30] D. L. James and C. D. Twigg. Skinning mesh animations. *ACM Transactions on Graphics (TOG)*, 24(3):399–407, 2005.

[31] B. Jiang, X. Chen, W. Liu, J. Yu, G. Yu, and T. Chen. Motiongpt: Human motion as a foreign language. *Advances in Neural Information Processing Systems*, 36:20067–20079, 2023.

[32] Y. Jiang, L. Zhang, J. Gao, W. Hu, and Y. Yao. Consistent4d: Consistent 360 {\deg} dynamic object generation from monocular video. *arXiv preprint arXiv:2311.02848*, 2023.

[33] Y. Jiang, C. Yu, C. Cao, F. Wang, W. Hu, and J. Gao. Animate3d: Animating any 3d model with multi-view video diffusion. *arXiv preprint arXiv:2407.11398*, 2024.

[34] JiMeng AI. Jimeng ai, 2025. URL https://jimeng.jianying.com.

[35] N. Karaev, I. Makarov, J. Wang, N. Neverova, A. Vedaldi, and C. Rupprecht. Cotracker3: Simpler and better point tracking by pseudo-labelling real videos. *arXiv preprint arXiv:2410.11831*, 2024.

[36] J. P. Lewis, M. Cordner, and N. Fong. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 811–818. 2023.

[37] P. Li, K. Aberman, R. Hanocka, L. Liu, O. Sorkine-Hornung, and B. Chen. Learning skeletal articulations with neural blend shapes. *ACM Transactions on Graphics (TOG)*, 40(4):1–15, 2021.

[38] X. Li, Q. Ma, T.-Y. Lin, Y. Chen, C. Jiang, M.-Y. Liu, and D. Xiang. Articulated kinematics distillation from video diffusion models. *arXiv preprint arXiv:2504.01204*, 2025. URL https://arxiv.org/abs/2504.01204.

[39] Z. Li, Y. Chen, and P. Liu. Dreammesh4d: Video-to-4d generation with sparse-controlled gaussian-mesh hybrid representation. *Advances in Neural Information Processing Systems*, 37: 21377–21400, 2024.

[40] Z. Li, D. Litvak, R. Li, Y. Zhang, T. Jakab, C. Rupprecht, S. Wu, A. Vedaldi, and J. Wu. Learning the 3d fauna of the web. In *CVPR*, 2024.

[41] H. Liang, Y. Yin, D. Xu, H. Liang, Z. Wang, K. N. Plataniotis, Y. Zhao, and Y. Wei. Diffusion4d: Fast spatial-temporal consistent 4d generation via video diffusion models. *arXiv preprint arXiv:2405.16645*, 2024.

[42] C. Lin, C. Li, Y. Liu, N. Chen, Y.-K. Choi, and W. Wang. Point2skeleton: Learning skeletal representations from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4277–4286, 2021.

[43] J. Lin, Z. Wang, Y. Hou, Y. Tang, and M. Jiang. Phy124: Fast physics-driven 4d content generation from a single image. *arXiv preprint arXiv:2409.07179*, 2024.

[44] H. Ling, S. W. Kim, A. Torralba, S. Fidler, and K. Kreis. Align your gaussians: Text-to-4d with dynamic 3d gaussians and composed diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8576–8588, 2024.

[45] I. Liu, Z. Xu, W. Yifan, H. Tan, Z. Xu, X. Wang, H. Su, and Z. Shi. Riganything: Template-free autoregressive rigging for diverse 3d assets. *arXiv preprint arXiv:2502.09615*, 2025.

[46] L. Liu, Y. Zheng, D. Tang, Y. Yuan, C. Fan, and K. Zhou. Neuroskinning: Automatic skin binding for production characters with deep graph networks. *ACM Transactions on Graphics (ToG)*, 38(4):1–12, 2019.

[47] M. Liu, M. A. Uy, D. Xiang, H. Su, S. Fidler, N. Sharp, and J. Gao. Partfield: Learning 3d feature fields for part segmentation and beyond. *arXiv preprint arXiv:2504.11451*, 2025.

[48] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, Oct. 2015.

[49] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Seminal graphics: pioneering efforts that shaped the field*, pages 347–353. 1998.

[50] Q. Miao, J. Quan, K. Li, and Y. Luo. Pla4d: Pixel-level alignments for text-to-4d gaussian splatting. *arXiv preprint arXiv:2405.19957*, 2024.

[51] Q. Miao, K. Li, J. Quan, Z. Min, S. Ma, Y. Xu, Y. Yang, and Y. Luo. Advances in 4d generation: A survey. *arXiv preprint arXiv:2503.14501*, 2025.

[52] M. B. S. Millán, A. Dai, and M. Nießner. Animating the uncaptured: Humanoid mesh animation with video diffusion models. *arXiv preprint arXiv:2503.15996*, 2025.

[53] H. Morimitsu, X. Zhu, R. M. Cesar-Jr., X. Ji, and X.-C. Yin. DPFlow: Adaptive optical flow estimation with a dual-pyramid framework. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[54] A. Mosella-Montoro and J. Ruiz-Hidalgo. Skinningnet: Two-stream graph convolutional neural network for skinning prediction of synthetic characters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18593–18602, 2022.

[55] X. Pan, J. Huang, J. Mai, H. Wang, H. Li, T. Su, W. Wang, and X. Jin. Heterskinnet: A heterogeneous network for skin weights prediction. *Proc. ACM Comput. Graph. Interact. Tech.*, 4(1), Apr. 2021. doi: 10.1145/3451262. URL https://doi.org/10.1145/3451262.

[56] R. Parent. *Computer animation: algorithms and techniques*. Newnes, 2012.

[57] N. Ravi, J. Reizenstein, D. Novotny, T. Gordon, W.-Y. Lo, J. Johnson, and G. Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020.

[58] J. Ren, L. Pan, J. Tang, C. Zhang, A. Cao, G. Zeng, and Z. Liu. Dreamgaussian4d: Generative 4d gaussian splatting. *arXiv preprint arXiv:2312.17142*, 2023.

[59] J. Ren, C. Xie, A. Mirzaei, K. Kreis, Z. Liu, A. Torralba, S. Fidler, S. W. Kim, H. Ling, et al. L4gm: Large 4d gaussian reconstruction model. *Advances in Neural Information Processing Systems*, 37:56828–56858, 2024.

[60] W. Shen, W. Yin, H. Wang, C. Wei, Z. Cai, L. Yang, and G. Lin. Hmr-adapter: A lightweight adapter with dual-path cross augmentation for expressive human mesh recovery. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 6093–6102, 2024.

[61] W. Shen, W. Yin, X. Yang, C. Chen, C. Song, Z. Cai, L. Yang, H. Wang, and G. Lin. Adhmr: Aligning diffusion-based human mesh recovery via direct preference optimization. *arXiv preprint arXiv:2505.10250*, 2025.

[62] Y. Siddiqui, A. Alliegro, A. Artemov, T. Tommasi, D. Sirigatti, V. Rosov, A. Dai, and M. Nießner. Meshgpt: Generating triangle meshes with decoder-only transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19615–19625, 2024.

[63] C. Song, J. Wei, R. Li, F. Liu, and G. Lin. 3d pose transfer with correspondence learning and mesh refinement. *Advances in Neural Information Processing Systems*, 34:3108–3120, 2021.

[64] C. Song, J. Wei, R. Li, F. Liu, and G. Lin. Unsupervised 3d pose transfer with cross consistency and dual reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45 (8):10488–10499, 2023.

[65] C. Song, J. Wei, T. Chen, Y. Chen, C.-S. Foo, F. Liu, and G. Lin. Moda: Modeling deformable 3d objects from casual videos. *International Journal of Computer Vision*, pages 1–20, 2024.

[66] C. Song, J. Wei, C. S. Foo, G. Lin, and F. Liu. Reacto: Reconstructing articulated objects from a single video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5384–5395, 2024.

[67] C. Song, J. Zhang, X. Li, F. Yang, Y. Chen, Z. Xu, J. H. Liew, X. Guo, F. Liu, J. Feng, and G. Lin. Magicarticulate: Make your 3d models articulation-ready. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

[68] K. Sun, D. Litvak, Y. Zhang, H. Li, J. Wu, and S. Wu. Ponymation: Learning articulated 3d animal motions from unlabeled online videos. In *ECCV*, 2024.

[69] M. Sun, J. Chen, J. Dong, Y. Chen, X. Jiang, S. Mao, P. Jiang, J. Wang, B. Dai, and R. Huang. Drive: Diffusion-based rigging empowers generation of versatile and expressive characters. *arXiv preprint arXiv:2411.17423*, 2024.

[70] Q. Sun, Z. Guo, Z. Wan, J. N. Yan, S. Yin, W. Zhou, J. Liao, and H. Li. Eg4d: Explicit generation of 4d object without score distillation. *arXiv preprint arXiv:2405.18132*, 2024.

[71] A. Tagliasacchi, I. Alhashim, M. Olson, and H. Zhang. Mean curvature skeletons. In *Computer Graphics Forum*, volume 31, pages 1735–1744. Wiley Online Library, 2012.

[72] J. Tang, Z. Li, Z. Hao, X. Liu, G. Zeng, M.-Y. Liu, and Q. Zhang. Edgerunner: Auto-regressive auto-encoder for artistic mesh generation. *arXiv preprint arXiv:2409.18114*, 2024.

[73] T. H. Team. Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation, 2025.

[74] G. Tevet, B. Gordon, A. Hertz, A. H. Bermano, and D. Cohen-Or. Motionclip: Exposing human motion generation to clip space. In *European Conference on Computer Vision*, pages 358–374. Springer, 2022.

[75] G. Tevet, S. Raab, B. Gordon, Y. Shafir, D. Cohen-Or, and A. H. Bermano. Human motion diffusion model. *arXiv preprint arXiv:2209.14916*, 2022.

[76] The Models-Resource. The models-resource, 2019. URL https://www.models-resource.com/.

[77] L. Uzolas, E. Eisemann, and P. Kellnhofer. Motiondreamer: Exploring semantic video diffusion features for zero-shot 3d mesh animation. In *International Conference on 3D Vision 2025*, 2025.

[78] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[79] P.-S. Wang, Y. Liu, and X. Tong. Dual octree graph networks for learning adaptive volumetric shape representations. *ACM Transactions on Graphics (TOG)*, 41(4):1–15, 2022.

[80] X. Wang, Y. Wang, J. Ye, F. Sun, Z. Wang, L. Wang, P. Liu, K. Sun, X. Wang, W. Xie, et al. Animatabledreamer: Text-guided non-rigid 3d model generation and reconstruction with canonical score distillation. In *European Conference on Computer Vision*, pages 321–339. Springer, 2024.

[81] S. Wu, R. Li, T. Jakab, C. Rupprecht, and A. Vedaldi. MagicPony: Learning articulated 3d animals in the wild. In *CVPR*, 2023.

[82] Z. Wu, C. Yu, Y. Jiang, C. Cao, F. Wang, and X. Bai. Sc4d: Sparse-controlled video-to-4d generation and motion transfer. In *European Conference on Computer Vision*, pages 361–379. Springer, 2024.

[83] Y. Xie, C.-H. Yao, V. Voleti, H. Jiang, and V. Jampani. Sv4d: Dynamic 3d content generation with multi-frame and multi-view consistency. *arXiv preprint arXiv:2407.17470*, 2024.

[84] J. L. Z. W. D. Xu and S. J. Y. G. M. Jiang. Phys4dgen: Physics-compliant 4d generation with multi-material composition perception.

[85] Z. Xu, Y. Zhou, E. Kalogerakis, and K. Singh. Predicting animation skeletons for 3d articulated models via volumetric nets. In *2019 international conference on 3D vision (3DV)*, pages 298–307. IEEE, 2019.

[86] Z. Xu, Y. Zhou, E. Kalogerakis, C. Landreth, and K. Singh. Rignet: Neural rigging for articulated characters. *arXiv preprint arXiv:2005.00559*, 2020.

[87] Z. Xu, Y. Zhou, L. Yi, and E. Kalogerakis. Morig: Motion-aware rigging of character meshes from point clouds. In *SIGGRAPH Asia 2022 conference papers*, pages 1–9, 2022.

[88] G. Yang, M. Vo, N. Neverova, D. Ramanan, A. Vedaldi, and H. Joo. Banmo: Building animatable 3d neural models from many casual videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2863–2873, 2022.

[89] C.-H. Yao, Y. Xie, V. Voleti, H. Jiang, and V. Jampani. Sv4d 2.0: Enhancing spatio-temporal consistency in multi-view video diffusion for high-quality 4d generation. *arXiv preprint arXiv:2503.16396*, 2025.

[90] Y. Yao, Z. Deng, and J. Hou. Riggs: Rigging of 3d gaussians for modeling articulated objects in videos. In *CVPR*, 2025.

[91] Y. Yin, D. Xu, Z. Wang, Y. Zhao, and Y. Wei. 4dgen: Grounded 4d content generation with spatial-temporal consistency. *arXiv preprint arXiv:2312.17225*, 2023.

[92] Q. Yu, J. He, X. Deng, X. Shen, and L.-C. Chen. Randomized autoregressive visual generation. *arXiv preprint arXiv:2411.00776*, 2024.

[93] Y.-J. Yuan, L. Kobbelt, J. Liu, Y. Zhang, P. Wan, Y.-K. Lai, and L. Gao. 4dynamic: Text-to-4d generation with hybrid priors. *arXiv preprint arXiv:2407.12684*, 2024.

[94] K. Yun, S. Hong, C. Kim, and J. Noh. Anymole: Any character motion in-betweening leveraging video diffusion models. *arXiv preprint arXiv:2503.08417*, 2025.

[95] B. Zeng, L. Yang, S. Li, J. Liu, Z. Zhang, J. Tian, K. Zhu, Y. Guo, F.-Y. Wang, M. Xu, et al. Trans4d: Realistic geometry-aware transition for compositional text-to-4d synthesis. *arXiv preprint arXiv:2410.07155*, 2024.

[96] Y. Zeng, Y. Jiang, S. Zhu, Y. Lu, Y. Lin, H. Zhu, W. Hu, X. Cao, and Y. Yao. Stag4d: Spatial-temporal anchored generative 4d gaussians. In *European Conference on Computer Vision*, pages 163–179. Springer, 2024.

[97] B. Zhang and P. Wonka. Functional diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4723–4732, 2024.

[98] H. Zhang, D. Chang, F. Li, M. Soleymani, and N. Ahuja. Magicpose4d: Crafting articulated models with appearance and motion control. *arXiv preprint arXiv:2405.14017*, 2024.

[99] H. Zhang, X. Chen, Y. Wang, X. Liu, Y. Wang, and Y. Qiao. 4diffusion: Multi-view video diffusion model for 4d generation. *Advances in Neural Information Processing Systems*, 37: 15272–15295, 2024.

[100] J.-P. Zhang, C.-F. Pu, M.-H. Guo, Y.-P. Cao, and S.-M. Hu. One model to rig them all: Diverse skeleton rigging with unirig. *arXiv preprint arXiv:2504.12451*, 2025.

[101] M. Zhang, Z. Cai, L. Pan, F. Hong, X. Guo, L. Yang, and Z. Liu. Motiondiffuse: Text-driven human motion generation with diffusion model. *IEEE transactions on pattern analysis and machine intelligence*, 46(6):4115–4128, 2024.

[102] M. Zhang, D. Jin, C. Gu, F. Hong, Z. Cai, J. Huang, C. Zhang, X. Guo, L. Yang, Y. He, et al. Large motion model for unified multi-modal motion generation. In *European Conference on Computer Vision*, pages 397–421. Springer, 2024.

[103] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

[104] Y. Zhao, Z. Yan, E. Xie, L. Hong, Z. Li, and G. H. Lee. Animate124: Animating one image to 4d dynamic scene. *arXiv preprint arXiv:2311.14603*, 2023.

[105] Z. Zhao, W. Liu, X. Chen, X. Zeng, R. Wang, P. Cheng, B. Fu, T. Chen, G. Yu, and S. Gao. Michelangelo: Conditional 3d shape generation based on shape-image-text aligned latent representation. *Advances in Neural Information Processing Systems*, 36, 2024.

[106] H. Zhu, T. He, X. Yu, J. Guo, Z. Chen, and J. Bian. Ar4d: Autoregressive 4d generation from monocular videos. *arXiv preprint arXiv:2501.01722*, 2025.

[107] S. Zuffi, A. Kanazawa, D. Jacobs, and M. J. Black. 3D menagerie: Modeling the 3D shape and pose of animals. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[108] S. Zuffi, A. Kanazawa, and M. J. Black. Lions and tigers and bears: Capturing non-rigid, 3D, articulated shape from images. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2018.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction clearly state our contributions: (1) expanding the Articulation-XL dataset to 59.4k rigged models, (2) introducing a novel auto-regressive skeleton generation approach, (3) presenting an attention-based architecture for skinning weight prediction, and (4) proposing a differentiable optimization-based animation method.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We discuss limitations in the appendix.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [NA]

Justification: We do not include theoretical results in this paper.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We provide the experimental dataset in Section 3.1, Section 5 and details to reproduce experimental results in Section 5 and the appendix.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: We will open source the data and codes upon acceptance.

Guidelines:

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

6. **Experimental setting/details**

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

Answer: [Yes]

Justification: Section 5 describes our experimental setup, including dataset splits and data augmentation strategies. Additional hyperparameters, optimization details, and hardware configuration (8 A100 GPUs) are provided in the appendix.

Guidelines:

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

7. **Experiment statistical significance**

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

Answer: [No]

Justification: We did not report error bar due to the prohibitive computation cost.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: In the appendix, we specify that our skeleton generation model was trained on 8 NVIDIA A100 GPUs for approximately 3 days and 20 hours, the skinning weight prediction model required approximately 1 day and 6 hours on the same hardware, and the animation optimization process completes in approximately 20 minutes on a single A100 GPU.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: No violation of Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: We discuss broader impacts in the appendix.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

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

11. **Safeguards**

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

    Answer: [Yes]

    Justification: We discuss this in the appendix.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We cite the datasets properly.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [NA]

    Justification: [NA]

    Guidelines:

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

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

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

    Answer: [NA]

    Justification: Our research does not involve crowdsourcing or human subjects.

    Guidelines:

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

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

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

    Answer: [NA]

    Justification: Our research does not involve human subjects or require IRB approval.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [NA]

    Justification: We only use LLM for writing.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.

Appendix

In this appendix, we provide additional details and experimental results for the main paper, including:

- Further details of Puppeteer (Section A) and Articulation-XL2.0 (Section B);
- Additional related works (Section C);
- Additional experimental results on skeleton generation, skinning weight prediction, and animation (Section D);
- A discussion of the limitations of our work and future works (Section E), and broader impact considerations (Section F).

A More details of Puppeteer

A.1 Implementation details

Skeleton generation. Our skeleton generation begins by encoding each mesh with a pre-trained shape encoder [105]. We first compute its signed distance function using [79], reconstruct a coarse mesh using Marching Cubes [49], and then sample 8,192 surface points (with normals). These points are finally encoded into a fixed sequence of 257 shape tokens. We normalize input points to $[-0.5, 0.5]$ and apply the same scale and translation to joint positions for alignment. Joint coordinates are then discretized into a 128^3 grids, and parent indices are appended—yielding a token sequence of length $4j$.

During training, we apply pose augmentation with a probability of 0.5. When pose augmentation is applied, each joint has a 0.3 probability of rotation, with rotation angles constrained to the range of $[-60°, 60°]$. Sequence ordering randomization is annealed following [92], with a permutation probability r that starts at 1 and falls to 0 (reverting to hierarchical order) over training:

$$r = \begin{cases} 1.0, & epoch \in [0, E/2], \\ 1 - \dfrac{epoch - E/2}{E/4}, & epoch \in [E/2, 3E/4], \\ 0.0, & epoch \in [3E/4, E], \end{cases} \tag{6}$$

where $epoch$ is the current epoch and E the total number of epochs. Besides, we apply target-aware positional indicators that mark which joint group the model should generate next. Both shape and skeleton token sequence (except for the final joint group) are augmented with an indicator denoting their next group; the indicator on the shape tokens specifically identifies the initial joint group to generate. The auto-regressive transformer is trained on 8 NVIDIA A100 GPUs (batch size 64 per GPU, effective batch 512) for approximately 3 days and 20 hours.

Skinning weight prediction. During training, we condition the attention-based network for skinning weight prediction on the ground truth skeleton and supervise it with the corresponding skinning weights. We sample 8,192 surface points (with normals) from each mesh—matching our skeleton pipeline—and assign each point the weights of its nearest vertex. During inference, these predicted weights are transferred back to mesh vertices through nearest-neighbor mapping.

Training is performed on Articulation-XL2.0 with 8 NVIDIA A100 GPUs for roughly 1 day and 6 hours, with a batch size of 16 per GPU. A valid-joint mask—supporting up to 70 joints—adapts the network to the varying skeleton sizes encountered during both training and evaluation.

Video-guided 3D animation. Here, we describe our video-guided 3D animation process in detail. For rendering supervision, we employ four distinct loss functions. Given a generated video $V = \{\mathbf{I}_0, \mathbf{I}_1, ..., \mathbf{I}_{n-1}\}$ and the rendered images $\mathbf{I}'_i$ using Pytorch3D [57], we calculate the following rendering losses:

$$\mathcal{L}_{rgb} = \sum_i \|\mathbf{M}_i \odot (\mathbf{I}_i - \mathbf{I}'_i)\|^2, \qquad \mathcal{L}_{mask} = \sum_i \mathrm{BCE}(\mathbf{M}_i, \mathbf{M}'_i),$$
$$\mathcal{L}_{depth} = \sum_i \|\mathbf{M}_i \odot (\mathbf{D}_i - \mathbf{D}'_i)\|^2, \quad \mathcal{L}_{flow} = \sum_i \|\mathbf{M}_i \odot (\mathbf{F}_i - \mathbf{F}'_i)\|^2, \tag{7}$$

where $\mathbf{M}_i$ represents the binary mask of foreground objects in the video frames, and $\mathbf{M}'_i$ denotes the mask of the 3D object rendered via Pytorch3D. We extract depth maps $\mathbf{D}_i$ using the method proposed in [10], while $\mathbf{D}'_i$ is obtained directly from the Pytorch3D renderer. We apply scale-shift alignment to handle the scale ambiguity between relative depth $\mathbf{D}_i$ and metric depth $\mathbf{D}'_i$. For optical flow estimation, we compute $\mathbf{F}_i$ using the approach from [53], and derive $\mathbf{F}'_i$ by projecting the 3D vertex flow onto the 2D image plane. The element-wise multiplication operator $\odot$ indicates that $\mathcal{L}_{rgb}$, $\mathcal{L}_{flow}$, and $\mathcal{L}_{depth}$ are all computed only within the foreground region defined by mask $\mathbf{M}_i$, ensuring that our optimization focuses on the target object.

The tracking losses incorporate a 2D joint tracking term and a 2D vertex tracking term that leverage Cotracker3 [35] to trace selected keypoints throughout the video sequence. We project visible joints and vertices of the 3D static object onto image planes to establish keypoints for the first frame. These keypoints are then tracked through the entire video sequence using Cotracker3 to obtain $\mathbf{p}_i$, which represents the tracked positions at each frame i. Simultaneously, $\mathbf{p}'_i$ is derived by projecting the deformed joints and mesh vertices onto image planes. The tracking losses are formulated as:

$$\mathcal{L}_{joint\_track} = \sum_i \left\|\mathbf{M}_j \odot (\mathbf{p}_{i,joint} - \mathbf{p}'_{i,joint})\right\|^2,$$
$$\mathcal{L}_{point\_track} = \sum_i \left\|\mathbf{M}_v \odot (\mathbf{p}_{i,vertex} - \mathbf{p}'_{i,vertex})\right\|^2, \tag{8}$$

where $\mathbf{M}_j$ and $\mathbf{M}_v$ denote the visibility masks for joints and vertices in the first frame, respectively. These masks ensure that only visible keypoints contribute to the optimization process. Additionally, we also incorporate regularization terms that prevent temporal jittering by penalizing large transformation changes between consecutive frames. To balance these losses, we weight each term to ensure comparable magnitudes. In practice, regularization losses are down-weighted by 3–4 orders of magnitude relative to rendering and tracking losses to prevent over-smoothing.

Our animation optimization takes approximately 20 minutes for objects with up to 10K vertices on a single NVIDIA A100 GPU, processing 5-second videos (approximately 50 frames at 10 FPS) generated by Kling AI [1] or JiMeng AI [34]. Runtime scales with both mesh complexity and frame count: (1) Mesh complexity: Models with more vertices will require additional Pytorch3D rendering time. For example, the bat case ($\sim$70K vertices) in our project page requires 90 minutes, while the turtle case ($\sim$15K vertices) takes 35 minutes. (2) Frame count: For a typical case taking 20 minutes at 50 frames (10 FPS, 5 seconds), increasing to 20 FPS (100 frames) extends optimization time to 41 minutes, while reducing to 4 FPS (20 frames) decreases it to 8 minutes, demonstrating approximately linear scaling with frame count.

After optimization, our animation process follows standard skeletal animation principles [56]: (1) Forward Kinematics (FK): We compute global joint transformations from optimized local transformations using hierarchical forward kinematics, traversing the skeleton from root to leaves; (2) Linear Blend Skinning (LBS) [36]: Deform mesh vertices using weighted combinations of joint transformations, where each vertex is influenced by multiple joints according to skinning weights. This produces the final mesh animation sequence.

A.2 Experimental details

For baseline comparisons, we use the publicly available implementations of UniRig [100], RigNet [86], and Pinocchio [6] from their respective GitHub repositories. The Geodesic Voxel Binding (GVB) [18] comparison utilizes the implementation in Autodesk Maya [27]. RigNet and MagicArticulate [67] are trained on Articulation-XL2.0 using the original data preprocessing pipelines and training schedules specified by their authors.

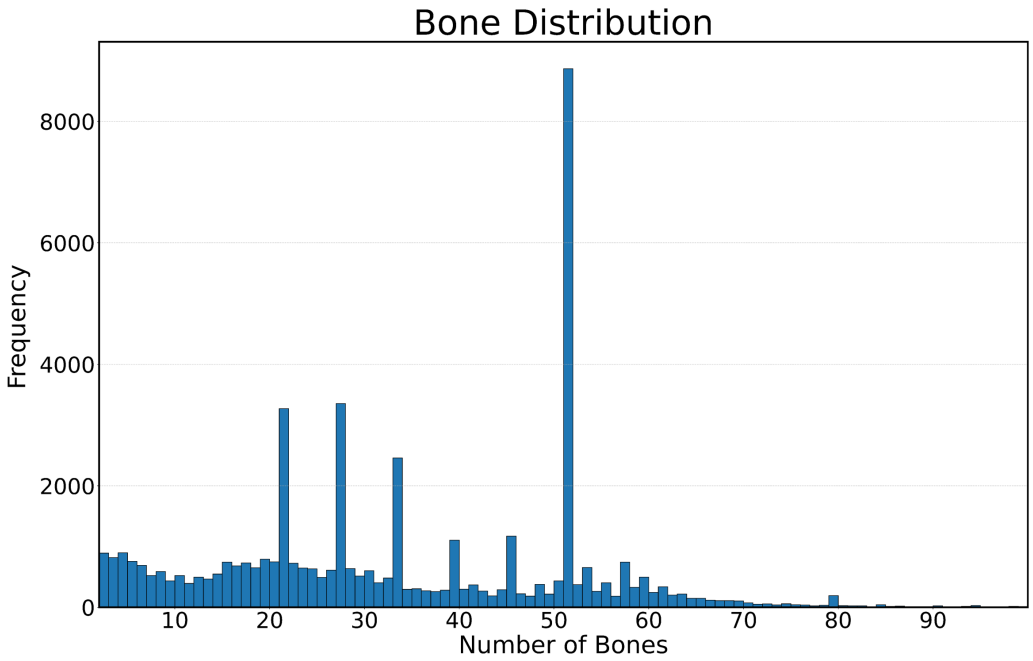


Figure S6: **Bone number distributions of Articulation-XL2.0.**

B More details of Articulation-XL2.0

Our dataset Articulation-XL2.0 is sourced from Objaverse-XL [15, 16], specifically focusing on the GitHub and Sketchfab subsets that include file types containing rigging information (e.g., glb/gltf, fbx, dae, blend, etc.). From an initial 2.97M models, we extract 74K rigged assets (after removing over 150K duplicates) and curate 48K high-quality rigged models through quality verification. Table S5 details these statistics. The number of bones in Articulation-XL2.0 ranges from 2 to 100, with the distribution illustrated in Figure S6.

The diverse-pose subset in Articulation-XL2.0 is derived from two sources. The first component consists of poses extracted from data with animation, where we specifically selected frames exhibiting maximum deviation from rest pose configurations to capture extreme articulations. The second component comprises synthetically generated poses created using SMALR [107, 108] with parameterizations derived from 41 distinct animal scans and randomized valid poses. These randomized valid poses are generated by applying random rotation angles to SMALR animal joints while constraining angles within anatomically valid ranges, ensuring diverse articulation states while maintaining biological plausibility. Refer to Figure S8 for the skeleton structure and joint names of SMALR data. There are originally two joints in index 0, we merge them into a single root joint. Some examples from this diverse-pose collection are illustrated in Figure S7. **Note that the models in the diverse-pose test set, along with their corresponding rest poses, are entirely excluded from the training data, ensuring a rigorous evaluation of generalization to both novel shapes and novel articulations.**

Table S5: **Data statics for Articulation-XL2.0.**

Source	All models	with rigging	high-quality rigging	low-quality rigging
Sketchfab	0.89M	64K	42K	22K
GitHub	2.08M	10K	6K	4K
Total	2.97M	74K	48K	26K

C More related works

Here, we review additional related works that complement the main paper.

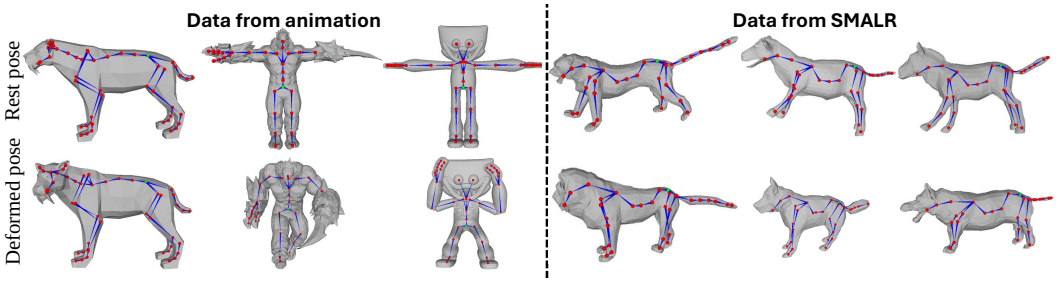
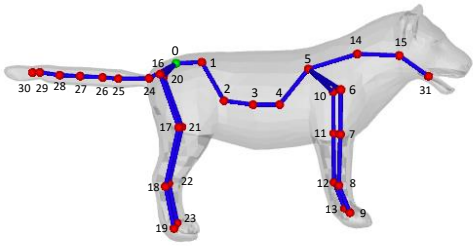


Figure S7: **Examples from the diverse-pose subset of Articulation-XL2.0.** The left group showcases animation-derived samples: the top row displays the original rest poses, while the bottom row shows their corresponding deformed articulations extracted from animation sequences at frames of maximum pose deviation. The right group displays synthetically generated articulations created using SMALR [107, 108].



```
"root_pelvis",      # 0     "spine1",          # 1     "spine2",          # 2
"spine3",           # 3     "spine4",          # 4     "spine5",          # 5
"right_shoulder",   # 6     "right_elbow",     # 7     "right_wrist",     # 8
"right_front_paw",  # 9     "left_shoulder",   # 10    "left_elbow",      # 11
"left_wrist",       # 12    "left_front_paw",  # 13    "neck",            # 14
"head",             # 15    "right_hip",       # 16    "right_knee",      # 17
"right_ankle",      # 18    "right_hind_paw",  # 19    "left_hip",        # 20
"left_knee",        # 21    "left_ankle",      # 22    "left_hind_paw",   # 23
"tail_base",        # 24    "tail1",           # 25    "tail2",           # 26
"tail3",            # 27    "tail4",           # 28    "tail5",           # 29
"tail_end",         # 30    "jaw".             # 31
```

Figure S8: **Skeleton structure and joint names of SMALR data [107, 108].**

3D animation. With the generated rigging, the next step is to animate the 3D models. In contrast to earlier work that focuses on human motion generation [31, 74, 101, 23, 102, 75, 63, 64, 60, 61], we aim to animate diverse 3D object categories that can be rigged. Our pipeline uses a reference video as motion guidance to animate the rigged mesh. The field of 4D generation has experienced rapid growth recently, spanning text/image-to-4D generation [91, 58, 44, 4, 50, 93, 8, 95, 41, 5, 70, 43, 84, 104] and video-to-4D generation [21, 32, 96, 82, 98, 99, 39, 20, 106, 89, 83, 9, 59, 80]. For a comprehensive overview, we refer readers to the survey by Miao et al. [51]. However, most existing 4D generation approaches do not take a specific 3D object as input to generate animations targeted for that object.

Many notable attempts have been made to address object animation. AnyMole [94] introduced a motion in-betweening method for diverse categories with context motions. AnyTop [22] proposed animating 3D rigged meshes using a diffusion model without explicit motion guidance, given only an input skeleton. Millan et al. [52] presented related work but focused exclusively on humanoid models with SMPL [48] proxies. Animate3D [33] proposed animating 3D objects with a Multi-view Video Diffusion Model, which requires extensive training. MotionDreamer [77] attempted to animate 3D models without rigging but produced suboptimal motion quality. The most recent related work, AKD [38], takes a 3D model, manually adds a skeleton, and uses [6] to predict skinning weights. They apply animation to this rigged model using video-based score distillation sampling (SDS). However, their approach is computationally intensive (requiring approximately 25 hours per object) and produces unstable animations with noticeable jittering artifacts. In contrast, we propose an optimization-based method that requires no neural network parameters to achieve more stable animations for diverse object categories by combining our generated rigging with reference video guidance.

D Additional experimental results

D.1 More results of skeleton generation

More qualitative results on AI-generated meshes. We evaluate our method's generalization capability on AI-generated meshes from Tripo2.0 [2] and Hunyuan3D 2.0 [73]. As shown in Figure S9, we compare our method with MagicArticulate [67]. MagicArticulate loses fine details

(e.g., the robot's hand in rows 3 and 5, the dolphin-hummingbird chimera's tail and wings in row 4, marked in yellow) and produces misaligned skeletons (dragon's tail in row 1, deer's legs in row 2). By contrast, our approach consistently generates valid, robust skeletons across all categories.

More qualitative results on test sets. Figure S10 presents additional qualitative comparisons of skeleton generation by our method, MagicArticulate [67], and RigNet [86] on meshes from Articulation-XL2.0, ModelsResource, and the diverse-pose subset. Our method produces more valid skeletons and even corrects artist-created errors, such as the missing deer legs in the first row, and the missing penguin arms in rows 4 and 5.

Sequence ordering. As a supplement to the ablation study in the main paper, Figure S11 compares skeletons generated with hierarchical and spatial ordering. Using spatial ordering always produces disconnected skeletons, as child joints generated before their parents create invalid parent references.

Inference time. We compare average inference times on Articulation-XL2.0-test (see Table S6). Excluding data preprocessing for all methods, our approach is 2.6× faster than Pinocchio [6], 3.0× faster than RigNet [86], 1.9× faster than UniRig [100], and 1.6× faster than MagicArticulate [67].

Table S6: **Inference time of skeleton generation.**

Method	Pinocchio	RigNet	UniRig	MagicArticulate	Ours
Inference time	3.9s	4.5s	2.9s	2.4s	1.5s

D.2 More results of skinning weight prediction

Deformation error report. In addition to the precision, recall, and L1-norm for evaluating skinning weight accuracy presented in the main paper, we also conducted a comprehensive assessment of practical efficacy through deformation error analysis. This supplementary metric quantifies the mean Euclidean distance between vertices deformed via predicted skinning weights and those deformed via ground truth weights across a diverse set of 10 randomly generated poses. As evidenced in Table S7, the proposed methodology exhibits better performance for all experimental datasets.

Ablation studies on block depth. In addition to the ablation results in Section 5.5, we also ablate the block depth. For the green attention block in Figure 2, we vary the number of stacked blocks, which we refer to as depth. When depth is 1 (44.3M parameters), this corresponds to a single block. We incrementally increase the block depth, corresponding to larger model size, and evaluate the resulting performance. The results are shown in Table S8: when depth is set to 2 (87.7M parameters), performance on Articulation-XL2.0 and the diverse-pose subset improves noticeably, but there is a slight degradation on ModelsResource. When depth is 3 (130.9M parameters), performance on Articulation-XL2.0 and the diverse-pose subset shows comparable results to depth=2 but still exhibits a drop on ModelsResource. We attribute the drop on ModelsResource to the orientation distribution difference between Articulation-XL2.0 and ModelsResource: after training on Articulation-XL2.0

Table S7: **Quantitative comparison of skinning weight prediction.** We evaluate our approach against GVB, RigNet, and MagicArticulate. For average L1-norm error (L1) and average distance error (avg Dist.), lower is better. Here, * denotes models trained on Articulation-XL2.0 with the diverse-pose subset.

Method	Articulation-XL2.0		ModelsResource		Diverse-pose	
	L1 ↓	avg Dist. ↓	L1 ↓	avg Dist. ↓	L1 ↓	avg Dist. ↓
GVB	0.745	0.0087	0.687	0.0067	0.786	0.0084
RigNet	0.729	0.0082	0.707	0.0078	0.746	0.0089
MagicArti.	0.451	0.0051	0.642	<u>0.0064</u>	0.479	0.0067
Ours	<u>0.335</u>	<u>0.0043</u>	<u>0.443</u>	**0.0044**	<u>0.405</u>	<u>0.0061</u>
Ours*	**0.333**	**0.0042**	**0.442**	**0.0044**	**0.353**	**0.0053**

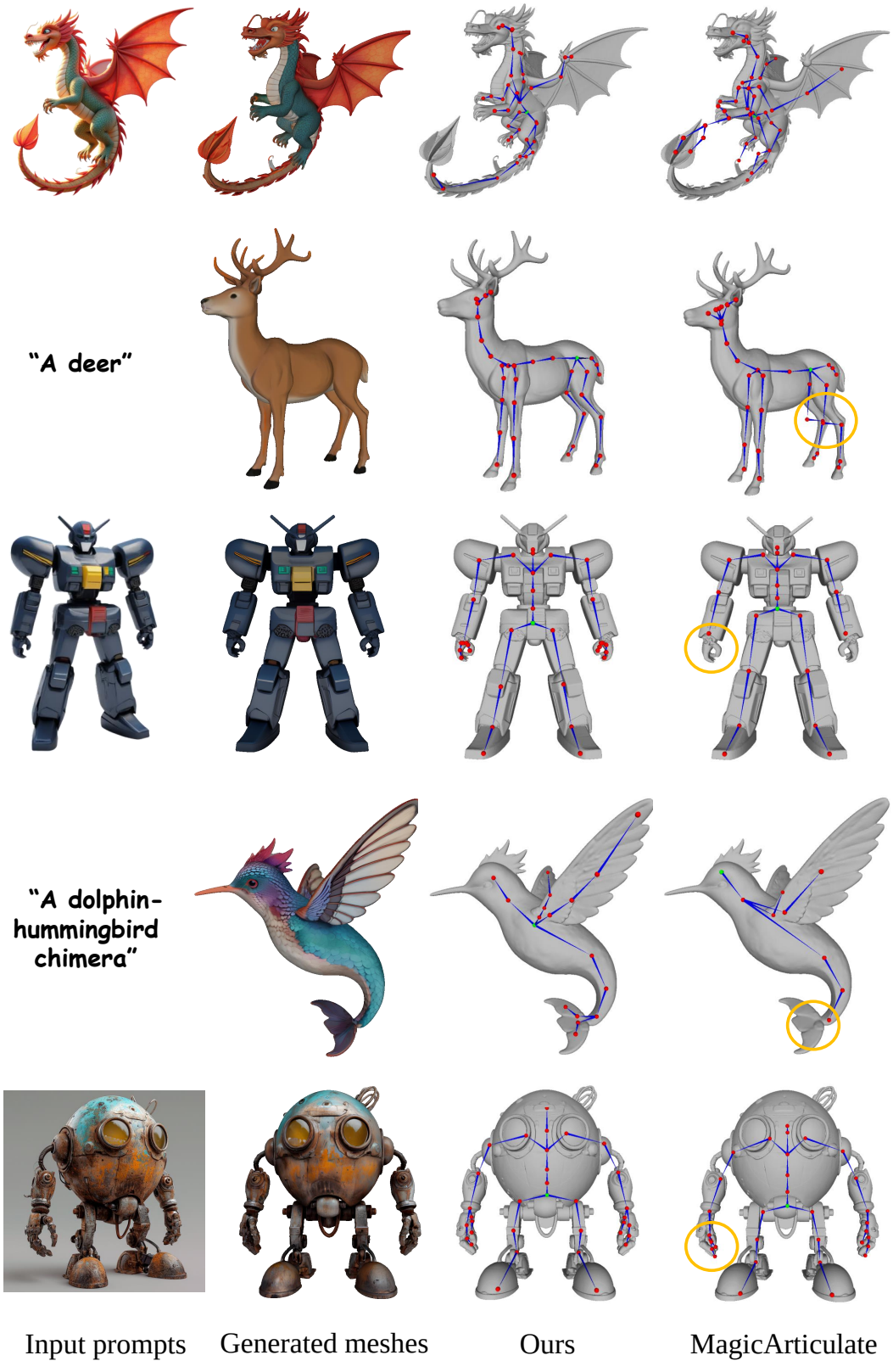


Figure S9: **Comparison of skeleton results on generated meshes.** The meshes are generated by Tripo 2.0 [2] and Hunyuan3D 2.0 [73].

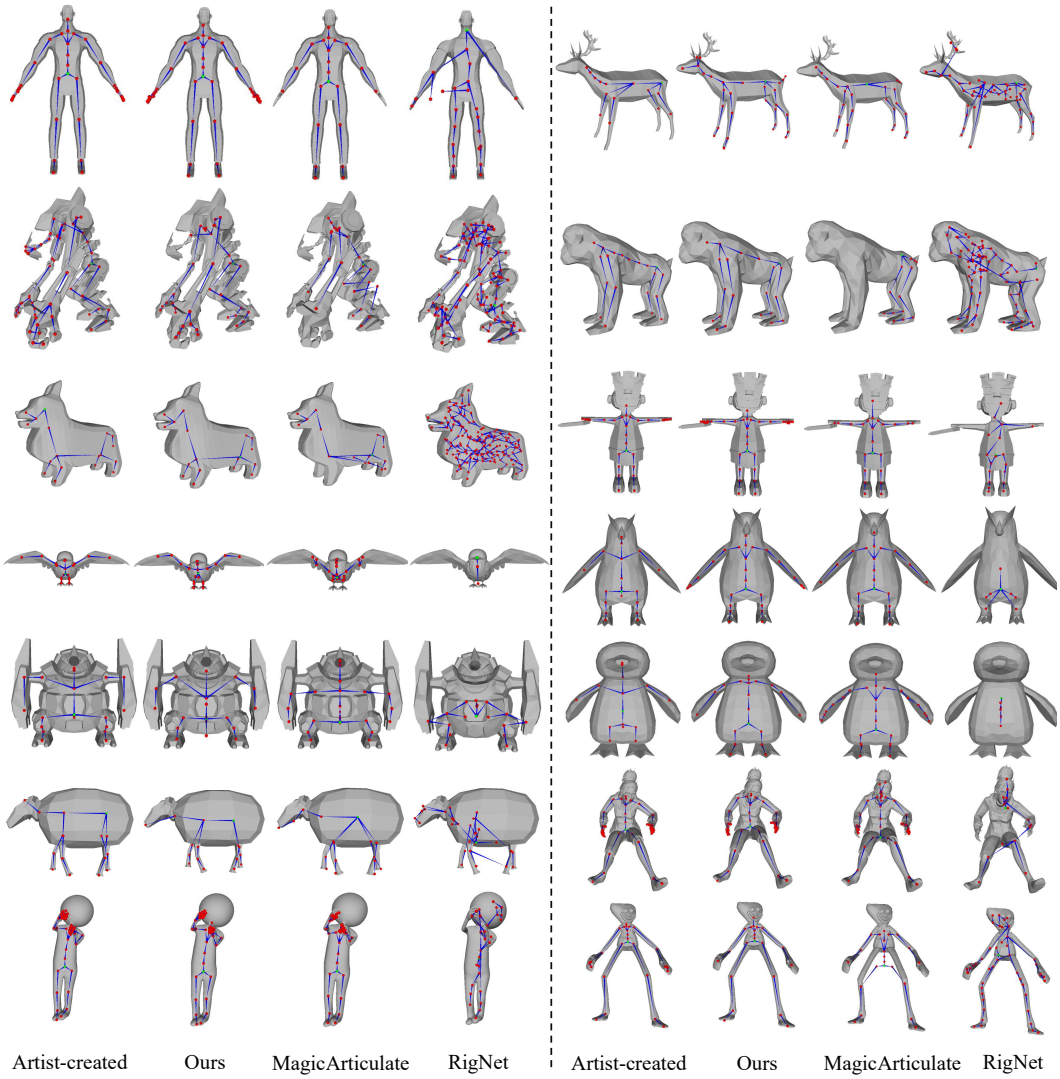


Figure S10: **Comparison of skeleton generation results on test sets.** From the top: six examples from Articulation-XL2.0, four from ModelsResource, and four from the diverse-pose subset. Our method produces valid skeletons and even corrects artist-created errors (e.g., missing deer legs in row 1, missing penguin arms in rows 4–5).

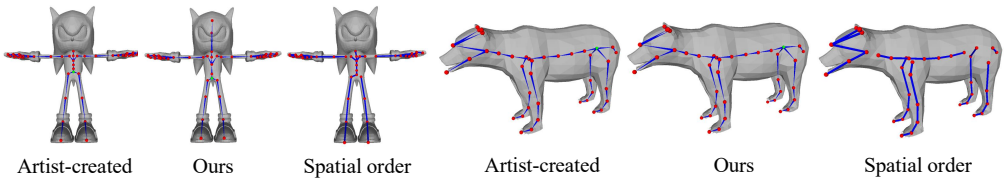


Figure S11: **Comparison of sequence ordering.** Skeletons generated with hierarchical ordering (ours) remain connected, while spatial ordering always yields disconnected structures.

Table S8: **Ablation study on attention block depth in the skinning weight prediction network.**

Method	Articulation-XL2.0			ModelsResource			Diverse-pose		
	Prec. ↑	Rec. ↑	L1 ↓	Prec. ↑	Rec. ↑	L1 ↓	Prec. ↑	Rec. ↑	L1 ↓
depth = 1	87.6%	**74.0%**	0.335	79.7%	**81.6%**	**0.443**	83.6%	**72.2%**	0.405
depth = 2	89.3%	73.0%	**0.316**	**79.8%**	80.2%	0.453	**85.8%**	71.1%	0.392
depth = 3	**89.4%**	72.5%	0.317	79.6%	79.7%	0.461	85.7%	70.8%	**0.391**

Table S9: **Inference time of skinning weight prediction.**

Method	GVB	RigNet	MagicArticulate	Ours
Inference time	1.895s	0.056s	1.430s	0.032s

with highly varied orientations, the larger model may become biased toward diverse orientations, leading to degraded performance when testing on ModelsResource, which contains only front-facing orientations. For depth=3, the comparable performance to depth=2 without obvious improvement suggests that the model capacity may have reached saturation for the current dataset size.

More qualitative results. Figure S12 presents additional qualitative comparisons of skinning weight predictions by our method, MagicArticulate [67], and RigNet [86] on meshes from Articulation-XL2.0, ModelsResource, and the diverse-pose subset. Each example pairs the predicted weight map with its L1 error map against artist-painted references, highlighting our method's superior accuracy across diverse object categories.

Inference time. We also compare average inference times for skinning weight prediction on Articulation-XL2.0-test (see Table S9). Excluding data preprocessing, our method is 59× faster than GVB [18], 1.75× faster than RigNet [86], and 45× faster than MagicArticulate [67].

D.3 More results of animation

More qualitative results. We present more animation results in Figure S13. The input meshes are generated by Tripo 2.0 [2] and Hunyuan3D 2.0 [73]. Despite well-aligned reference views, L4GM [59] consistently produces geometric distortions (highlighted in red), even with ground-truth multi-view renderings. MotionDreamer [77] generates subtle animations and introduces unintended deformations in rigid parts like the turtle shell. In contrast, our approach produces accurate, artifact-free animations with our generated rigging.

User study. We conducted user studies with 21 participants to evaluate animation quality across three methods: L4GM [59], MotionDreamer [77], and our approach. Participants compared 8 animation examples across three evaluation criteria: (1) Video-Animation Alignment: Which animation result shows better alignment with the input video? (2) Motion Quality: Which one has a more natural and realistic motion? (3) 3D Geometry Preservation: Which method better maintains the original 3D object geometry without introducing distortions or artifacts? Results are shown in Section D.3. Our method outperforms both L4GM and MotionDreamer across all three evaluation dimensions. Note that Video-Animation Alignment is not evaluated for MotionDreamer since it uses text-driven motion generation rather than video guidance.

Table S10: **User study evaluation of animation results.**

Method	Video-Animation Align.	Motion Quality	Geometry Preservation
MotionDreamer	-	0	0
L4GM	19.64%	16.67%	18.45%
Ours	**80.36%**	**83.33%**	**81.55%**

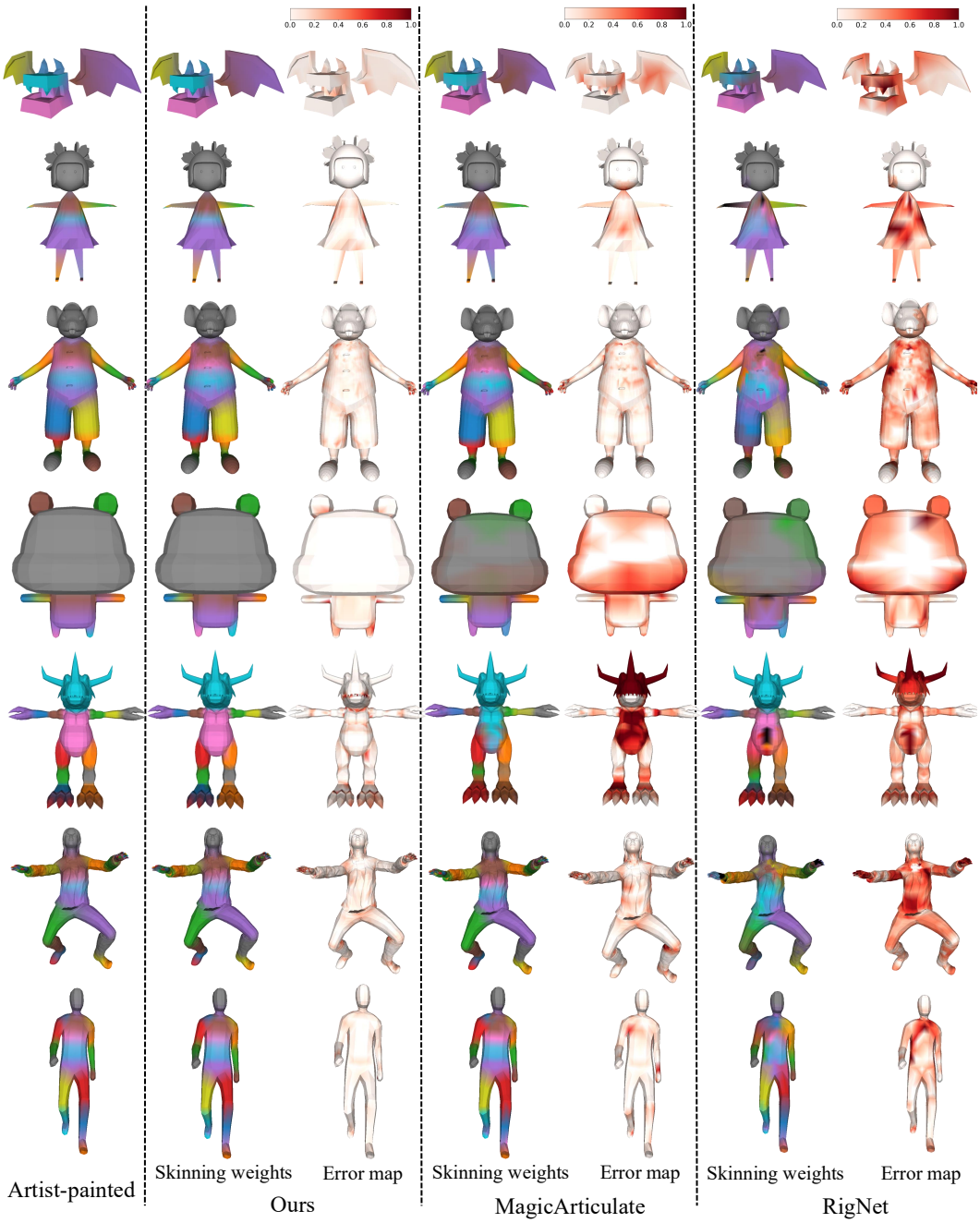


Figure S12: **Comparison of skinning weight prediction results.** From the top: three examples from Articulation-XL2.0, two from ModelsResource, and two from the diverse-pose subset. Each pair shows the predicted weight visualization alongside its L1 error map. Our predictions more closely match the artist-painted references.

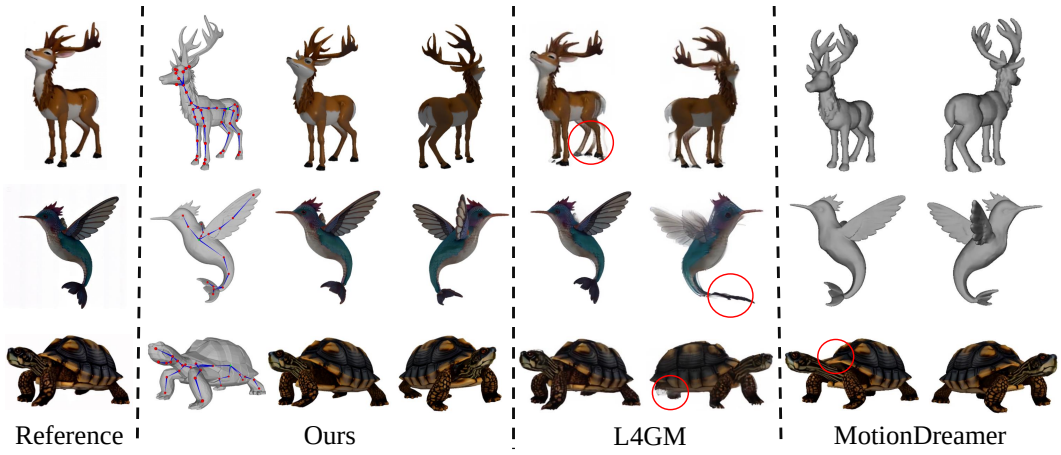


Figure S13: **Comparison of animation results on AI-generated meshes.** The meshes are generated by Tripo 2.0 [2] and Hunyuan3D 2.0 [73]. **The shapes with skeletons represent the rest poses.**

E Discussions

Despite its strong performance, our framework has two main limitations. First, it cannot capture fine-scale deformations, such as flowing hair or fluttering cloth, because no skeletons are generated for these highly deformable parts. A clear example is the swimming turtle sequence on our project page. While the reference video shows very soft motion of the turtle's forelimbs, our animation results appear less smooth due to insufficient joint density in these regions. This limitation stems from our skeleton generation producing few joints in areas requiring fine-scale deformations. Animation-driven joint refinement could improve smoothness but remains future work. Second, the animation stage relies on per-scene optimization, which prevents real-time deployment. An end-to-end feed-forward model that predicts animation directly would eliminate this bottleneck.

Beyond these structural issues, several practical factors also affect animation quality. (1) Complex motion: Rapid movements with large joint rotations present challenges. For instance, in the seahorse case on our project page, while we capture the overall tail oscillation pattern, precise alignment with fine-scale movements remains difficult. Adaptive frame sampling based on optical flow magnitude—sampling more densely during rapid motion—could potentially address these challenges. (2) Video generation quality: Although current text-to-video models (Kling AI [1], JiMeng AI [34]) can generate complex motions with high success rates, video quality directly impacts animation fidelity. Motion blur or temporal inconsistencies degrade joint/vertex tracking accuracy and make optimization more challenging. We mitigate this by generating multiple video candidates and selecting the highest-quality one based on visual clarity and motion consistency. While this approach helps reduce the impact of poor-quality videos, video generation quality still represents a limitation for highly complex motion scenarios. (3) Viewpoint and occlusion issues: Suboptimal camera angles can cause depth ambiguities and tracking failures. While we can select optimal viewpoints that maximize joint visibility using the input 3D mesh, single-view optimization inherently struggles when critical joints remain occluded throughout the sequence. Multi-view priors could provide better geometric understanding of occluded regions.

F Broader impact

Beyond its technical breakthroughs, this work holds significant societal implications by removing the need for specialized expertise and empowering a diverse range of creators who were once excluded from animation tools. As digital environments increasingly influence how we work, learn, and connect, expanding access to animated content creation becomes not only technically valuable but socially essential. However, democratizing animation technology also raises concerns about potential misuse in creating deceptive content and impacts on traditional animation industry employment. Our ultimate vision is to transform 3D animation from an exclusive professional skill into an intuitive creative medium accessible to everyone, while encouraging responsible use.