One Model to Rig Them All: Diverse Skeleton Rigging with *UniRig*

JIA-PENG ZHANG, BNRist, Department of Computer Science and Technology, Tsinghua University, China
CHENG-FENG PU, Zhili College, Tsinghua University, China
MENG-HAO GUO, BNRist, Department of Computer Science and Technology, Tsinghua University, China
YAN-PEI CAO, VAST, China
SHI-MIN HU, BNRist, Department of Computer Science and Technology, Tsinghua University, China

Fig. 1. Diverse 3D models rigged using *UniRig*. The models, spanning various categories including animals, humans, and fictional characters, demonstrate the versatility of our method. Selected models are visualized with their predicted skeletons. © Tira

The rapid evolution of 3D content creation, encompassing both AI-powered methods and traditional workflows, is driving an unprecedented demand for automated rigging solutions that can keep pace with the increasing complexity and diversity of 3D models. We introduce *UniRig*, a novel, unified framework for automatic skeletal rigging that leverages the power of large autoregressive models and a bone-point cross-attention mechanism to generate both high-quality skeletons and skinning weights. Unlike previous methods that struggle with complex or non-standard topologies, *UniRig* accurately predicts topologically valid skeleton structures thanks to a new *Skeleton Tree Tokenization* method that efficiently encodes hierarchical relationships within the skeleton. To train and evaluate UniRig, we present *Rig-XL*, a new large-scale dataset of over 14,000 rigged 3D models spanning a wide range of categories. *UniRig* significantly outperforms state-of-the-art academic and commercial methods, achieving a 215% improvement in rigging accuracy and a 194% improvement in motion accuracy on challenging datasets. Our method works seamlessly across diverse object categories, from detailed anime characters to complex organic and inorganic structures, demonstrating its versatility and robustness. By automating the tedious and time-consuming rigging process, *UniRig* has the potential to speed up animation pipelines with unprecedented ease and efficiency. Project Page: https://zjp-shadow.github.io/works/UniRig/

Authors' addresses: Jia-Peng Zhang, zjp24@mails.tsinghua.edu.cn, BNRist, Department of Computer Science and Technology, Tsinghua University, Beijing, China; Cheng-Feng Pu, pcf22@mails.tsinghua.edu.cn, Zhili College, Tsinghua University, Beijing, China; Meng-Hao Guo, gmh20@mails.tsinghua.edu.cn, BNRist, Department of Computer Science and Technology, Tsinghua University, Beijing, China; Yan-Pei Cao, caoyanpei@gmail.com, VAST, Beijing, China; Shi-Min Hu, shimin@tsinghua.edu.cn, BNRist, Department of Computer Science and Technology, Tsinghua University, Beijing, China.

XXXX-XXXX/2025/4-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Additional Key Words and Phrases: Auto Rigging method, Auto-regressive model

ACM Reference Format:
Jia-Peng Zhang, Cheng-Feng Pu, Meng-Hao Guo, Yan-Pei Cao, and Shi-Min Hu. 2025. One Model to Rig Them All: Diverse Skeleton Rigging with *UniRig* . 1, 1 (April 2025), 18 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The rapid advancements in AI-driven 3D content creation [Holden et al. 2017; Peng et al. 2024; Poole et al. 2022; Siddiqui et al. 2024; Yu et al. 2024; Zhang et al. 2024b] are revolutionizing computer graphics, enabling the generation of complex 3D models at an unprecedented scale and speed. This surge in automatically generated 3D content has created a critical need for efficient and robust rigging solutions, as manual rigging remains a time-consuming and expertise-intensive bottleneck in the animation pipeline. While skeletal animation has long been a cornerstone of 3D animation, traditional rigging techniques often require expert knowledge and hours of time to complete for a single model.

The rise of deep learning has spurred the development of automatic rigging methods, offering the potential to dramatically accelerate this process. Existing methods can be broadly categorized as template-based or template-free. Template-based approaches [Chu et al. 2024; Li et al. 2021; Liu et al. 2019] rely on predefined skeleton templates (e.g., SMPL [Loper et al. 2023]) and achieve high accuracy in predicting bone positions within those templates. However, they are limited to specific skeleton topologies and struggle with models that deviate from the predefined templates. Template-free methods, such as RigNet [Xu et al. 2020], offer greater flexibility by predicting skeleton joints and their connectivity without relying on a template. However, these methods often produce less stable results and may generate topologically implausible skeletons. Furthermore, retargeting motion to these generated skeletons can be challenging.

Another line of research has explored skeleton-free mesh deformation [Aigerman et al. 2022; Liao et al. 2022; Wang et al. 2023b], which bypasses the need for explicit skeleton structures. While these methods offer intriguing possibilities, they often rely heavily on existing motion data, making them less generalizable to new and unseen motions. They also tend to be less compatible with established industry pipelines that rely on skeletal animation. Fully neural network-based methods can be computationally expensive, limiting their applicability in resource-constrained scenarios.

Despite these advancements, existing automatic rigging techniques still fall short in addressing the growing demand for rigging diverse 3D models. As highlighted in Table 1, many methods are limited to specific model categories, struggle with complex topologies, or rely on manual intervention. To overcome these limitations, we propose *UniRig*, a novel learning-based framework for automatic rigging of diverse 3D models.

A key challenge in automatic rigging is the inherent complexity of representing and generating valid skeleton structures. They possess a hierarchical tree structure with complex interdependencies between joints. Previous template-free methods often struggled to accurately capture these topological constraints, leading to unstable or unrealistic skeletons. *UniRig* addresses this challenge by leveraging the power of autoregressive models, which excel at capturing sequential dependencies and generating structured outputs. Specifically, *UniRig* employs an autoregressive model to predict the skeleton tree in a topologically sorted order, ensuring the generation of valid and well-structured skeletons. This is enabled by a novel *Skeleton Tree Tokenization* method that efficiently encodes the skeleton's hierarchical structure into a sequence of tokens. This tokenization scheme is designed to explicitly represent the parent-child relationships within the skeleton tree, guiding the autoregressive model to produce topologically sound outputs. Furthermore, the tokenization incorporates information about specific bone types (e.g., spring bones, template bones), facilitating downstream tasks such as motion retargeting. *UniRig* also leverages a Bone-Point Cross Attention mechanism to accurately predict skinning weights, capturing the complex relationships between the generated skeleton and the input mesh.

To train *UniRig* , we curated *Rig-XL* , a new large-scale dataset of over 14,000 3D models with diverse skeletal structures and corresponding skinning weights. *Rig-XL* significantly expands upon existing datasets in terms of both size and diversity, enabling us to train a highly generalizable model. We also leverage *VRoid*, a dataset of anime-style characters, to refine our model's ability to handle detailed character models.

Our contributions can be summarized as follows:

- We propose a novel Skeleton Tree Tokenization method that efficiently encodes skeletal structures, enabling the autoregressive model to generate topologically valid and well-structured skeletons.
- We curate and present *Rig-XL* , a new large-scale and diverse dataset of 3D rigged models. This dataset has been carefully cleaned and provides a high-quality, generalized resource for subsequent auto-rigging tasks.
- We introduce *UniRig*, a unified framework for automatic rigging that combines an autoregressive model for skeleton prediction with a Bone-Point Cross Attention mechanism for skin weight prediction. We demonstrate that *UniRig* achieves state-of-the-art results in both skeleton prediction and skinning weight prediction, outperforming existing methods on a wide range of object categories and skeletal structures.

2 RELATED WORKS

2.1 Data-Driven Mesh Deformation Transfer

The skeleton animation system [Marr and Nishihara 1978] is a foundational technique in computer graphics animation. However, some studies [Xu et al. 2020; Zhang et al. 2023a] suggest that mastering rigging methods can be challenging for non-experts. Recently, in the field of character animation, driven by advancements in deep learning and the availability of numerous datasets [Blackman 2014; Chu et al. 2024; Models-Resource 2019; Xu et al. 2019], mesh-deformation methods that bypass traditional rigging processes have emerged. These methods can be broadly classified into two categories, as outlined below:

senza scheletro

2.1.1 Skeleton-free Mesh Deformation. Some methods [Wang et al. 2023a; Zhang et al. 2024a] bypass the explicit representation of a

Table 1. **Comparison of *UniRig* with Prior Work in Automatic Rigging.** ∗ Tripo supports only human and quadruped categories. † Inference time depends on the number of bones and the complexity of the model.

Method	Template Based	Template Free	Automation Level	Multi Categories	Cost Time
RigNet [Xu et al. 2020]	✗	✓	Automated	✓	1s ∼ 20min†
NBS [Li et al. 2021]	✓	✗	Automated	✗	1 s
TaRig [Ma and Zhang 2023]	✓	✓	Automated	✗	30 s
Anything World [Anything-World 2024]	✓	✓	Semi-Automated	✓	5 min
Tripo [VAST 2025]	✓	✓	Automated	✓∗	2 min
Meshy [Meshy 2024]	✓	✗	Semi-Automated	✗	1 ∼ 2 min
Accurig [Auto-Rig 2024]	✓	✗	Semi-Automated	✗	1 min
UniRig (Ours)	✓	✓	Automated	✓	1 ∼ 5 s

skeleton and instead learn to directly deform the mesh based on input parameters or learned motion patterns.

SfPT [Liao et al. 2022] introduces a center-based Linear Blend Skinning (LBS) [Kavan et al. 2007] method and constructs a Pose Transfer Network that leverages deep learning to facilitate motion transfer across characters. Building on this approach, HMC [Wang et al. 2023a] proposes an iterative method for mesh deformation prediction, improving accuracy by refining predictions from coarse to fine levels. Tapmo [Zhang et al. 2023a], inspired by SfPT, employs a Mesh Handle Predictor and Motion Diffusion to generate motion sequences and retarget them to diverse characters.

2.1.2 Vertex Displacement Prediction. Another approach is to drive entirely through neural networks, and some research[Groueix et al. 2018; Yu et al. 2025] efforts have also explored this. [Wang et al. 2020] introduced the first neural pose transfer model for human characters. [Gao et al. 2018] proposed a VAE-Cycle-GAN framework that uses cycle consistency loss between source and target characters to predict mesh deformation automatically. ZPT [Wang et al. 2023b] develops a correspondence-aware shape understanding module to enable zero-shot retargeting of stylized characters.

While promising, the skeleton-free and direct vertex displacement approaches described in Sections 2.1.1 and 2.1.2 face challenges in integrating with established industry workflows, which heavily rely on traditional skeletal rigging and animation systems.

2.2 Automatic Rigging Methods

Automatic rigging aims to automate the process of creating a skeleton and associating it with a 3D mesh. Existing approaches can be categorized as either traditional geometry-based methods or more recent deep learning-based techniques.

2.2.1 Traditional Geometric Methods. Early methods [Amenta and Bern 1998; Tagliasacchi et al. 2009] relied on traditional geometric features to predict skeletons without requiring data. Pinocchio [Baran and Popović 2007] approximates the medial surface using signed distance fields and optimizes skeleton embedding via discrete penalty functions. Geometric techniques like Voxel Cores [Yan et al. 2018] and Erosion Thickness [Yan et al. 2016], which fit medial axes and surfaces, also use these structures to drive 3D meshes in a manner similar to skeletons. Although these traditional methods can effectively handle objects with complex topologies, they often require significant manual intervention within industrial pipelines. For instance, tools such as LazyBones [Nile 2025], based on medial axis fitting, still necessitate considerable animator input to fine-tune skeletons before they can be used in production.

2.2.2 Deep Learning Algorithms. With the rapid advancement of deep learning, several data-driven auto-rigging methods [Liu et al. 2019; Ma and Zhang 2023; Wang et al. 2025] have emerged in animation. RigNet [Xu et al. 2020] is a notable example, which uses animated character data to predict joint heatmaps and employs the Minimum Spanning Tree algorithm to connect joints, achieving automatic skeletal rigging for various objects. MoRig [Xu et al. 2022] enhances RigNet by using a motion encoder to capture geometric features, improving both accuracy and precision in the joint extraction process. To address the artifacts commonly seen in LBS-based systems, Neural Blend Shapes [Li et al. 2021] introduces a residual deformation branch to improve deformation quality at joint regions. DRiVE [Sun et al. 2024] applies Gaussian Splatting conditioned Diffusion to predict joint positions. However, these methods often require a separate step to infer bone connectivity from the predicted joints, which can introduce topological errors.

Many existing deep learning-based methods suffer from limitations that hinder their widespread applicability. Some methods are restricted to specific skeleton topologies (e.g., humanoids), while others rely on indirect prediction of bone connections, leading to potential topological errors. These methods often struggle to balance flexibility with stability and precision. Our work addresses these limitations by leveraging an autoregressive model for skeleton prediction. This approach is inspired by recent advancements in 3D autoregressive generation [Chen et al. 2024; Hao et al. 2024; Siddiqui et al. 2024] that have shown promise in modeling 3D shapes using tokenization and sequential prediction.

3 OVERVIEW

The core challenge in automated skeletal rigging lies in accurately predicting both a plausible skeleton structure and the associated skinning weights that define mesh deformation. Previous methods often struggle with the diversity of 3D model topologies, requiring manual intervention or specialized approaches for different categories. To address this, we propose *UniRig* , a unified learning-based framework for rigging diverse 3D models. *UniRig* employs a novel paradigm that effectively combines two learned models into a single streamlined rigging process. It consists of two key stages: (1) autoregressive skeleton tree prediction from an input mesh (Section 5), leveraging a novel tokenization method for efficient processing, and (2) efficient per-point skin weight prediction conditioned on the

Fig. 2. Examples from *Rig-XL* , demonstrating well-defined skeleton structures.

predicted skeleton, using a Bone-Point Cross Attention mechanism (Section 6).

To train and evaluate *UniRig* , we introduce two datasets: VRoid (Section 4.1), a collection of anime-style 3D human models, and *Rig-XL* (Section 4.2), a new large-scale dataset spanning over 14,000 diverse and high-quality 3D models. VRoid helps refine our method's ability to model fine details, while *Rig-XL* ensures generalizability across a wide range of object categories.

We evaluate UniRig's performance through extensive experiments (Section 7), comparing it against state-of-the-art methods and commercial tools. Our results demonstrate significant improvements in both rigging accuracy and animation fidelity. We further showcase UniRig's practical applications in human-assisted auto-rigging and character animation (Section 8). Finally, we discuss limitations and future work (Section 9).

4 DATASET

4.1 VRoid Dataset Curation

To facilitate the development of detailed and expressive skeletal rigs, particularly for human-like characters, we have curated a dataset of 2, 061 anime-style 3D models from VRoidHub [Isozaki et al. 2021].

This dataset, which we refer to as *VRoid*, is valuable for training models capable of capturing the nuances of character animation, including subtle movements and deformations. It complements our larger and more diverse Rig-XL dataset (Section 4.2) by providing a focused collection of models with detailed skeletal structures.

The VRoid dataset was compiled by first filtering the available models on VRoidHub based on the number of bones. These models were further refined through a manual selection process to ensure data quality and consistency in skeletal structure and to eliminate models with incomplete or improperly defined rigs.

4.1.1 VRM Format. The models in the VRoid dataset are provided in the VRM format, a standardized file format for 3D avatars used in virtual reality applications. A key feature of the VRM format is its standardized humanoid skeleton definition, which is compatible with the widely used Mixamo [Blackman 2014] skeleton. This standardization simplifies the process of retargeting and animating these models. Furthermore, the VRM format supports *spring bones* [Isozaki et al. 2021], which are special bones that simulate physical interactions like swaying and bouncing. These spring bones are crucial for creating realistic and dynamic motion in parts of the model such as hair, clothing, and tails, as demonstrated in Figure 6. The behavior of these spring bones is governed by a physics simulation, detailed in Section 6.2. The inclusion of spring bones in the VRoid dataset allows our model to learn to generate rigs that support these dynamic effects, leading to more lifelike and engaging animations.

4.2 *Rig-XL* Dataset Curation

To train a truly generalizable rigging model capable of handling diverse object categories, a large-scale dataset with varied skeletal structures and complete skinning weights is essential. To this end, we curated *Rig-XL* , a new dataset derived from the Objaverse-XL dataset [Deitke et al. 2024], which contains over 10 million 3D models. While Objaverse-XL is a valuable resource, it primarily consists of static objects and lacks the consistent skeletal structure and skinning weight information required for our task. We address this by filtering and refining the dataset.

We initially focused on a subset of 54,000 models from Objaverse-XL provided by Diffusion4D [Liang et al. 2024], as these models exhibit movable characteristics and better geometric quality compared to the full dataset. However, many of these models were unsuitable for our purposes due to issues such as scene-based animations (multiple objects combined), the absence of skeletons or skinning weights, and a heavy bias towards human body-related models. This necessitated a rigorous preprocessing pipeline to create a high-quality dataset suitable for training our model.

4.2.1 Dataset Preprocessing. Our preprocessing pipeline addressed the aforementioned challenges through a combination of empirical rules and the use of vision-language models (VLMs). This pipeline involved the following key steps:

1 **Skeleton-Based Filtering:** We retained only the 3D assets with a bone count within the range of [10, 256], while ensuring that each asset has a single, connected skeleton tree. This step ensured that each model had a well-defined skeletal structure while removing overly simplistic or complex models and scenes containing multiple objects.
2 **Automated Categorization:** We rendered each object under consistent texture and illumination conditions and deduplicated objects by computing the perceptual hashing value of the rendered images [Farid 2021]. We then employed the vision-language model ChatGPT-4o [Hurst et al. 2024] to generate descriptive captions for each model. These captions were used to categorize the models into eight groups: Mixamo, Biped, Quadruped, Bird & Flyer, Insect & Arachnid, Water Creature, Static, and Other. Specifically, Static means some static objects such as pillows. This categorization, based on semantic understanding, allowed us to address the long-tail distribution problem and ensure sufficient representation of various object types. Notably, we pre-screened skeletons conforming to the Mixamo [Blackman 2014] format by their bone names and placed them in a separate category.
3 **Manual Verification and Refinement:** We re-rendered each model with its skeleton displayed to enable manual inspection of the skeletal structure and associated data. This crucial step allowed us to identify and correct common errors. One such issue is the incorrect marking of bone edges as "not connected," which can result in many bones being directly connected to the root and an unreasonable topology. These issues introduce bias during network training and deviate from expected anatomical configurations. Specific corrections are detailed in Appendix A.1.1.

4.2.2 Dataset Details. After this rigorous preprocessing, the *Rig-XL* dataset comprises 14,611 unique 3D models, each with a well-defined skeleton and complete skinning weights. The distribution across the eight categories is shown in 3. Notably, human-related models (Mixamo and Biped) are still dominant, reflecting the composition of the original Objaverse-XL. 4 shows the distribution of skeleton counts, with a primary mode at 52, corresponding to Mixamo models with hands, and a secondary mode at 28, corresponding to Mixamo models without hands. This detailed breakdown of the dataset's composition highlights its diversity and suitability for training a generalizable rigging model.

5 AUTOREGRESSIVE SKELETON TREE GENERATION

Predicting a valid and well-formed skeleton tree from a 3D mesh is a challenging problem due to the complex interdependencies between joints and the need to capture both the geometry and topology of the underlying structure. Unlike traditional methods that often rely on predefined templates or struggle with diverse topologies, we propose an autoregressive approach that generates the skeleton tree sequentially, conditioning each joint prediction on the previously generated ones. This allows us to effectively model the hierarchical relationships inherent in skeletal structures and generate diverse, topologically valid skeleton trees.

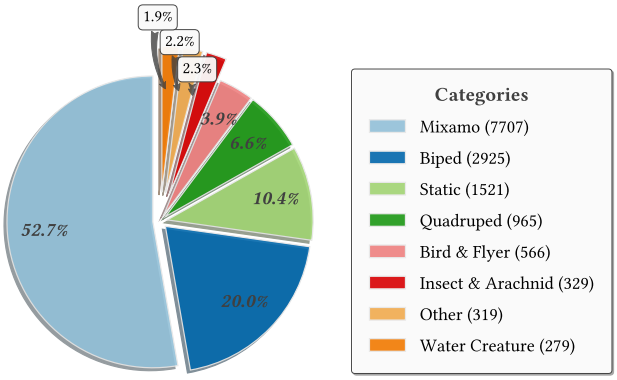

Fig. 3. **Category distribution of *Rig-XL* .** The percentages indicate the proportion of models belonging to each category.

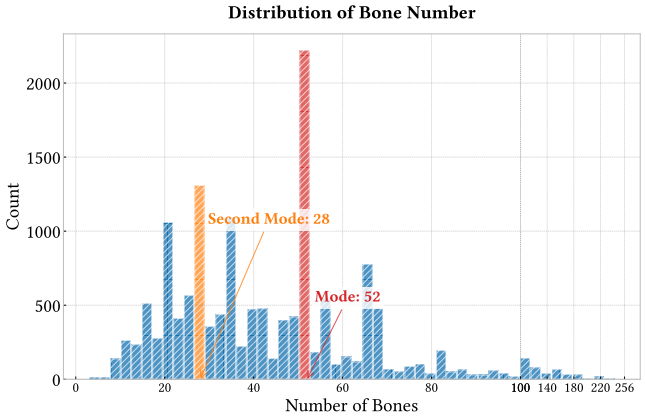

Fig. 4. **Distribution of bone numbers in *Rig-XL* .** The histogram shows the frequency of different bone counts across all models in the dataset.

Formally, let $\mathcal{M} = \{\mathcal{V} \in \mathbb{R}^{V\times 3}, \mathcal{F}\}$ represent a 3D mesh, where $\mathcal{V}$ denotes the set of vertices and $\mathcal{F}$ represents the faces. Our goal is to predict the joint positions $\mathcal{J} \in \mathbb{R}^{J\times 3}$, where J is the number of bones, along with the joint-parent relationships $\mathcal{P} \in \mathbb{N}^{J-1}$ that define the connectivity of the skeleton tree.

To facilitate this prediction, we first convert the input mesh ($\mathcal{M}$) into a point cloud representation that captures both local geometric details and overall shape information. We sample $N = 65536$ points from the mesh surface $\mathcal{F}$, yielding a point cloud $\mathcal{X} \in \mathbb{R}^{N\times 3}$ and corresponding normal vectors $\mathcal{N} \in \mathbb{R}^{N\times 3}$. Point clouds provide a flexible and efficient representation for capturing the geometric features of 3D shapes, and the inclusion of surface normals encodes important information about local surface orientation. The point cloud is normalized to coordinates within the range $[-1, 1]^3$. These vectors are then passed through a geometric encoder $E_G : (\mathcal{X}, \mathcal{N}) \mapsto \mathcal{F}_G \in \mathbb{R}^{V\times F}$, where F denotes the feature dimension, generating the geometric embedding $\mathcal{F}_G$. We utilize a shape encoder based on the 3DShape2Vecset representation [Zhang et al. 2023b] due to its proven ability to capture fine-grained geometric details of 3D

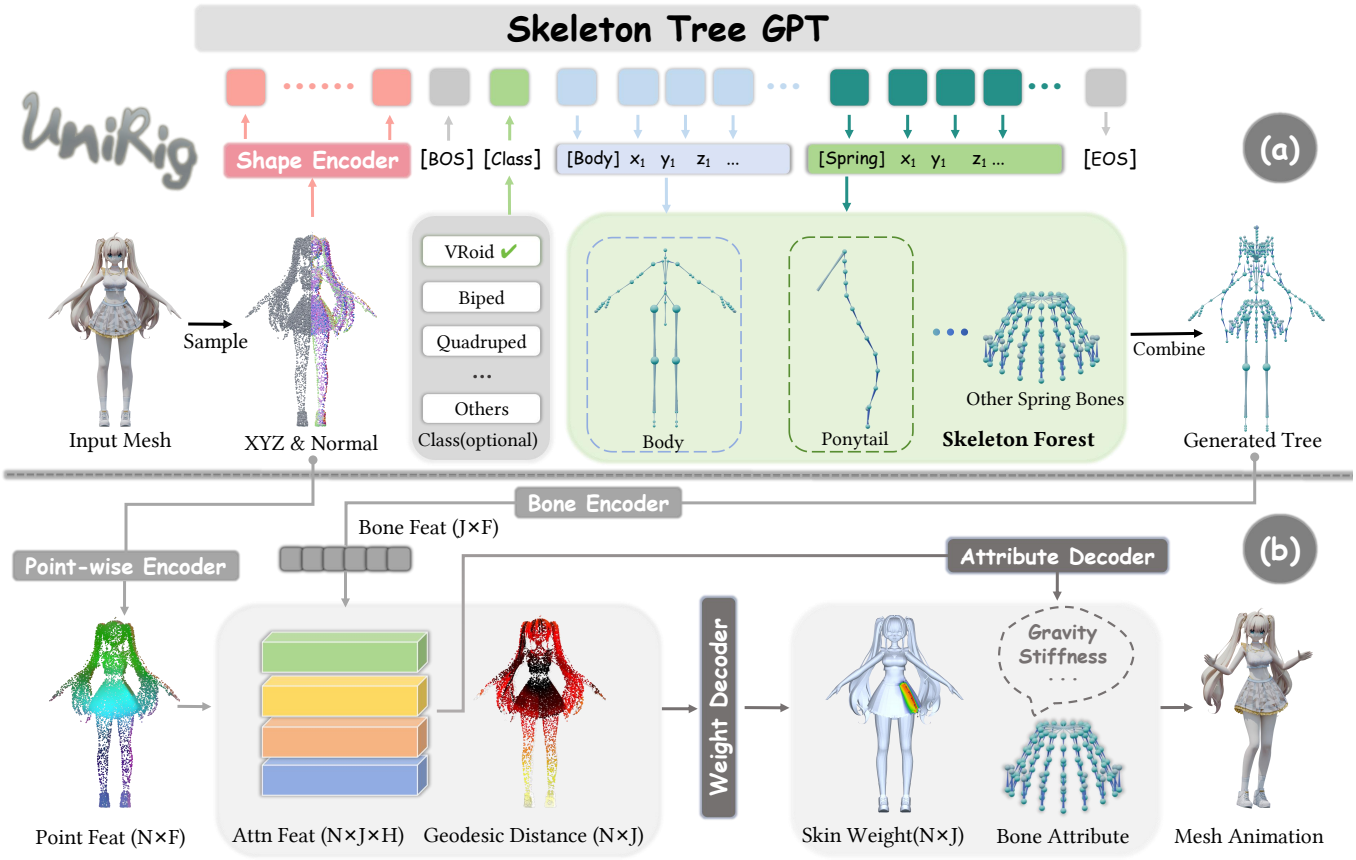

Fig. 5. **Overview of the *UniRig* framework.** The framework consists of two main stages: (a) **Skeleton Tree Prediction** and (b) **Skin Weight Prediction**. **(a)** The skeleton prediction stage (detailed in Section 5) takes a point cloud sampled from the 3D meshes as input, which is first processed by the Shape Encoder to extract geometric features. These features, along with optional class information, are then fed into an autoregressive Skeleton Tree GPT to generate a token sequence representing the skeleton tree. The token sequence is then decoded into a hierarchical skeleton structure. **(b)** The skin weight prediction stage (detailed in Section 6) takes the predicted skeleton tree from (a) and the point cloud as input. A Point-wise Encoder extracts features from the point cloud, while a Bone Encoder processes the skeleton tree. These features are then combined using a Bone-Point Cross Attention mechanism to predict the skinning weights and bone attributes. Finally, the predicted rig can be used to animate the mesh. © kinoko7

objects. For the encoder E_G, we do not use any pretrained weights but instead initialize its parameters randomly using a Gaussian distribution. The resulting geometric embedding $\mathcal{F}_G$ serves as a conditioning context for the autoregressive generation process.

We employ an autoregressive model based on the OPT architecture [Zhang et al. 2022] to sequentially generate the skeleton tree. OPT's decoder-only transformer architecture is well-suited for this task due to its ability to model long-range dependencies and generate sequences in a causally consistent manner. To adapt OPT for skeleton tree generation, we first need to represent the tree $\{\mathcal{J}, \mathcal{P}\}$ as a discrete sequence $\mathcal{S}$. This is achieved through a novel tree tokenization process (detailed in Section 5.1) that converts the tree structure into a sequence of tokens, enabling the autoregressive model to process it effectively.

During training, the autoregressive model is trained to predict the next token in the sequence based on the preceding tokens and the geometric embedding $\mathcal{F}_G$. This is achieved using the Next Token Prediction (NTP) loss, which is particularly well-suited for training autoregressive models on sequential data. The NTP loss is formally defined as:

$$\mathcal{L}_{\text{NTP}} = -\sum_{t=1}^{T} \log P(s_t \mid s_1, s_2, \ldots, s_{t-1}, \mathcal{F}_G),$$

where T denotes the total sequence length $\mathcal{S} = \{s_1, s_2, \ldots, s_T\}$, and $P(s_t \mid s_1, \ldots, s_{t-1})$ is the conditional probability of token s_t given the preceding tokens in the sequence. By minimizing this loss, the model learns to generate skeleton trees that are both geometrically consistent with the input mesh and topologically valid, as evidenced by the quantitative results in Table 3 and Supplementary Table 9. The geometric embedding $\mathcal{F}_G$ is prepended to the tokenized sequence to provide the necessary geometric context for the autoregressive generation.

5.1 Skeleton Tree Tokenization

A core challenge in autoregressively predicting skeleton trees is representing the tree structure in a sequential format suitable for a transformer-based model. This involves encoding both the spatial coordinates of each bone and the hierarchical relationships between bones. A naive approach would be to simply concatenate the coordinates of each bone in a depth-first or breadth-first order. However, this approach leads to several challenges, including difficulty in enforcing structural constraints, redundant tokens and inefficient training and inference.

To address these challenges, we propose a novel skeleton tree tokenization scheme. Inspired by recent advances in 3D generative model [Chen et al. 2024; Hao et al. 2024; Siddiqui et al. 2024], our method discretizes the continuous bone coordinates and employs special tokens to represent structural information. While inspired by these 3D generation approaches, our tokenization scheme is specifically designed for the unique challenge of representing the *hierarchical structure* of a skeleton tree in a sequential format suitable for autoregressive rigging.

We first discretize the normalized bone coordinates, which lie in the range $[-1, 1]$, into a set of D= 256 discrete tokens. This is done by mapping the continuous values to integers using the following function: $M : x \in [-1, 1] \mapsto d = \lfloor \frac{x+1}{2} \times D \rfloor \in \mathbb{Z}_D$. The inverse mapping is given by: $M^{-1} : d \in \mathbb{Z}_D \mapsto x = \frac{2d}{D} - 1 \in [-1, 1]$. This discretization allows us to represent bone coordinates as sequences of discrete tokens. The average relative error during discretization is $O(\frac{1}{D})$, which is negligible for our application.

Let $\mathcal{J}_i$ be the i-th joint in the skeleton tree. We define the discrete index of the i-th bone as $d_i = (dx_i, dy_i, dz_i)$, where $dx_i = M(\mathcal{J}_i(x))$, $dy_i = M(\mathcal{J}_i(y))$, and $dz_i = M(\mathcal{J}_i(z))$ are the discretized coordinates of the tail of the i-th bone.

A straightforward way to tokenize the skeleton tree would be to concatenate these bone tokens in a topological order (e.g., depth-first), resulting in a sequence like:

$$\texttt{<bos>}\ dx_1\ dy_1\ dz_1\ dx_{\mathcal{P}_2}\ dy_{\mathcal{P}_2}\ dz_{\mathcal{P}_2}\ dx_2\ dy_2\ dz_2 \cdots dx_{\mathcal{P}_T}\ dy_{\mathcal{P}_T}\ dz_{\mathcal{P}_T}\ dx_T\ dy_T\ dz_T\ \texttt{<eos>}$$

where **<bos>** and **<eos>** denote the beginning and end of the sequence, respectively, and $\mathcal{P}_i$ denotes the parent joint of the i-th joint.

However, this naive approach has several drawbacks. First, it introduces redundant tokens, as the coordinates of a joint are repeated for each of its children. Second, it does not explicitly encode the different types of bones (e.g., spring bones, template bones), which can have different structural properties. Finally, during inference, we observed that this representation often leads to repetitive token sequences.

To overcome these limitations, we propose an optimized tokenization scheme that leverages the specific characteristics of skeletal structures. Our key insight is that decomposing skeleton tree into certain bone sequences, such as spring bones in VRoid models or bones belonging to a known template (e.g., Mixamo), can be represented more compactly. Furthermore, explicitly encoding these bone types using dedicated type identifiers provides valuable information to the model, improving its ability to learn and generalize to different skeletal structures. For instance, knowing that a bone belongs to a specific template (e.g., Mixamo) allows for efficient motion retargeting, as the mapping between the template and the target skeleton is already known.

We introduce special "type identifier" tokens, denoted as **<type>**, to indicate the type of a bone sequence. For example, a sequence of spring bone chain can be represented as

$$\texttt{<spring\_bone>}\ dx_s\ dy_s\ dz_s\ \ldots\ dx_t\ dy_t\ dz_t,$$

where $dx_s\ dy_s\ dz_s$ and $dx_t\ dy_t\ dz_t$ are the discretized coordinates of the first and last spring bones in the chain, respectively. Similarly, bones belonging to a template can be represented using a template identifier, such as **<mixamo:body>**. This allows us to omit the parent coordinates for bones in a template, as they can be inferred from the template definition. We also add a class token **<cls>** (e.g. **<mixamo>**) at the beginning of each sequence.

This results in a more compact tokenized sequence:

$$\texttt{<bos>}\ \texttt{<cls>}\ \texttt{<type}_1\texttt{>}\ dx_1\ dy_1\ dz_1\ dx_2\ dy_2\ dz_2 \cdots \texttt{<type}_2\texttt{>} \ldots \texttt{<type}_k\texttt{>} dx_t\ dy_t\ dz_t \ldots dx_T\ dy_T\ dz_T\ \texttt{<eos>}$$

For more general cases where no specific bone type can be identified, we use a Depth-First Search (DFS) algorithm to identify and extract linear bone chains, and represent them as compact subsequences. The DFS traversal identifies separate bone chains (branches) originating from the main skeleton structure or forming disconnected components. Each newly identified branch is then prefixed with a **<branch_token>** in the token sequence. We also ensure the children of each joint are sorted based on their tail coordinates (z, y, x) order in the rest pose(where the z-axis represents the vertical direction in our coordinate convention). This maintains a consistent ordering that respects the topological structure of the skeleton. The specific steps of this optimized tokenization process are summarized in Algorithm 1.

For instance, consider an anime-style 3D girl with a spring-bone-based skirt, as shown in Figure 5(a). Using our optimized tokenization, this could be represented as:

$$\texttt{<bos>}\ \texttt{<VRoid>}\ \texttt{<mixamo:body>}\ dx_1\ dy_1\ dz_1 \ldots dx_{22}\ dy_{22}\ dz_{22}\ \texttt{<mixamo:hand>}\ dx_{23}\ dy_{23}\ dz_{23} \ldots dx_{52}\ dy_{52}\ dz_{52} \ldots\ \texttt{<spring\_bone>}\ dx_s\ dy_s\ dz_s \ldots dx_t\ dy_t\ dz_t \ldots\ \texttt{<eos>}$$

This demonstrates how our tokenization scheme compactly represents different bone types and structures.

During de-tokenization, connectivity between different bone chains (identified by their respective tokens) is established by merging joints whose decoded coordinates fall within a predefined distance threshold, effectively reconstructing the complete skeleton tree.

This optimized tokenization significantly reduces the sequence length compared to the naive approach. Formally, the naive approach requires $6T - 3 + K$ tokens (excluding **<bos>** and **<eos>**), where T is the number of bones. In contrast, our optimized tokenization requires only $3T + M + S \times 4 + 1$ tokens, where M is the number of templates (usually less than 2), and S is the number of branches in the skeleton tree after removing the templates to form a forest. As

ALGORITHM 1: Skeleton Tree Tokenization

Input: *bones* $\mathcal{B} = (\mathcal{J}_\mathcal{P}, \mathcal{J}) \in \mathbb{R}^{J\times 6}$ (with skeleton Tree structure), *templates* $\mathcal{T}$ and class type of dataset C
Output: *token sequence* $\mathcal{S} \in \mathbb{N}^T$

Function tokenize(*bones* $\mathcal{B}$, *templates* $\mathcal{T}$, *class type* C)**:**
  $d_i = (dx_i, dy_i, dz_i) \leftarrow (M(\mathcal{J}_i(x))M(\mathcal{J}_i(y)), M(\mathcal{J}_i(z)))$;
  $\mathcal{S} \leftarrow$ [<**bos**>, <C>];
  Match Set $\mathcal{M} \leftarrow \emptyset$; // Store the match bones
  for template $P \in \mathcal{T}$ **do**
    if $\mathcal{B}$ *match* P **then**
      // $\mathcal{B}$ match P: requires tree structure and name matching
      $\mathcal{S} \leftarrow [\mathcal{S},$ <**tempalte_token of** P>] ;
      $\mathcal{S} \leftarrow [\mathcal{S}, dx_{P_0}, dy_{P_0}, dz_{P_0}, \ldots, dx_{P_{|P|}}, dy_{P_{|P|}}, dz_{P_{|P|}}]$;
      $\mathcal{M} \leftarrow \{\mathcal{M}, P\}$
  for $R \in \mathcal{J}$ **do**
    if $R \notin \mathcal{M}$ ***and*** $\mathcal{P}_R \in \mathcal{M}$ **then**
      // check R is a root of remain forests
      stack.push(R);
      last_bone $\leftarrow$ None;
      while |stack| > 0 **do**
        bone $b \leftarrow$ stack.top(); // get bone index b
        stack.pop();
        if parent[b] $\neq$ last_bone **then**
          $\mathcal{S} \leftarrow [\mathcal{S},$ <**branch_token**>] ;
          $\mathcal{S} \leftarrow [\mathcal{S}, dx_{\mathcal{P}_b}, dy_{\mathcal{P}_b}, dz_{\mathcal{P}_b}]$;
        $\mathcal{S} \leftarrow [\mathcal{S}, dx_b, dy_b, dz_b]$;
        last_bone $\leftarrow b$;
        children[b] **sorted by** (z, y, x);
        stack.push(children[b]);
  $\mathcal{S} \leftarrow [\mathcal{S},$ <**eos**>];
  return $\mathcal{S}$;

Table 2. The average token costs in representing a skeleton tree of different datasets. Our optimized tokenization can reduce about 30% tokens.

Dataset \ Method	Naïve	Optimized	Tokens Reduction
VRoid	667.27	483.95	27.47 %
Rig-XL	266.28	187.15	29.72 %

shown in Table 2, we observe an average token reduction of 27.47% on VRoid and 29.72% on *Rig-XL* .

In addition to reducing the number of tokens required to represent the skeletal tree, our representation ensures that when generating based on a template, the generated fixed positions correspond precisely to the skeleton. By leveraging positional encoding and an autoregressive model, this tokenization approach enables higher accuracy in template-specified predictions. These lead to reduced memory consumption during training and faster inference, making our method more efficient.

6 SKIN WEIGHT PREDICTION VIA BONE-POINT CROSS ATTENTION

Having predicted the skeleton tree in Section 5, we now focus on predicting the skinning weights that govern mesh deformation. These weights determine the influence of each bone on each vertex of the mesh. Formally, we aim to predict a weight matrix $\mathcal{W} \in \mathbb{R}^{N\times J}$, where N is the number of vertices in the mesh and J is the number of bones. In our case, N can be in the tens of thousands due to the complexity of models in *Rig-XL* , and J can be in the hundreds. The high dimensionality of $\mathcal{W}$ poses a significant computational challenge.

Additionally, many applications require the prediction of bone-specific attributes, denoted by $\mathcal{A} \in \mathbb{R}^{J\times B}$, where B is the dimensionality of the attribute vector. These attributes can encode various physical properties, such as stiffness or gravity coefficients, which are crucial for realistic physical simulations (detailed in Section 6.2). Some bones might also act purely as connectors without influencing mesh deformation, as indicated by the "connected" option in Blender [Blender 2018].

To address these challenges, we propose a novel framework for skin weight and bone attribute prediction that leverages a bone-informed cross-attention mechanism [Vaswani 2017]. This approach allows us to efficiently model the complex relationships between the predicted skeleton and the input mesh.

Our framework utilizes two specialized encoders: a bone encoder E_B and a point-wise encoder E_P. The bone encoder, E_B, is a Multi-Layer Perceptron (MLP) with positional encoding that processes the head and tail coordinates of each bone, represented as $(\mathcal{J}_\mathcal{P}, \mathcal{J}) \in \mathbb{R}^{J\times 6}$. This yields bone features $\mathcal{F}_B \in \mathbb{R}^{J\times F}$, where F is the feature dimensionality.

For geometric feature extraction, we employ a pretrained Point Transformer V3 [Wu et al. 2024] as our point-wise encoder, E_P. Specifically, we use the architecture and weights from SAMPart3D [Yang et al. 2024], which was pretrained on a large dataset of 3D objects [Deitke et al. 2024]. SAMPart3D's removal of standard downsampling layers enhances its ability to capture fine-grained geometric details. The point-wise encoder takes the input point cloud, $\mathcal{X} \in \mathbb{R}^{N\times 3}$, and produces point-wise features $\mathcal{F}_P \in \mathbb{R}^{N\times F}$.

To predict skinning weights, we incorporate a cross-attention mechanism to model the interactions between bone features and point-wise features. We project the point-wise features $\mathcal{F}_P$ into query vectors $\mathcal{Q}_W$, and the bone features $\mathcal{F}_B$ to key and value vectors $\mathcal{K}_W$ and $\mathcal{V}_W$. The attention weights $\mathcal{F}_W \in \mathbb{R}^{N\times J\times H}$ are then computed as:

$$\mathcal{F}_W = \text{softmax}\left(\frac{\mathcal{Q}_W \mathcal{K}_W^T}{\sqrt{F}}\right),$$

where H is the number of attention heads. Each element $\mathcal{F}_W(i, j)$ represents the attention weight between the i-th vertex and the j-th bone, essentially capturing the influence of each bone on each vertex.

We further augment the attention weights by incorporating the voxel geodesic distance[Dionne and de Lasa 2013] $\mathcal{D} \in \mathbb{R}^{N\times J}$ between each vertex and each bone, following previous work [Xu et al. 2020, 2022]. This distance provides valuable information about the

spatial proximity of bones and vertices, which is crucial for accurate skin weight prediction. The geodesic distance $\mathcal{D}$ is precomputed and concatenated with the attention weights $\mathcal{F}_W$. Finally, the skinning weights $\mathcal{W}$ are obtained by passing the concatenated features through an MLP, E_W, followed by a softmax layer for normalization:

$$\mathcal{W} = \text{softmax}\left(E_W\left(\text{concat}\left(\text{softmax}\left(\frac{Q_W K_W^T}{\sqrt{F}}\right), \mathcal{D}\right)\right)\right).$$

For the prediction of bone attributes $\mathcal{A}$, we reverse the roles of bones and vertices in the cross-attention mechanism. Bone features $\mathcal{F}_B$ become the query, and point-wise features $\mathcal{F}_P$ are projected to key and value vectors. The bone attributes are then predicted using another MLP, E_A:

$$\mathcal{A} = E_A\left(\text{cross\_attn}\left(\mathcal{F}_B, \mathcal{F}_P\right)\right).$$

We use the Kullback-Leibler (KL) divergence [Van Erven and Harremos 2014] between the predicted and ground-truth skinning weights ($\mathcal{W}_{\text{pred}}$ and $\mathcal{W}$) and the L2 loss between the predicted and ground-truth bone attributes ($\mathcal{A}_{\text{pred}}$ and $\mathcal{A}$). The combined loss function is given by:

$$\lambda_{\mathcal{W}}\mathcal{L}_{\text{KL}}(\mathcal{W}, \mathcal{W}_{\text{pred}}) + \lambda_{\mathcal{A}}\mathcal{L}_2(\mathcal{A}, \mathcal{A}_{\text{pred}})$$

6.1 Training Strategy Based on Skeletal Equivalence

A naive approach to training would involve uniformly sampling points from the mesh surface. However, this leads to an imbalance in the training of different bones. Bones in densely sampled regions, such as the hip, tend to learn faster than those in sparsely sampled regions, such as hair or fingers. Additionally, using hierarchical point cloud sampling based on skinning weights can introduce discrepancies between the training and inference processes, ultimately hurting the model's performance during inference.

To address these issues, we propose a training strategy based on *skeletal equivalence*. Our key insight is that each bone should contribute equally to the overall training objective, regardless of the number of mesh vertices it influences. To achieve this, we introduce two key modifications to our training procedure. *First*, during each training iteration, we randomly freeze a subset of bones with a probability p. For these frozen bones, we use the ground-truth skinning weights and do not compute gradients. This ensures that all bones, even those in sparsely sampled regions, have an equal chance of being updated during training. *Second*, we introduce a bone-centric loss normalization scheme. Instead of averaging the loss over all vertices, we normalize the loss for each bone by the number of vertices it influences. This prevents bones that influence many vertices from dominating the loss function. Formally, our normalized loss function is given by:

$$\sum_{i=1}^{J}\frac{1}{J}\sum_{k=1}^{N}\frac{[\mathcal{W}_{k,i}>0]\mathcal{L}_2^{(k)}}{S_k=\sum_{k=1\ldots N}[\mathcal{W}_{k,i}>0]} = \frac{1}{J}\sum_{k=1}^{N}\mathcal{L}_2^{(k)}\left(\sum_{i=1}^{J}\frac{[\mathcal{W}_{k,i}>0]}{S_k}\right),$$

where S_k denotes the normalization factor based on the number of active points in each bone. It means we average the loss weight according to bone number instead of sample point number. where J is the number of bones, N is the number of vertices, and $[\mathcal{W}_{k,i} > 0]$ is an indicator function(iverson bracket) that is 1 if vertex i is influenced by bone j, and 0 otherwise. This can also be interpreted as first averaging the loss for each bone, and then averaging across all bones. $\mathcal{L}_2^{(k)}$ means the k-th vertex reconstruction loss of indirect supervision in Section 6.2. By incorporating these two techniques, our training strategy ensures that all bones are trained equally, leading to improved performance, especially for bones in sparsely sampled regions.

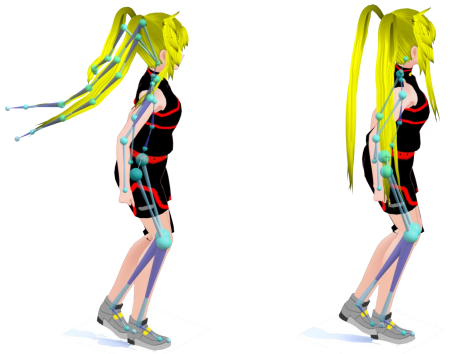

Fig. 6. **Comparison of model animation with and without spring bones.** The model on the left utilizes spring bones, resulting in more natural and dynamic movement of the hair and skirt. The model on the right does not use spring bones, leading to a stiffer and less realistic appearance, with only rigid body motion.

6.2 Indirect Supervision via Physical Simulation

While direct supervision using skinning weight loss can yield good results, it may not always guarantee visually realistic motion. This is because different combinations of skinning weights can produce similar deformations under simple transformations, even if one set of weights is physically implausible. To address this issue, we introduce an indirect supervision method that incorporates physical simulation to guide the learning process toward more realistic results. This method provides a more robust training signal by evaluating the quality of the predicted skinning weights and bone attributes based on the resulting motion.

Our approach extends beyond traditional Linear Blend Skinning (LBS) by incorporating a differentiable Verlet integration-based physical simulation, inspired by the spring bone dynamics in VRoid models [Isozaki et al. 2021]. This simulation allows us to model the behavior of bones under the influence of physical forces like gravity and stiffness, as defined by the predicted bone attributes. By comparing the simulated motion generated using the predicted parameters with that generated using the ground-truth parameters, we can obtain a more accurate measure of the prediction quality. Figure 6 illustrates the impact of spring bones on the realism of the animation.

In the VRM standard, spring motion is governed by several physical parameters, including drag coefficient η_d, stiffness coefficient η_s, gravity coefficient η_g, and gravity direction $\mathbf{g}$. For simplicity, we assume a uniform downward gravity direction and neglect collisions. Verlet integration is used to compute the bone's tail position at each time step, requiring both the current and previous frames' positions. To prevent numerical instability, the bone length is normalized after

each integration step. The details of the simulation are provided in Algorithm 2 in the supplementary material.

To incorporate this physical simulation into our training, we randomly sample a short motion sequence M from the Mixamo dataset of length T and apply it to both the predicted and ground-truth parameters. This results in two sets of simulated vertex positions: $\mathcal{X}^{\mathcal{M}}_{\text{pred}}$ (using predicted skinning weights $\mathcal{W}_{\text{pred}}$ and bone attributes $\mathcal{A}_{\text{pred}}\}$) and $\mathcal{X}^{\mathcal{M}}$ (using ground-truth $\mathcal{W}$ and $\mathcal{A}$). To ensure gradient stability, we use a short sequence length of $T = 3$, which is sufficient to capture the effects of the physical simulation.

We then use the L2 distance between the simulated vertex positions as a reconstruction loss, which serves as our indirect supervision signal. This loss, combined with the direct supervision losses from Section 6 forms our final loss function:

$$\lambda_{\mathcal{W}}\mathcal{L}_{\text{KL}}(\mathcal{W}, \mathcal{W}_{\text{pred}}) + \lambda_{\mathcal{A}}\mathcal{L}_2(\mathcal{A}, \mathcal{A}_{\text{pred}}) + \lambda_{\mathcal{X}} \sum_{i=1}^{T} \mathcal{L}_2(\mathcal{X}^{\mathcal{M}_i}, \mathcal{X}^{\mathcal{M}_i}_{\text{pred}}).$$

where $\lambda_{\mathcal{W}}$, $\lambda_{\mathcal{A}}$, and $\lambda_{\mathcal{X}}$ are weighting factors that balance the different loss terms. This combined loss function encourages the model to predict skinning weights and bone attributes that not only match the ground truth directly but also produce physically realistic motion.

7 EXPERIMENTS

7.1 Implementation Details

7.1.1 Dataset Preprocessing. As illustrated in Figure 3, the original *Rig-XL* dataset exhibits a highly skewed distribution, with human-related categories (Mixamo and Biped) being significantly overrepresented. Directly training on this unbalanced distribution would lead to suboptimal performance, particularly for underrepresented categories. To mitigate this issue and ensure a more balanced training set across diverse skeleton types, we adjusted the sampling probabilities for each category as follows: VRoid: 25%, Mixamo: 5%, Biped: 10%, Quadruped: 20%, Bird & Flyer: 15%, Static: 5%, and Insect & Arachnid: 10%. This distribution prioritizes high-quality data (VRoid) while ensuring sufficient representation of other categories.

To further enhance the robustness and generalizability of our model, we employed two key data augmentation techniques:

1. **Random Rotation & Scaling:** With a probability of $p_r = 0.4$, we randomly rotated the entire point cloud around each of the three coordinate axes by an Euler angle $r \in [-30°, 30°]$ (XYZ order). Independently, with a probability of $p_s = 0.5$, we scaled the point cloud by a factor $s \in [0.8, 1.0]$.
2. **Motion-Based Augmentation:** We applied motion sequences to the models to augment the training data with a wider range of poses. For models in the Mixamo and VRoid categories, we applied motion sequences from the Mixamo action database with a probability of $p_{m1} = 0.6$. For models in other categories, we randomly rotated individual bones with a probability of $p_{m2} = 0.4$, with rotation angles sampled from $r \in [-15°, 15°]$.

7.1.2 Training Strategy. Our training process consists of two stages: skeleton tree prediction and skin weight prediction. For *skeleton tree prediction* (Section 5), we employed the OPT-125M transformer [Zhang et al. 2022] as our autoregressive model, combined with a geometric encoder based on the 3DShape2Vecset framework [Zhang et al. 2023b; Zhao et al. 2024]. The model was trained for 3 days on 8 NVIDIA A100 GPUs, utilizing the AdamW optimizer [Loshchilov 2017] with parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a weight decay of 0.01. We trained for a total of 500 epochs with a cosine annealing learning rate schedule, starting at a learning rate of 1×10^{-3} and decreasing to 2×10^{-4}. For *skin weight prediction* (Section 6), we sampled 16,384 points from each mesh during training. We used a reduced model to save training resources, which includes a frozen pretrained Point Transformer from SAMPart3D [Yang et al. 2024] and only a small portion of parameters in the Bone Encoder, Cross Attention, and Weight Decoder modules are trainable. The learning rate was fixed at 1×10^{-3} during this stage. This phase of training required 1 day on 8 NVIDIA A100 GPUs.

7.2 Results and Comparison

To evaluate the effectiveness of our proposed method, we conducted a comprehensive comparison against both state-of-the-art academic methods and widely used commercial tools. Our evaluation focuses on two key aspects: *skeleton prediction accuracy* and *skinning quality*. For *quantitative evaluation* of skeleton prediction, we compared UniRig with several prominent open-source methods: RigNet [Xu et al. 2020], NBS [Li et al. 2021], and TA-Rig [Ma and Zhang 2023]. These methods represent the current state-of-the-art in data-driven rigging. We used a validation set consisting of 50 samples from the VRoid dataset and 100 samples from the *Rig-XL* dataset. The validation set and training dataset are guaranteed to never overlap after we deduplicate them carefully in Section 4.2. The validation samples in *Rig-XL* are selected uniformly from each class. The VRoid samples allowed us to assess the performance on detailed, anime-style characters, while the *Rig-XL* samples tested the generalizability of our method across diverse object categories. We also performed a *qualitative comparison* against several commercial and closed-source systems, including Meshy [Meshy 2024], Anything World [Anything-World 2024], and Accurig [Auto-Rig 2024]. Due to the closed-source nature of these systems, a direct quantitative comparison was not feasible. Instead, we compared the visual quality of the generated skeletons and the resulting mesh animations. The qualitative results are presented and discussed.

7.2.1 Bone Prediction. To evaluate the accuracy of our bone prediction, we used three metrics based on chamfer distance:

- **Joint-to-Joint Chamfer Distance (J2J):** Measures the average chamfer distance between corresponding predicted and ground-truth joint positions.
- **Joint-to-Bone Chamfer Distance (J2B):** Measures the average chamfer distance between predicted joint positions and their closest points on the ground-truth bone segments.
- **Bone-to-Bone Chamfer Distance (B2B):** Measures the average chamfer distance between points on the predicted bone segments and their closest points on the ground-truth bone segments.

Lower values for these metrics indicate better prediction accuracy. For a fair comparison with prior work on the Mixamo and VRoid datasets, we evaluated the metrics using a reduced set of 52 bones (or 22 bones). For the *Rig-XL* dataset, which contains more diverse skeletal structures, we used the complete set of predicted bones. All

Table 3. Quantitative comparison of Joint-to-Joint Chamfer Distance (J2J). * indicates the evaluation dataset is under the data augmentation of random rotation, scale, and applying random motion. † indicates the model cannot be finetuned because RigNet does not provide data preprocess tools and TA-Rig does not provide training scripts. The best results are **bold**

Method \ Dataset	Mixamo	VRoid	Mixamo*	VRoid*	*Rig-XL* *
Ours	**0.0101**	**0.0092**	**0.0103**	**0.0101**	**0.0549**
RigNet† [Xu et al. 2020]	0.1022	0.2405	0.2171	0.2484	0.2388
NBS [Li et al. 2021]	0.0338	0.0205	0.0429	0.0214	N/A
TA-Rig† [Ma and Zhang 2023]	0.1007	0.0886	0.1093	0.0934	0.2175

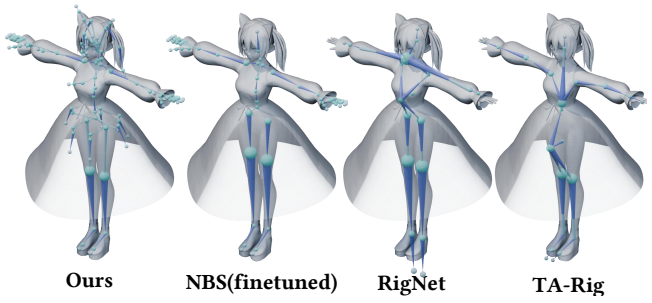

Fig. 7. Comparison of predicted skeletons between NBS (fine-tuned), RigNet, and TA-Rig on the VRoid dataset. Our method (UniRig) generates skeletons that are more detailed and accurate.

mesh models were normalized to a unit cube ($[-1, 1]^3$) to ensure consistent evaluation across datasets. All mesh models were normalized to a unit cube ($[-1, 1]^3$) to ensure consistent evaluation across datasets.

Table 3 presents the quantitative results for the J2J metric. Our method, UniRig, outperforms all other methods across all datasets, demonstrating its superior accuracy in predicting joint positions. Additional results for the J2B and B2B metrics are provided in Supplementary Table 9, further demonstrating the effectiveness of our approach.

Figure 7 provides a visual comparison of the predicted skeletons against RigNet, NBS, and TA-Rig on the VRoid dataset. The results show that UniRig generates more detailed and accurate skeletons. Further visual comparisons with academic methods are available in Supplementary Figure 13.

We also conducted a qualitative comparison against commercial tools, including Tripo [VAST 2025], Meshy [Meshy 2024], and Anything World [Anything-World 2024]. As illustrated in Figure 8, our method substantially outperforms these commercial systems, offering superior accuracy across a diverse range of mesh types, while also improving the completeness of the predicted skeletons.

7.2.2 Skinning Weight Prediction and Mesh Deformation Robustness. To evaluate the quality of our predicted skinning weights, we adopted a two-pronged approach: (1) *direct comparison of skinning weights* and (2) *evaluation of mesh deformation robustness under animation*. The former directly assesses the accuracy of the predicted weights, while the latter provides a more holistic measure of their ability to drive realistic animations.

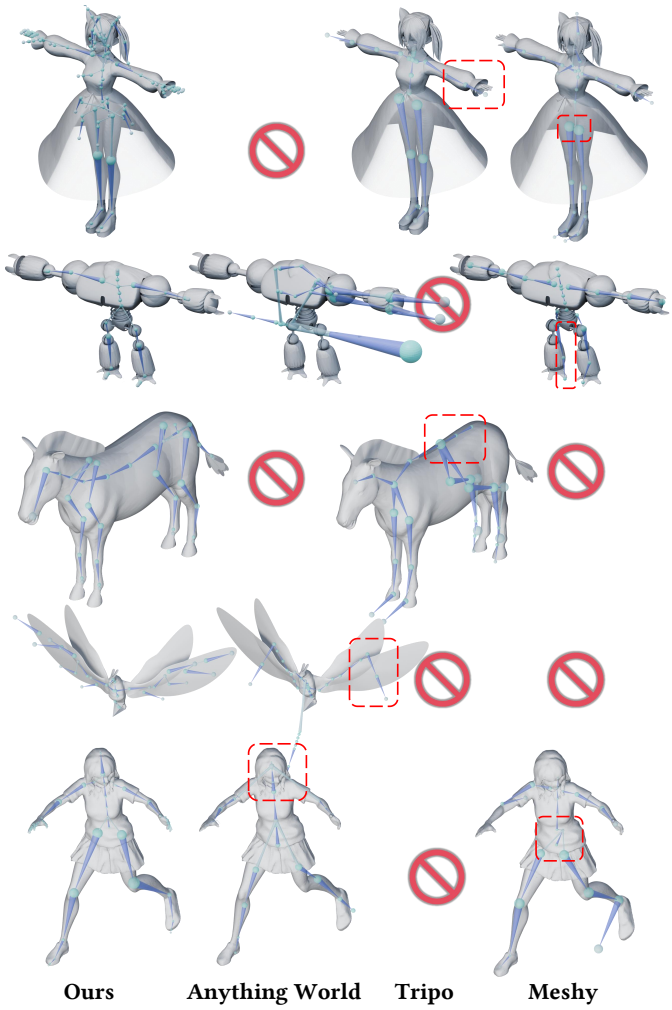

Fig. 8. **Qualitative comparison of predicted skeletons against commercial tools.** Our method (UniRig) outperforms Tripo [VAST 2025], Meshy [Meshy 2024], Anything World [Anything-World 2024], and Accurig [AutoRig 2024] in terms of both accuracy and detail. Red stop signs indicate that the corresponding tool failed to generate a skeleton.

Table 4. Comparison of skinning weight prediction accuracy using per-vertex L1 loss between predicted and ground-truth skinning weights. * means the evaluation dataset is under the data augmentation of random rotation, scale, and applying random motion. † indicates the model cannot be finetuned because RigNet does not provide data preprocess tools and TA-Rig does not provide training scripts.

Method \ Dataset	Mixamo	VRoid	Mixamo*	VRoid*	*Rig-XL* *
Ours	**0.0055**	**0.0028**	**0.0059**	**0.0038**	0.0329
RigNet† [Xu et al. 2020]	0.04540	0.04893	0.05367	0.06146	N/A
NBS[Li et al. 2021]	0.07898	0.02721	0.08211	0.03339	N/A

Table 5. Comparison of mesh deformation robustness using reconstruction loss under various animation sequences. * means the evaluation dataset is under the data augmentation of random rotation, scale, and applying random motion.

Method \ Dataset	Mixamo	VRoid	Mixamo*	VRoid*	VRoid with Spring*	*Rig-XL*
Ours	$\mathbf{4.00 \times 10^{-4}}$	$\mathbf{4.00 \times 10^{-4}}$	$\mathbf{6.00 \times 10^{-4}}$	$\mathbf{1.10 \times 10^{-3}}$	$\mathbf{1.70 \times 10^{-3}}$	3.5×10^{-3}
NBS [Li et al. 2021]	8.03×10^{-4}	5.82×10^{-2}	1.38×10^{-3}	2.34×10^{-3}	2.71×10^{-3}	N/A

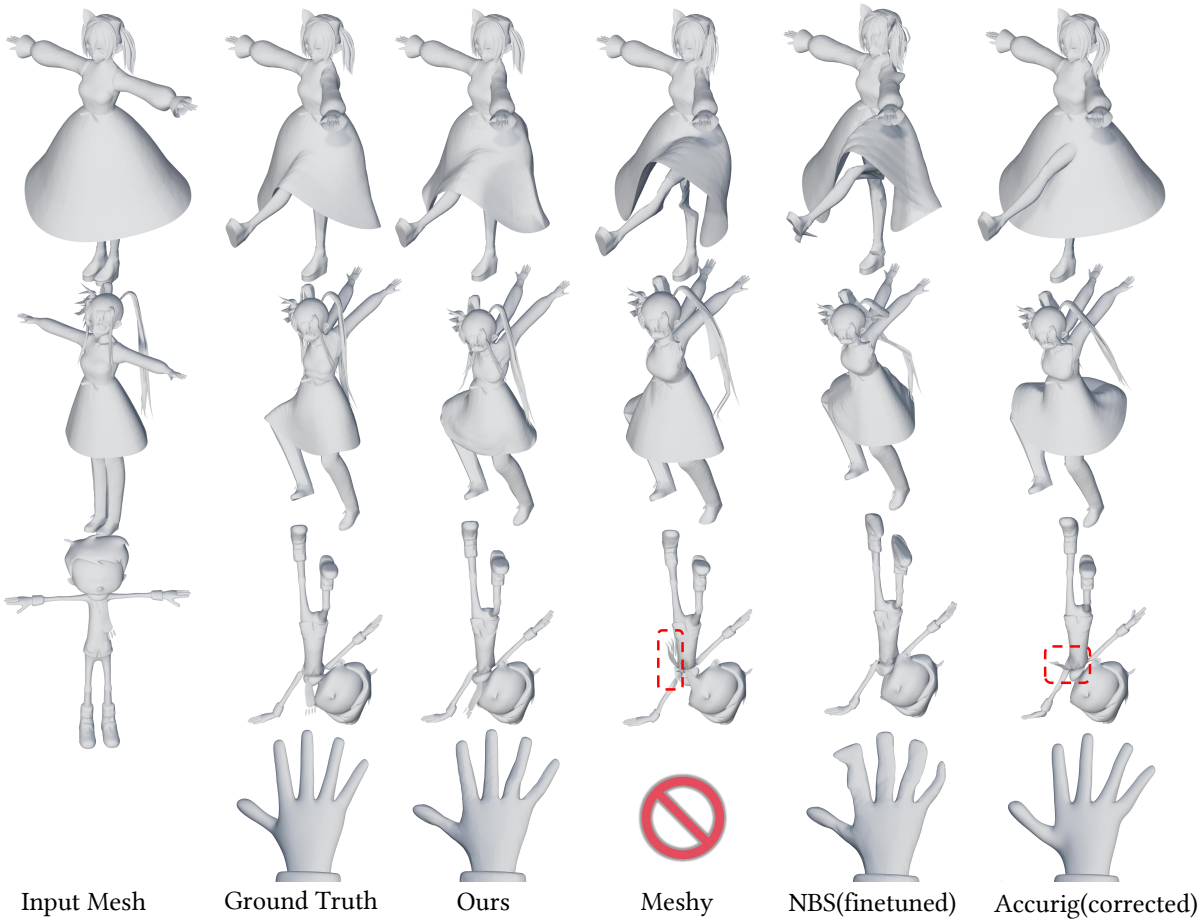

Fig. 9. **Qualitative comparison of mesh deformation under motion.** Our method (UniRig) is compared with commercial tools (Meshy [Meshy 2024] and Accurig [Auto-Rig 2024]) and a state-of-the-art academic method (NBS [Li et al. 2021]) on several models. Our model and the ground truth both exhibit realistic physical simulation of spring bones, resulting in more natural hair and clothing movement. Our method also demonstrates precise hand weight prediction, enabling fine-grained hand movements. Note that NBS was fine-tuned on the VRoid dataset, while Accurig requires joint manually corrected.

For the *direct comparison of skinning weights*, we computed the per-vertex L1 loss between the predicted and ground-truth skinning weights. We compared our method against RigNet [Xu et al. 2020], Neural Blend Shapes (NBS) [Li et al. 2021], and TA-Rig [Ma and Zhang 2023], all of which also predict skinning weights. As shown in Table 4, UniRig significantly outperforms these methods across all datasets, demonstrating the superior accuracy of our skin weight prediction.

As shown in Sections 7.2.1 and 7.2.2, our method demonstrates substantial advantages in both skeleton rigging and skinning weight prediction, while also facilitating an efficient retargeting process. Consequently, the deformed meshes driven by our predictions exhibit good robustness across various animated poses. To quantify and validate this, we applied a set of 2,446 diverse animation sequences from the Mixamo dataset to the rigged models (VRoid and Mixamo). For each animation sequence, we sampled one frame and computed the L2 reconstruction loss between the ground-truth mesh

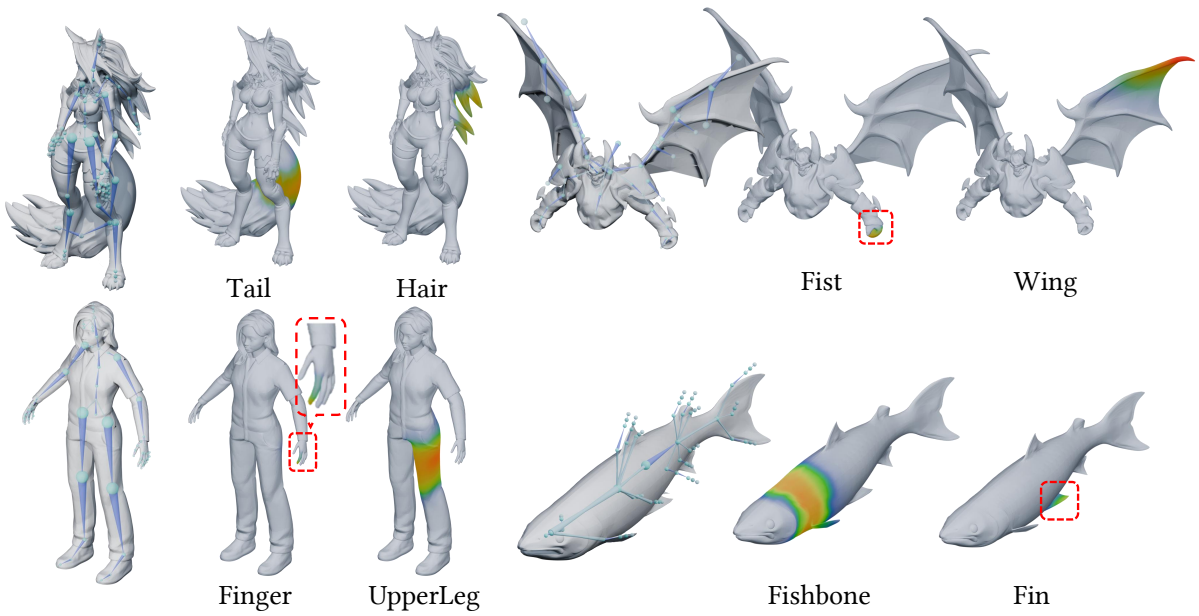

Fig. 10. **Qualitative results of *UniRig* on various object categories.** The figure showcases the predicted skeletons, skinning weights, and the resulting deformed meshes. Our method demonstrates the ability to predict highly detailed skeletal structures and accurate local skin weight mappings.

Table 6. **Comparison of different tokenization strategies.** The values for the naive method are shown on the left, while the values for our optimized method are shown on the right. ★ Inference time is tested on an RTX 4090 GPU. † indicates that the models were trained for only 160 epochs for this ablation study, to control for variables, so the results are not as good as full training.

Metrics \ Dataset	Mixamo*	VRoid*	*Rig-XL* *
Average Tokens	369.53 | **214.89**	621.76 | **522.88**	495.46 | **237.94**
Inference Time(s)★	3.57 | **2.16**	5.39 | **4.53**	4.29 | **1.99**
J2J Distance†	0.1761 | **0.0838**	0.1484 | **0.1374**	0.1395 | **0.1266**
J2B Distance†	0.1640 | **0.0779**	0.1287 | **0.0891**	0.1258 | **0.1017**
B2B Distance†	0.1519 | **0.0715**	0.1132 | **0.0766**	0.1099 | **0.0966**

and the mesh deformed using the predicted skeleton and skinning weights. This metric quantifies the ability of our method to produce realistic deformations across a wide range of motions.

Table 5 shows the reconstruction loss for UniRig and NBS. Our method achieves significantly lower reconstruction losses across all datasets, indicating its superior ability to generate robust and accurate mesh deformations. Notably, the results on "VRoid with Spring*" demonstrate the effectiveness of our method in handling dynamic simulations driven by spring bones.

Figure 9 provides a qualitative comparison of mesh deformation under motion against commercial tools (Meshy and Accurig) and NBS. The results demonstrate that our method produces more realistic deformations, particularly in areas with complex motion, such as the hair and hands. Figure 10 showcases the predicted skeletons, skinning weights, and resulting mesh deformations for various object types, further demonstrating the effectiveness of our approach.

7.3 Ablation Study

To validate the effectiveness of key components of our method, we conducted a series of ablation studies. Specifically, we investigated the impact of (1) our proposed tokenization strategy, (2) the use of indirect supervision via physical simulation, and (3) the training strategy based on skeletal equivalence.

7.3.1 Tokenize Strategy. In this comparative experiment, we assessed the performance of the naive tokenization method, as outlined in Section 5, against our optimized approach. We evaluated both methods based on the following metrics: average token sequence length, inference time, and bone prediction accuracy (measured by J2J distances). For a fair comparison, both models were trained for 160 epochs. Table 6 shows the results of this comparison. Our optimized tokenization strategy significantly reduces the average token sequence length, leading to a decrease in inference time. Notably, it also improves bone prediction accuracy across all datasets, demonstrating the effectiveness of our approach in capturing skeletal structure. The inference time is tested on a single RTX 4090 GPU.

7.3.2 Indirect Supervision based on Physical Simulation. To evaluate the impact of indirect supervision using physical simulation (Section 6.2), we compared the performance of our model with and without this component during training. We focused on the VRoid dataset for this experiment, as it contains spring bones that are directly affected by the physical simulation. Table 7 shows that training with indirect supervision leads to a significant improvement in both deformation error (L2 loss) and skinning weight error (L1 loss). This demonstrates that incorporating physical simulation into

Table 7. Ablation study on the use of indirect supervision via physical simulation. Deformation error is tested using the L2 loss under the same motion, while skinning error is evaluated using the L1 loss of per-vertex skinning weights.

Method \ Metrics	Deformation Error	Skin Error
UniRig	$\mathbf{7.74 \times 10^{-4}}$	$\mathbf{5.42 \times 10^{-3}}$
w/o Physical Simulation	8.59×10^{-4}	5.78×10^{-3}

Table 8. Ablation study on the training strategy based on skeletal equivalence. ⋆ indicates that the evaluation dataset is under the data augmentation of random rotation, scale, and applying random motion.

Metrics \ Dataset	Mixamo*	VRoid*	*Rig-XL* *
UniRig	$\mathbf{4.42 \times 10^{-4}}$	1.28×10^{-3}	$\mathbf{3.72 \times 10^{-3}}$
w/o skeleton frozen	4.92×10^{-4}	$\mathbf{1.25 \times 10^{-3}}$	3.84×10^{-3}
w/o bone loss normalization	4.63×10^{-4}	1.33×10^{-3}	3.92×10^{-3}

the training process helps the model learn more realistic skinning weights and bone attributes.

7.3.3 Training Strategy Based on Skeletal Equivalence. To validate the effectiveness of our training strategy based on skeletal equivalence (Section 6), we compared the performance of our model with and without this strategy. Specifically, we evaluated the impact of two key components: (1) randomly freezing bones during training and (2) normalizing the loss by the number of influenced vertices for each bone. Table 8 shows the results of this comparison. Using the full skeletal equivalence strategy (*UniRig*) yields the best performance in terms of reconstruction loss. Disabling either component ("w/o skeleton frozen" or "w/o bone loss normalization") leads to a degradation in performance, highlighting the importance of both aspects of our training strategy in achieving optimal results.

8 APPLICATIONS

8.1 Human-Assisted Auto-rigging

Compared to prior automatic rigging techniques, a key advantage of our approach lies in its ability to facilitate human-machine interaction. This is achieved through the ability to edit the predicted skeleton tree and trigger subsequent regeneration of the affected parts. As shown in Figure 11, users can perform operations such as adding new bone branches or removing existing ones (e.g., removing spring bones to achieve a more rigid structure). This allows for efficient correction of any inaccuracies in the automatic prediction and customization of the rig to specific needs. For instance, a user might add a new branch to represent a tail that was not automatically detected, or they might remove automatically generated spring bones that are not desired for a particular animation. The edited skeleton tree can then be fed back into the *UniRig* pipeline, generating an updated rig that incorporates the user's modifications. This iterative process empowers users to quickly and easily refine the automatically generated rigs, combining the speed of automation with the precision of manual control.

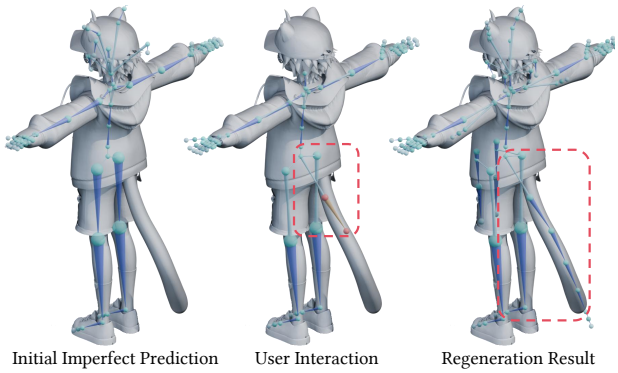

Fig. 11. **Human-assisted skeleton editing and regeneration with *UniRig* .** In this example, the initial prediction lacks a tail and has unsatisfactory spring bones. The user removes the spring bones, keeps the Mixamo template skeleton, and adds a prompt for a tail bone. *UniRig* then regenerates the skeleton based on these modifications, resulting in a more accurate and desirable rig.

Fig. 12. **VTuber live streaming with a UniRig-generated model.** The character, rigged using our method, exhibits smooth and realistic spring bone motion during live streaming in Warudo [Tang and Thompson 2024].

8.2 Character Animation

UniRig's ability to predict spring bone parameters, trained on the VRoid and Rig-XL dataset, makes it particularly well-suited for creating animated characters. Our method can generate VRM-compatible models from simple mesh inputs, enabling users to easily export their creations to various animation platforms. This streamlines the process of creating and animating virtual characters. For example, users can leverage tools like Warudo [Tang and Thompson 2024] to bring their rigged characters to life in a virtual environment, as demonstrated in Figure 12. This capability is especially valuable for applications like VTubing, where realistic and expressive character motion is highly desirable. The smooth and natural movements generated by our spring bone simulation contribute to a more engaging and immersive VTubing experience.

9 CONCLUSIONS

This paper presents *UniRig* , a unified learning-based framework for automatic rigging of 3D models. Our model, combined with a novel tokenization strategy and a two-stage training process, achieves state-of-the-art results in skeleton prediction and skinning weight prediction. The large-scale and diverse *Rig-XL* dataset, along with the curated VRoid dataset, enables training a generalizable model that can handle a wide variety of object categories and skeletal structures.

Limitations and Discussions. Despite its strengths, *UniRig* has certain limitations. Like other learning-based approaches, the performance of our method is inherently tied to the quality and diversity of the training data. While *Rig-XL* is a large and diverse dataset, it may not fully encompass the vast range of possible skeletal structures and object categories. Consequently, *UniRig* might perform suboptimally when presented with objects that significantly deviate from those in the training data. For instance, it might struggle with highly unusual skeletal structures, such as those found in abstract or highly stylized characters. As mentioned in Section 8.1, user edits can be used as a valuable source of data for further refining the model. By incorporating user feedback and expanding the training dataset, we can continuously improve the robustness and generalizability of *UniRig* . There are several avenues for future work. One direction is to explore the use of different modalities, such as images or videos, as input to the rigging process. Furthermore, incorporating more sophisticated physical simulation techniques could enhance the realism of the generated animations.

In conclusion, *UniRig* represents a step towards fully automated and generalizable rigging. Its ability to handle diverse object categories, coupled with its support for human-in-the-loop editing and realistic animation, makes it a powerful tool for both researchers and practitioners in the field of 3D computer graphics.

REFERENCES

Noam Aigerman, Kunal Gupta, Vladimir G Kim, Siddhartha Chaudhuri, Jun Saito, and Thibault Groueix. 2022. Neural jacobian fields: Learning intrinsic mappings of arbitrary meshes. *arXiv preprint arXiv:2205.02904* (2022).

Nina Amenta and Marshall Bern. 1998. Surface reconstruction by Voronoi filtering. In *Proceedings of the fourteenth annual symposium on Computational geometry*. 39–48.

Anything-World. 2024. *Animation and automated rigging.* https://www.anythingworld.com.

Auto-Rig. 2024. *Free Auto Rig for any 3D Character | AccuRIG.* https://actorcore.reallusion.com/accurig.

Ilya Baran and Jovan Popović. 2007. Automatic rigging and animation of 3d characters. *ACM Transactions on graphics (TOG)* 26, 3 (2007), 72–es.

Sue Blackman. 2014. Rigging with mixamo. *Unity for Absolute Beginners* (2014), 565–573.

Blender. 2018. *Blender - a 3D modelling and rendering package.* Blender Foundation, Stichting Blender Foundation, Amsterdam. http://www.blender.org

Yiwen Chen, Tong He, Di Huang, Weicai Ye, Sijin Chen, Jiaxiang Tang, Xin Chen, Zhongang Cai, Lei Yang, Gang Yu, et al. 2024. MeshAnything: Artist-Created Mesh Generation with Autoregressive Transformers. *arXiv preprint arXiv:2406.10163* (2024).

Zedong Chu, Feng Xiong, Meiduo Liu, Jinzhi Zhang, Mingqi Shao, Zhaoxu Sun, Di Wang, and Mu Xu. 2024. HumanRig: Learning Automatic Rigging for Humanoid Character in a Large Scale Dataset. *arXiv preprint arXiv:2412.02317* (2024).

Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. 2024. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems* 36 (2024).

Olivier Dionne and Martin de Lasa. 2013. Geodesic voxel binding for production character meshes. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 173–180.

Hany Farid. 2021. An overview of perceptual hashing. *Journal of Online Trust and Safety* 1, 1 (2021).

Lin Gao, Jie Yang, Yi-Ling Qiao, Yu-Kun Lai, Paul L Rosin, Weiwei Xu, and Shihong Xia. 2018. Automatic unpaired shape deformation transfer. *ACM Transactions on Graphics (ToG)* 37, 6 (2018), 1–15.

Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. 2018. 3d-coded: 3d correspondences by deep deformation. In *Proceedings of the european conference on computer vision (ECCV)*. 230–246.

Zekun Hao, David W Romero, Tsung-Yi Lin, and Ming-Yu Liu. 2024. Meshtron: High-Fidelity, Artist-Like 3D Mesh Generation at Scale. *arXiv preprint arXiv:2412.09548* (2024).

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

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276* (2024).

Nozomi Isozaki, Shigeyoshi Ishima, Yusuke Yamada, Yutaka Obuchi, Rika Sato, and Norio Shimizu. 2021. VRoid studio: a tool for making anime-like 3D characters using your imagination. In *SIGGRAPH Asia 2021 Real-Time Live!* 1–1.

Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. 2007. Skinning with dual quaternions. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*. 39–46.

Peizhuo Li, Kfir Aberman, Rana Hanocka, Libin Liu, Olga Sorkine-Hornung, and Baoquan Chen. 2021. Learning skeletal articulations with neural blend shapes. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–15.

Hanwen Liang, Yuyang Yin, Dejia Xu, Hanxue Liang, Zhangyang Wang, Konstantinos N Plataniotis, Yao Zhao, and Yunchao Wei. 2024. Diffusion4D: Fast Spatial-temporal Consistent 4D Generation via Video Diffusion Models. *arXiv preprint arXiv:2405.16645* (2024).

Zhouyingcheng Liao, Jimei Yang, Jun Saito, Gerard Pons-Moll, and Yang Zhou. 2022. Skeleton-free pose transfer for stylized 3d characters. In *European Conference on Computer Vision*. Springer, 640–656.

Lijuan Liu, Youyi Zheng, Di Tang, Yi Yuan, Changjie Fan, and Kun Zhou. 2019. Neuroskinning: Automatic skin binding for production characters with deep graph networks. *ACM Transactions on Graphics (ToG)* 38, 4 (2019), 1–12.

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. 2023. SMPL: A skinned multi-person linear model. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*. 851–866.

I Loshchilov. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101* (2017).

Jing Ma and Dongliang Zhang. 2023. TARig: Adaptive template-aware neural rigging for humanoid characters. *Computers & Graphics* 114 (2023), 158–167.

David Marr and Herbert Keith Nishihara. 1978. Representation and recognition of the spatial organization of three-dimensional shapes. *Proceedings of the Royal Society of London. Series B. Biological Sciences* 200, 1140 (1978), 269–294.

Meshy. 2024. *Meshy - convert text and images to 3D models.* https://www.meshy.com.

Models-Resource. 2019. The Models-Resource.

Blue Nile. 2025. *Lazy Bones.* https://blendermarket.com/products/lazy-bones.

Hao-Yang Peng, Jia-Peng Zhang, Meng-Hao Guo, Yan-Pei Cao, and Shi-Min Hu. 2024. CharacterGen: Efficient 3D Character Generation from Single Images with Multi-View Pose Canonicalization. *ACM Transactions on Graphics (TOG)* 43, 4 (2024). https://doi.org/10.1145/3658217

Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. 2022. Dreamfusion: Text-to-3d using 2d diffusion. *arXiv preprint arXiv:2209.14988* (2022).

Yawar Siddiqui, Antonio Alliegro, Alexey Artemov, Tatiana Tommasi, Daniele Sirigatti, Vladislav Rosov, Angela Dai, and Matthias Nießner. 2024. Meshgpt: Generating triangle meshes with decoder-only transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 19615–19625.

Mingze Sun, Junhao Chen, Junting Dong, Yurun Chen, Xinyu Jiang, Shiwei Mao, Puhua Jiang, Jingbo Wang, Bo Dai, and Ruqi Huang. 2024. DRiVE: Diffusion-based Rigging Empowers Generation of Versatile and Expressive Characters. *arXiv preprint arXiv:2411.17423* (2024).

Andrea Tagliasacchi, Hao Zhang, and Daniel Cohen-Or. 2009. Curve skeleton extraction from incomplete point cloud. In *ACM SIGGRAPH 2009 papers*. 1–9.

Man To Tang and Jesse Thompson. 2024. Warudo: Interactive and Accessible Live Performance Capture. In *ACM SIGGRAPH 2024 Real-Time Live!* 1–2.

Tim Van Erven and Peter Harremos. 2014. Rényi divergence and Kullback-Leibler divergence. *IEEE Transactions on Information Theory* 60, 7 (2014), 3797–3820.

VAST. 2025. *Tripo AI.* https://www.tripoai.com.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems* (2017).

Haoyu Wang, Shaoli Huang, Fang Zhao, Chun Yuan, and Ying Shan. 2023a. Hmc: Hierarchical mesh coarsening for skeleton-free motion retargeting. *arXiv preprint arXiv:2303.10941* (2023).

Jiashun Wang, Xueting Li, Sifei Liu, Shalini De Mello, Orazio Gallo, Xiaolong Wang, and Jan Kautz. 2023b. Zero-shot pose transfer for unrigged stylized 3d characters. In

Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 8704–8714.

Jiashun Wang, Chao Wen, Yanwei Fu, Haitao Lin, Tianyun Zou, Xiangyang Xue, and Yinda Zhang. 2020. Neural pose transfer by spatially adaptive instance normalization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 5831–5839.

Rong Wang, Wei Mao, Changsheng Lu, and Hongdong Li. 2025. Towards High-Quality 3D Motion Transfer with Realistic Apparel Animation. In *European Conference on Computer Vision*. Springer, 35–51.

Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. 2024. Point Transformer V3: Simpler Faster Stronger. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4840–4851.

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

Zhan Xu, Yang Zhou, Evangelos Kalogerakis, and Karan Singh. 2019. Predicting animation skeletons for 3d articulated models via volumetric nets. In *2019 international conference on 3D vision (3DV)*. IEEE, 298–307.

Zhan Xu, Yang Zhou, Li Yi, and Evangelos Kalogerakis. 2022. Morig: Motion-aware rigging of character meshes from point clouds. In *SIGGRAPH Asia 2022 conference papers*. 1–9.

Yajie Yan, David Letscher, and Tao Ju. 2018. Voxel cores: Efficient, robust, and provably good approximation of 3d medial axes. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–13.

Yajie Yan, Kyle Sykes, Erin Chambers, David Letscher, and Tao Ju. 2016. Erosion thickness on medial axes of 3D shapes. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–12.

Yunhan Yang, Yukun Huang, Yuan-Chen Guo, Liangjun Lu, Xiaoyang Wu, Edmund Y Lam, Yan-Pei Cao, and Xihui Liu. 2024. Sampart3d: Segment any part in 3d objects. *arXiv preprint arXiv:2411.07184* (2024).

Xin Yu, Ze Yuan, Yuan-Chen Guo, Ying-Tian Liu, Jianhui Liu, Yangguang Li, Yan-Pei Cao, Ding Liang, and Xiaojuan Qi. 2024. Texgen: a generative diffusion model for mesh textures. *ACM Transactions on Graphics (TOG)* 43, 6 (2024), 1–14.

Zhenbo Yu, Junjie Wang, Hang Wang, Zhiyuan Zhang, Jinxian Liu, Zefan Li, Bingbing Ni, and Wenjun Zhang. 2025. Mesh2Animation: Unsupervised Animating for Quadruped 3D Objects. *IEEE Transactions on Circuits and Systems for Video Technology* (2025).

Biao Zhang, Jiapeng Tang, Matthias Niessner, and Peter Wonka. 2023b. 3dshape2vecset: A 3d shape representation for neural fields and generative diffusion models. *ACM Transactions on Graphics (TOG)* 42, 4 (2023), 1–16.

Jiaxu Zhang, Shaoli Huang, Zhigang Tu, Xin Chen, Xiaohang Zhan, Gang Yu, and Ying Shan. 2023a. TapMo: Shape-aware Motion Generation of Skeleton-free Characters. *arXiv preprint arXiv:2310.12678* (2023).

Jia-Qi Zhang, Miao Wang, Fu-Cheng Zhang, and Fang-Lue Zhang. 2024a. Skinned Motion Retargeting with Preservation of Body Part Relationships. *IEEE Transactions on Visualization and Computer Graphics* (2024).

Longwen Zhang, Ziyu Wang, Qixuan Zhang, Qiwei Qiu, Anqi Pang, Haoran Jiang, Wei Yang, Lan Xu, and Jingyi Yu. 2024b. CLAY: A Controllable Large-scale Generative Model for Creating High-quality 3D Assets. *ACM Transactions on Graphics (TOG)* 43, 4 (2024), 1–20.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068* (2022).

Zibo Zhao, Wen Liu, Xin Chen, Xianfang Zeng, Rui Wang, Pei Cheng, Bin Fu, Tao Chen, Gang Yu, and Shenghua Gao. 2024. Michelangelo: Conditional 3d shape generation based on shape-image-text aligned latent representation. *Advances in Neural Information Processing Systems* 36 (2024).

ALGORITHM 2: Verlet Integration for Bone Position Update

Input: T_{current}: Bone tail of current frame, T_{prev}: Bone tail of previous frame, L_{bone}: Bone length, η_d: Drag coefficient, η_s: Stiffness coefficient, η_g: Gravity coefficient, g: Gravity direction, Δt: Time step.

Output: T_{next}: Updated bone tail position of the next frame.

Function `UpdatePosition`($T_{current}, T_{prev}, L_{bone}, \eta_d, \eta_s, \eta_g, g, \Delta t$):

- $\mathbf{I} \leftarrow (T_{\text{current}} - T_{\text{prev}}) \cdot (1 - \eta_d)$; `// Calculate interia`
- $\mathbf{S} \leftarrow \eta_s R_{\text{head}}^{-1} R_{\text{tail}}$; `// Calculate stiffness, R is the rotation matrix under world coordinate system`
- $\mathbf{G} \leftarrow \eta_g \cdot \mathbf{g}$; `// Calculate gravity`
- $\Delta x \leftarrow (\mathbf{I} + \mathbf{S} + \mathbf{G}) \cdot \Delta t$; `// Calculate displacement of the bone tail under three forces`
- $T_{\text{next}} \leftarrow H_{\text{next}} + L_{\text{bone}} \frac{\Delta x}{|\Delta x|}$ `// Update next tail position under length normalization`
- **return** T_{next};

A APPENDIX

A.1 Datasets

A.1.1 Rig-XL Data Process.

Fix the problem of lacking a reasonable topological relationship. When processing Objaverse, we found that many animators do not rig a reasonable topology, because sometimes they directly use keyframe animation to adjust the bones individually to create the animation. This situation can be filtered by a simple rule: if the out-degree of the root node is greater than 4, and the subtree size of the root node's **heavy child** exceeds half the size of the skeleton Tree, the vast majority of such data can be filtered out. To address this issue, we cut off all outgoing edges of the root node, treat the **heavy child** as the new root, and then connect the remaining forest using a minimum spanning tree(MST) based on Euclidean distance.

A.2 More filter rules about the *Rig-XL*

A.2.1 Capture outlier through reconstruction loss. In the blend skinning weight training in Section 6, we found that although many data points were filtered, there were still a few outliers in the reconstruction loss. This is actually because there were still some non-compliant data that were not cleared during the Objaverse data preprocessing. Therefore, we used the current average reconstruction loss multiplied by 10 as a threshold and filtered out the incorrectly preprocessed data during multiple epochs of training, removing it from the dataset. In addition, we removed samples where the skinning weights of some points were completely lost, because softmax is applied on each point, which makes it impossible to fit situations where all weights of the point are zero.

A.3 Methods

A.3.1 Physical Simulation on VRM. When deforming the VRM body, it first calculates the basic motion of the body using the forward kinematics method (i.e., the standard Mixamo template). Then, for each spring bone, the Verlet integration is applied sequentially from top to bottom along the chain to compute the position of each spring bone, resulting in a coherent animation effect. Whole process is shown in Algorithm 2.

We show more visualization results for detailed comparison. In Figure 13, we compare *UniRig* with NBS and RigNet on different types of examples for automatic rigging, which can be observed that it can predict highly accurate and detailed results even for non-standard poses and various complex meshes. Figure 14 demonstrates the precision of *UniRig* in predicting skinning weights such as hair better than previous work. Finally, Figure 15 showcases the high-precision skeleton rigging and excellent weight generated achieved by *UniRig* on more complex examples, such as ants.

A.4 More Results

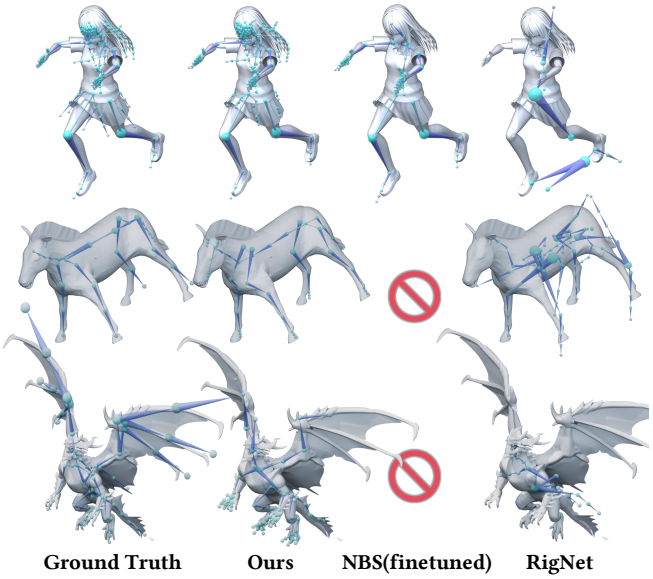

Fig. 13. We compare auto-rigging skeleton with NBS(finetuned) and RigNet on different kinds of 3D models.

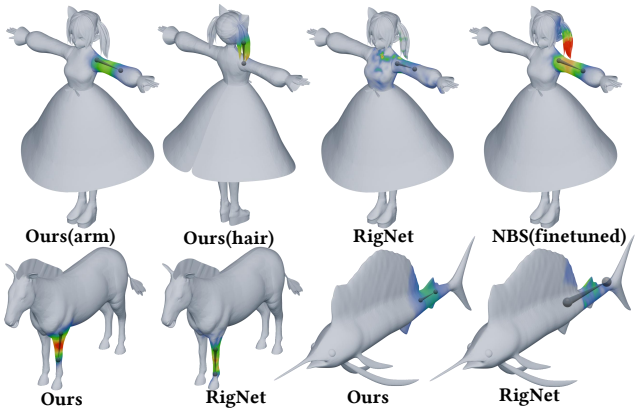

Fig. 14. We compare blend skinning weight with NBS(finetuned) and RigNet on different kinds of 3D models.

Table 9. Joint to bone (J2B) and Bone to bone (B2B) Chamfer distance. Left is CD-J2B, and right is CD-B2B. * means the evaluation dataset is under the data augmentation of random rotation, scale and applying random motion. † means we cannot finetune the model because RigNet do not provide data preprocess tools and TA-Rig do not provide training scripts.

Method \ Dataset	Mixamo	VRoid	Mixamo*	VRoid*	*Rig-XL* *
Ours	**0.0077** | **0.0044**	**0.0076** | **0.0043**	**0.0075** | **0.0040**	**0.0085** | **0.0046**	**0.0456** | **0.0276**
RigNet† [Xu et al. 2020]	0.0470 | 0.0398	0.1992 | 0.1793	0.1719 | 0.1534	0.2082 | 0.1833	0.1847 | 0.1519
Neural Blend-Shape[Li et al. 2021]	0.0277 | 0.0181	0.0158 | 0.0108	0.0349 | 0.0232	0.0168 | 0.0113	N/A
TA-Rig† [Ma and Zhang 2023]	0.0937 | 0.0775	0.0832 | 0.0682	0.1027 | 0.0860	0.0884 | 0.0726	0.1892 | 0.1465

Table 10. Quantitative comparison of skeleton prediction on Model Resources-RigNet[Models-Resource 2019; Xu et al. 2020].

Method \ Metrics	CD-J2J	CD-J2B	CD-B2B	Skin L1	Motion L2
Ours	**0.0332**	0.0266	**0.0194**	**0.0455**	**0.0019**
RigNet†[Xu et al. 2020]	0.039	**0.024**	0.022	0.39	N/A
Anything World	0.0540	0.0528	0.0338	N/A	N/A

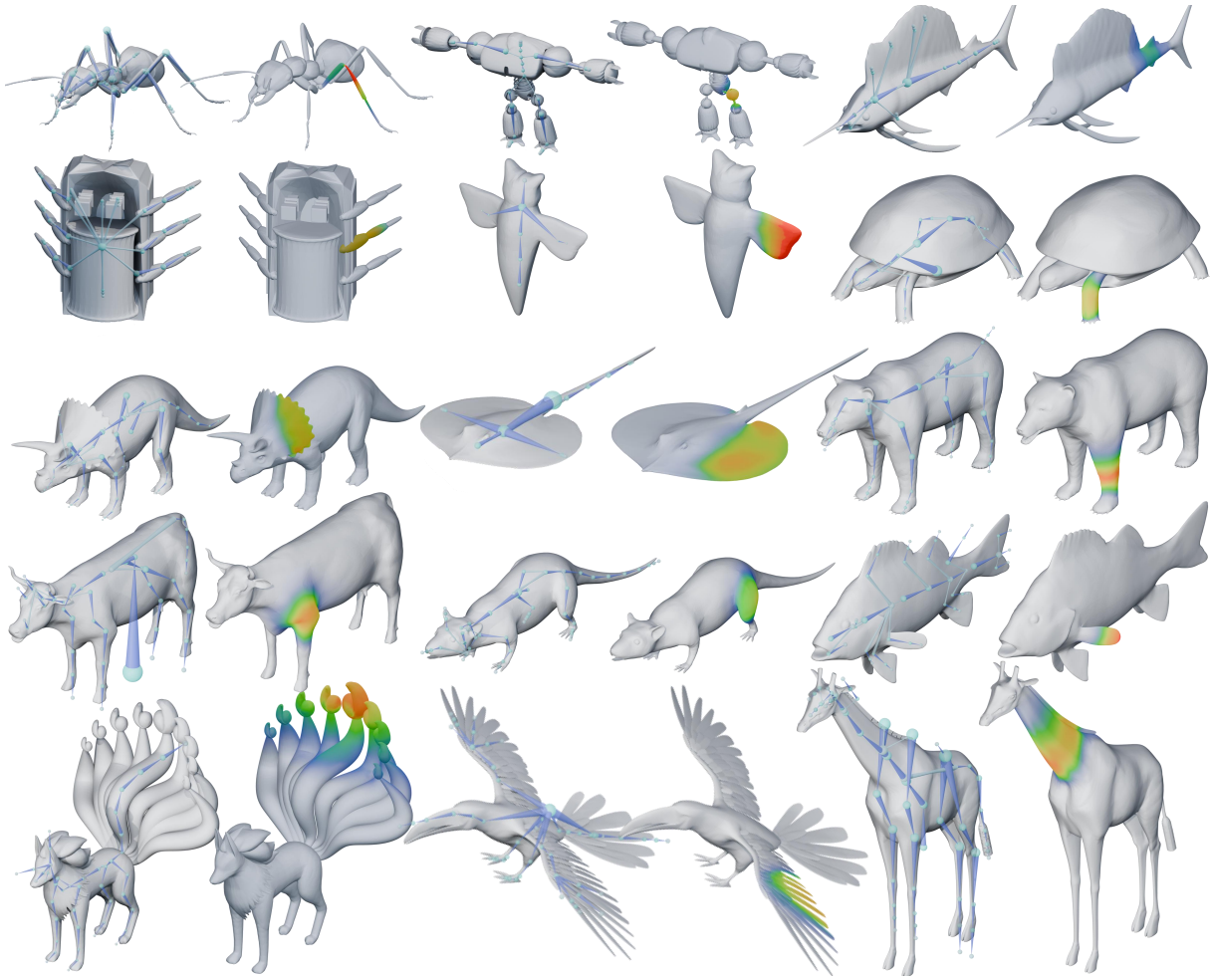

Fig. 15. We present more examples of *UniRig* here, demonstrating highly detailed and accurate skeleton rigging and weight generation.