Anymate: A Dataset and Baselines for Learning 3D Object Rigging

YUFAN DENG*, Stanford University, USA
YUHAO ZHANG*, Stanford University, USA
CHEN GENG, Stanford University, USA
SHANGZHE WU†, Stanford University, USA and University of Cambridge, UK
JIAJUN WU†, Stanford University, USA

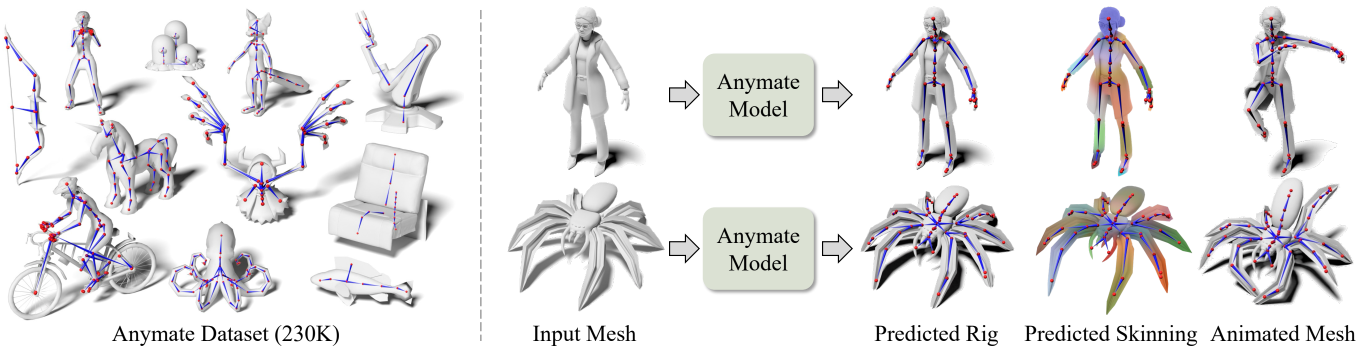


Fig. 1. We present the **Anymate Dataset** and a learning-based framework for automatic 3D object rigging. This dataset comprises 230K 3D assets with expert-crafted rigging and skinning information. Using this dataset, our framework learns to predict bone skeletons and skinning weights automatically from a given 3D mesh, allowing users to create realistic animations by manipulating the predicted skeleton.

Rigging and skinning are essential steps to create realistic 3D animations, often requiring significant expertise and manual effort. Traditional attempts at automating these processes rely heavily on geometric heuristics and often struggle with objects of complex geometry. Recent data-driven approaches show potential for better generality, but are often constrained by limited training data. We present the **Anymate Dataset**, a large-scale dataset of 230K 3D assets paired with expert-crafted rigging and skinning information—70 times larger than existing datasets. Using this dataset, we propose a learning-based auto-rigging framework with three sequential modules for joint, connectivity, and skinning weight prediction. We systematically design and experiment with various architectures as baselines for each module and conduct comprehensive evaluations on our dataset to compare their performance. Our models significantly outperform existing methods, providing a foundation for comparing future methods in automated rigging and skinning. Code and dataset can be found at https://anymate3d.github.io/.

CCS Concepts: • **Computing methodologies** → **Animation**.

Additional Key Words and Phrases: Data Driven Animation, Auto Rigging

*Equal contribution. The order of authorship was determined alphabetically. Work was done when Y. Deng and Y. Zhang were visiting students at Stanford University; they are currently with the Hong Kong University of Science and Technology.
†Equal advising.

Authors' addresses: Yufan Deng, Stanford University, Stanford, USA; Yuhao Zhang, Stanford University, Stanford, USA; Chen Geng, Stanford University, Stanford, USA; Shangzhe Wu, Stanford University, Stanford, USA and University of Cambridge, Cambridge, UK; Jiajun Wu, Stanford University, Stanford, USA.

SIGGRAPH Conference Papers '25, August 10–14, 2025, Vancouver, BC, Canada

ACM ISBN 979-8-4007-1540-2/2025/08
https://doi.org/10.1145/3721238.3730743

ACM Reference Format:
Yufan Deng, Yuhao Zhang, Chen Geng, Shangzhe Wu, and Jiajun Wu. 2025. Anymate: A Dataset and Baselines for Learning 3D Object Rigging. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers (SIGGRAPH Conference Papers '25), August 10–14, 2025, Vancouver, BC, Canada.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3721238.3730743

1 INTRODUCTION

With rapid advances of immersive media comes a growing demand for automated 3D content creation. 3D animation is essential nowadays across the entertainment industry, from game design and movie production to a wide range of AR/VR applications. Yet, producing realistic 3D animations remains one of the most labor-intensive tasks, often requiring enormous expert effort.

Creating an animation of a 3D object generally involves two steps. The first is to describe the desired motion using a set of sparse, interpretable motion handles, such as keypoints or bones—often referred to as a "rig." The second step, known as "skinning," then defines how the 3D object deforms densely according to the movement of these handles. Both steps require specialized expertise and meticulous effort to ensure realistic animation results.

In this paper, our goal is to develop a model that can efficiently and fully automatically turn a static 3D asset into an animatable version. To this end, we propose a learning-based auto-rigging system that generates a rig and the corresponding skinning mechanism from a given 3D mesh. This allows users to create realistic 3D animations by simply manipulating the underlying rig in an intuitive manner.

The key insight is to leverage existing large-scale animated 3D assets handcrafted by expert artists. Large-scale 3D data has proven

[1]Equal contribution, alphabetical order. †Equal advising.

crucial in training high-quality 3D generation models [Liu et al. 2024; Shi et al. 2023]. Notably, the Objaverse datasets [Deitke et al. 2024, 2022], aggregating over 10 million 3D assets from various public sources, have powered many recent state-of-the-art methods. However, prior efforts primarily use such assets for static 3D generation, neglecting the fact that many of these assets also come with valuable animation information skillfully crafted by the artists.

To train a learning-based model for 3D object rigging, we assemble a large-scale dataset of 230K rigged 3D assets curated from the Objaverse-XL Dataset [Deitke et al. 2024], each paired with artist-created rigging and skinning in a unified format, dubbed the *Anymate Dataset*. This dataset is 70 times larger than existing public rigging datasets, as summarized in Table 1, and contains a wide spectrum of 3D objects, ranging from humanoid and animal characters to animated everyday objects like furniture and machines.

Using the Anymate Dataset, we develop a learning-based framework that predicts a 3D bone skeleton and corresponding skinning weights fully automatically from any given 3D object mesh. This framework consists of three sequential modules for joint, connectivity, and skinning weight prediction. For each module, we design several architectures as strong baselines and provide a comprehensive set of evaluations and comparisons on the proposed dataset. These experiments show that the larger scale of training data significantly improves prediction results. Moreover, our proposed architectures demonstrate better scalability, outperforming existing methods by a considerable margin. The dataset, code, and pretrained weights will be publicly released to facilitate future research.

Our contributions are summarized as follows:

(1) We introduce the Anymate Dataset consisting of 230K 3D assets with rigging and skinning information.
(2) We develop a learning-based framework for automatic 3D object rigging and skinning.
(3) We design a comprehensive set of baselines with a variety of architectures and provide thorough evaluations and comparisons on the proposed dataset, providing a reference point for future comparisons.

2 RELATED WORK

2.1 3D Object Rigging Datasets

Auto-rigging has traditionally been approached through geometry-based optimization. Hence, limited efforts were dedicated to collecting and open-sourcing large-scale rigging datasets before the recent rise of data-driven methods. Most of the existing public rigging datasets focus primarily on humanoid and animal characters, such as SMPL [Loper et al. 2015], DeformingThings4D [Li et al. 2021], Maximo [Adobe 2024], Planet Zoo [Wu et al. 2022], Model Resource [Xu et al. 2019], and RaBit [Luo et al. 2023]. The largest of these, Model Resource, contains only 3.3K assets. Our dataset consists of 230K assets—around 70x larger—which is crucial for training robust learning-based models. Table 1 compares the key features of these datasets.

Our Anymate Dataset is built upon the recently released Objaverse-XL Dataset [Deitke et al. 2024], which consists of over 10M 3D assets. The authors of [Li et al. 2024] also extract a set of 14K animated assets in Objaverse [Deitke et al. 2022], dubbed Objaverse-Animation.

Table 1. **3D Object Rigging Datasets.** Existing datasets are limited in both size and variety. In contrast, our Anymate Dataset is 70 times larger than the existing datasets with rigging.

Dataset	Size	Category	Rigging
DeformingThings4D [Li et al. 2021]	2.0K	humanoid, animal	✗
Objaverse-Animation [Li et al. 2024]	14K	generic	✗
SMPL [Loper et al. 2015]	0.2K	human	✓
Mixamo [Adobe 2024]	0.1K	humanoid	✓
Planet Zoo [Wu et al. 2022]	0.25K	animal	✓
RaBit [Luo et al. 2023]	1.5K	humanoid	✓
Model Resource [Xu et al. 2020]	3.3K	humanoid, animal	✓
Anymate Dataset (ours)	230K	generic	✓

Unlike ours, this dataset is curated to train an implicit image-based model for object animation, offering no direct access to rigging and skinning information.

2.2 Skeleton-based Automatic 3D Object Rigging

Early auto-rigging methods, such as Pinocchio [Baran and Popović 2007] and Avatar Reshaping [Feng et al. 2015], fit a template skeleton to a given mesh through optimization. This requirement of a template skeleton heavily limits the range of applicable objects, typically to humanoid or quadruped characters. Several works have explored data-driven approaches for rigging prediction [Ma and Zhang 2023; Xu et al. 2020, 2019].

In particular, [Xu et al. 2019] uses a 3D convolution-based architecture that operates on voxelized shapes, whereas the follow-up works of RigNet [Xu et al. 2020] and Tarig [Ma and Zhang 2023] replace it with graph neural networks that directly process mesh vertices. However, these models are all trained on small-scale datasets with less than 3K objects and often generalize poorly to complex objects, as shown in fig. 7.

Concurrent to our work, RigAnything [Liu et al. 2025] and MagicArticulate [Song et al. 2025] present auto-regressive rigging models learned from large-scale data. Compared to those works, we focus on presenting the dataset, benchmark, and a set of baseline methods for future comparison.

2.3 Non-skeletal Automatic 3D Object Rigging

There also exist works that achieve automatic rigging using representations other than skeleton. We summarize them as handle-based, cage-based, and neural-based methods.

Handle-based. KeypointDeformer [Jakab et al. 2021] and DeepMetaHandles [Liu et al. 2021] pioneer the unsupervised learning of deformation handles. While being annotation-free, these methods often require datasets with minor deformations or aligned poses, which limits their applications to broader categories. In contrast, skeleton-based approaches benefit from large-scale artist-annotated ground truth for supervised learning. We have tried KeypointDeformer [Jakab et al. 2021] but find that instead of manipulating motion, handle-based methods focus on manipulating the mesh's shape, i.e, adjusting an airplane's wing length.

Cage-based. Methods like Neural Cages [Yifan et al. 2020] offer promising approaches to learning rigging without large annotated datasets. However, they struggle with objects with complex hierarchical kinematic structures and large-scale deformations.

Neural-based. Rigging methods with a neural representation [Aigerman et al. 2022; Qin et al. 2023] are promising for modeling complex motions like faces, but often require an extensive amount of observations under known poses for training. Moreover, it remains challenging to obtain an interpretable neural representation.

In summary, different rigging modalities serve distinct purposes. The skeleton-based approach benefits from large-scale annotated datasets, applies to diverse categories, and integrates seamlessly with traditional graphics pipelines. Other modalities excel in specific contexts, such as with aligned datasets or synthetic data.

2.4 Automatic Skinning

In 3D animation, artists usually need to paint skinning weights on a mesh to indicate how the vertices should deform in response to bone movement. Techniques like Linear Blend Skinning (LBS) [Magnenat et al. 1988] are then used to drive the mesh based on bone animations. Various geometric methods have been proposed to automatically determine skinning weights [Bang and Lee 2018; Chen et al. 2021; Dionne and de Lasa 2013; Dionne and De Lasa 2014; Jacobson et al. 2011; Kavan and Sorkine 2012; Song et al. 2024], such as using Laplacian energy [Jacobson et al. 2011] or geodesic distance [Dionne and de Lasa 2013]. However, these methods often struggle with objects of complex geometry.

Previous data-driven approaches have shown promise for better generality [Liu et al. 2019; Ma and Zhang 2023; Mosella-Montoro and Ruiz-Hidalgo 2022; Xu et al. 2020]. Nevertheless, similar to rigging, previous methods either focus on specific categories and rely on templates [Liao et al. 2024] or train on limited datasets [Mosella-Montoro and Ruiz-Hidalgo 2022; Xu et al. 2020].

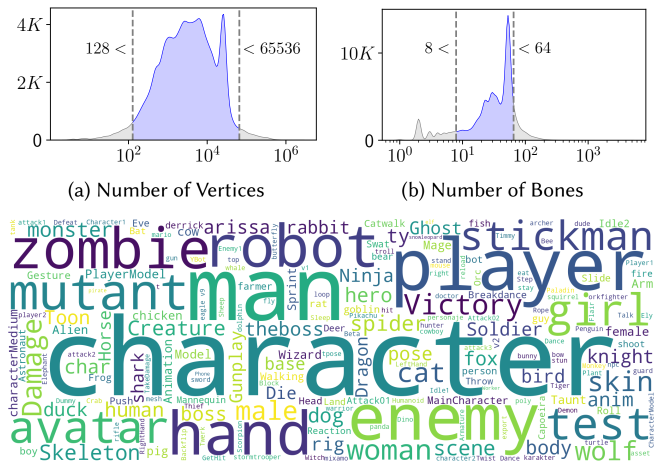
3 ANYMATE DATASET

To learn 3D object auto-rigging, a sizable collection of examples is essential. We introduce the *Anymate Dataset*, a large-scale dataset of manually crafted animated 3D objects, building upon the existing Objaverse-XL Dataset [Deitke et al. 2024]. In the following, we first give an overview of the representation for 3D object animation, and then describe the procedures used to build the Anymate Dataset.

3.1 Preliminary: 3D Object Animation

When artists create an animation of a 3D object, two key components are typically required: rigging and skinning. The former establishes a set of sparse, interpretable handles that serve as a scaffold to outline the intended motion. The latter, in turn, defines how the 3D object should deform densely in response to the motion scaffold. In modern computer graphics, the rig is often represented by a skeleton of bones, and the skinning is usually defined by weights associating mesh vertices to these underlying bones.

Concretely, given a 3D mesh with vertices denoted by $\mathbf{V} \subset \mathbb{R}^3$, we first define a set of bones, indexed by $b = 1, ..., B$. Each bone is an oriented line segment specified by a pair of head and tail joints $\mathbf{J}_b^{\text{head}}, \mathbf{J}_b^{\text{tail}} \in \mathbb{R}^3$ in the world coordinate space. In practice, these



(a) Number of Vertices
(b) Number of Bones
(c) Objects in the Anymate Dataset

Fig. 2. **Statistics of the Anymate Dataset.** We filter out assets with a vertex count beyond $(128, 65535)$ and a bone count beyond $(8, 64)$, as shown in (a) and (b). The remaining assets span a wide spectrum of objects, as visualized in the word cloud in (c).

bones are typically connected via shared joints, forming a kinematic tree structure.

To animate the object, we then deform each vertex $\mathbf{V}_i$ of the mesh according to the poses ξ of these bones via the Linear Blend Skinning (LBS) equation [Lewis et al. 2000; Magnenat et al. 1988]:

$$\mathbf{V}'_i(\xi) = \left(\sum_{b=1}^{B} w_{i,b} G_b(\xi) G_b(\xi^*)^{-1} \right) \mathbf{V}_i, \quad G_1 = g_1,\ G_b = G_{\pi(b)} \circ g_b,\ g_b(\xi_b) \in SE(3). \tag{1}$$

Here, $\xi = \{\xi_1, ..., \xi_B\}$ denotes the poses of the bones in their local coordinate frames, and ξ^* denotes the rest pose, which can be derived from the joint locations $\mathbf{J}_b^{\text{head}}$ and $\mathbf{J}_b^{\text{tail}}$ (see the supp. mat.). G stands for the global transformation matrix after composing all the local transformations g_b along the kinematic chain. $\pi(b)$ retrieves the index of the parent of bone b in the kinematic tree. $w_{i,b}$ is the *skinning weight* of vertex $\mathbf{V}_i$ with respect to bone b. Essentially, for each bone b, vertex $\mathbf{V}_i$ in the world coordinate frame is first transformed to the local coordinate frame of the bone following the inverse kinematic chain G_b^{-1}, and then reposed together with the bone following the forward kinematic chain G_b. The final position of the vertex $\mathbf{V}'_i$ is the sum of transformations from all bones weighted by $w_{i,b}$.

Therefore, to animate a mesh, we need a set of bones $\mathcal{B}$, each specified by a pair of joints $\mathcal{B}_b = (\mathbf{J}_b^{\text{head}}, \mathbf{J}_b^{\text{tail}})$, and a function $\mathcal{S}$ that returns the skinning weights for a given point on the surface. The goal of our model is thus to predict these bones and skinning weights from a given mesh.

3.2 Dataset Processing

To train such a model, we prepare a large dataset of rigged 3D objects. Our dataset contains 230K 3D assets of common objects from the Objaverse-XL dataset [Deitke et al. 2024]. Each asset comes with a 3D mesh, a bone skeleton, and skinning weights for each mesh vertex in a unified format.

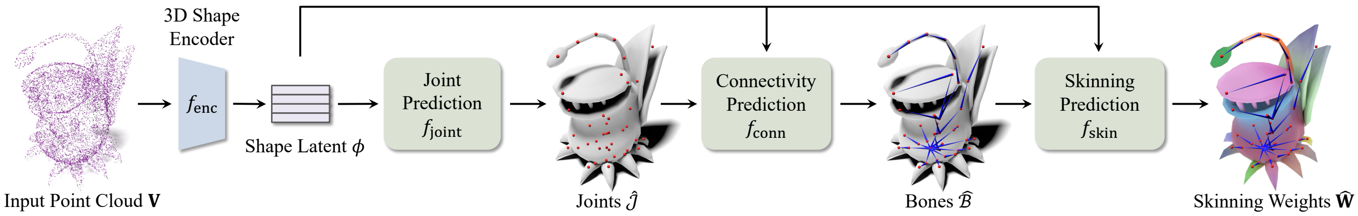


Fig. 3. **Overview of Our Learning-based Auto-rigging Framework.** Given an input 3D mesh, we first sample surface points $\mathbf{V}$ and extract shape latent features ϕ. Three sequential modules then predict a set of joints $\hat{\mathcal{J}}$, joint connectivity to establish bones $\hat{\mathcal{B}}$, and finally the skinning weights with respect to each bone $\hat{\mathbf{W}}$, all based on the shape latent. See section 4 for details.

Filtering. Objaverse-XL provides links to over 10 million 3D assets from various public sources in various formats. We identify the assets that potentially contain rigging and skinning information with a file format of *.fbx*, *.dae*, *.gltf*, or *.glb*. While Objaverse-XL reports 438K objects with armatures, we find only 256K that are still publicly available. We then filter the assets, retaining roughly 200K with a vertex count of $128 < |\mathbf{V}| < 65,535$ and a bone count of $8 < |\mathcal{B}| < 64$. Fig. 2 gives an illustration of the statistics. After this, we eliminate redundant data based on skeleton similarity. Next, we remove assets where most of the skeleton (> 70%) lies outside the mesh. Many assets contain animation information, which is used to augment the dataset. We use no more than three extra key-frame poses from the same asset. After augmentation, we unify the data format and fix bones and joints. More details can be found in the supp. mat.

Pre-processing. We load these assets into Blender[1] and extract the mesh, armature (bones), and skinning weights. For training efficiency, we resample 8,192 points on the mesh, and recompute their skinning weights using barycentric interpolation based on the original annotated skinning weights at the vertices. Finally, for each asset, we export the original mesh in a unified *.obj* format, as well as the resampled point clouds and the corresponding bone locations and skinning weights.

Dataset Statistics. The final dataset contains 230, 716 assets. We randomly select 5.6K instances as the test set (Anymate-test), and use the rest 225K for training (Anymate-train). To facilitate efficient ablation experiments, we also construct a smaller 14K training set (Anymate-small) with a subset of training instances that are included in the original Objaverse 1.0 Dataset [Deitke et al. 2022], which is likely of higher quality.

4 LEARNING TO RIG 3D OBJECTS

Using the Anymate Dataset, we train learning-based models to predict bone skeleton and skinning weights from a given 3D object mesh. These allow users to create 3D animations of the object by simply manipulating the skeleton.

We divide this task into three sequential sub-tasks: joint prediction, connectivity prediction, and skinning weight prediction, as outlined in Fig. 3. Starting with an input 3D mesh, we first use an encoder f_{enc} to extract shape features, which are then fed into the three subsequent prediction modules. The first module f_{joint} predicts a set 3D joints. The second module f_{conn} then connects these joints into bones by predicting binary connectivity for each pair of joints, forming a skeleton. Finally, the skinning module f_{skin} estimates the skinning weight between each mesh vertex and each bone, based on the constructed skeleton. For each sub-task, we experiment with various architectures as baselines and conduct evaluations on our Anymate Dataset, comparing against previous methods.

In the following, we first introduce the shape encoder (section 4.1) shared across all prediction modules, and then discuss each individual module (sections 4.2 to 4.4), followed by visual results generated using the predictions (section 4.5).

4.1 Shape Encoding

Given a 3D mesh, we first uniformly resample 8, 192 points across its surface and extract shape latent features ϕ using an encoder f_{enc}. We experiment with two popular transformer-based point cloud encoders, **Michelangelo** [Zhao et al. 2024] and **Point-BERT** [Yu et al. 2022]. Michelangelo trains a perceiver-based transformer with learnable query tokens and several self-attention blocks. Point-BERT trains a discrete Variational Auto-Encoder (dVAE) which first maps a point cloud into a set of patch tokens and subsequently mixes them via a transformer. Both models encode the point cloud into a set of feature tokens (257 and 513 respectively). In our experiments, we load the pre-trained weights from existing models and fine-tune them with each subsequent prediction module individually. Specifically, we load the Michelangelo model pre-trained on the ShapeNet Dataset (50K) [Chang et al. 2015], and the Point-BERT model pre-trained on the Cap3D Dataset (660K) [Luo et al. 2024] released by PointLLM [Xu et al. 2024b], as Cap3D is also derived from Objaverse [Deitke et al. 2022].

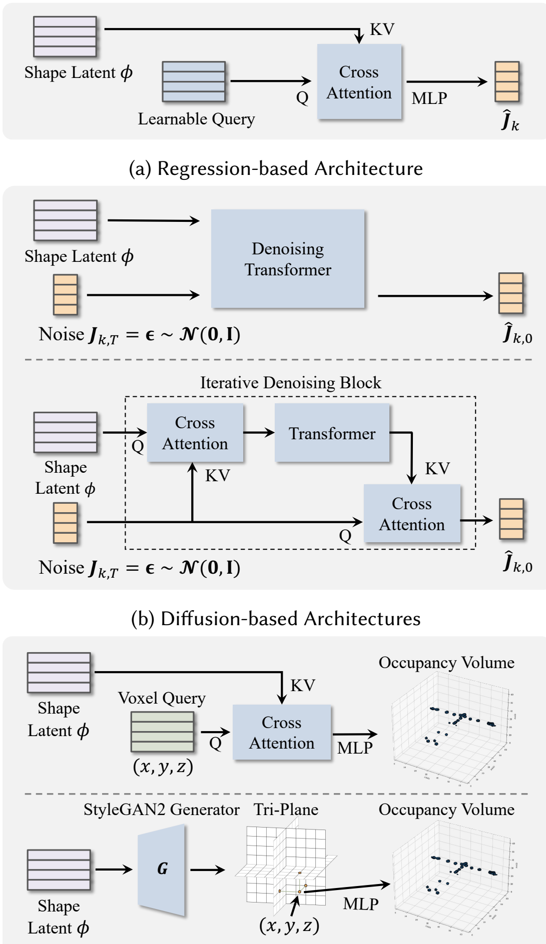
4.2 Joint Prediction

After obtaining the shape latent ϕ, the joint prediction module f_{joint} predicts a set of joints $\hat{\mathcal{J}} = f_{\text{joint}}(\phi) = \{\hat{\mathbf{J}}_k\}_{k=1}^{K}$, where each joint $\hat{\mathbf{J}}_k \in \mathbb{R}^3$ is a 3D coordinate in world space, and the number of joints K may vary across instances.

4.2.1 Architectures. We explore three types of architectures for f_{joint}: regression-based, diffusion-based, and volume-based.

Regression-based. A regression-based model is perhaps the most intuitive approach for joint prediction, which simply regresses a set of 3D coordinates from the extracted shape features and can be directly supervised by the ground-truth joints during training. However, there are two unique features that make our joint prediction task different from typical regression problems: (1) the number of

[1] https://www.blender.org/



(a) Regression-based Architecture

(b) Diffusion-based Architectures

(c) Volume-based Architectures

Fig. 4. **Architectures for Joint Prediction.** The model predicts a set of joint locations $\{\hat{J_k}\}$ from the shape latent ϕ (section 4.2.1).

target joints varies across instances, and (2) the order of the joints is immaterial at this stage[2]. To address these challenges, we design a transformer-based architecture [Jaegle et al. 2021; Vaswani et al. 2017], which is permutation-equivariant and has proven to be highly scalable, followed by a clustering step [Ester et al. 1996].

The architecture is illustrated in Fig. 4a. Specifically, we use a single-layer perceiver-based transformer [Jaegle et al. 2021] with a fixed number of $K_{\text{pred}} = 96$ learnable query tokens and the shape latent ϕ as key and value tokens, followed by a 3-layer MLP to produce K_{pred} 3D joint locations $\hat{\mathcal{J}}$. We supervise the model directly using a standard Chamfer Distance between the predicted joints and the ground-truth joints.

Since the number of GT joints K_{gt} varies across training instances, we cluster the raw predicted joints using the technique of [Ester et al. 1996] and use the cluster centers as the final output.

[2] Although an order could technically be established based on the skeleton, inconsistencies may arise across instances created by different artists.

Diffusion-based. Another commonly used architecture for point set generation is diffusion-based models [Sohl-Dickstein et al. 2015], which tend to excel with large-scale training. However, unlike most existing work that focuses either on generating dense point clouds with high geometric redundancy [Huang et al. 2024; Luo and Hu 2021; Nichol et al. 2022] or on producing a fixed set of semantic keypoints like human poses [Xu et al. 2024a; Zhou et al. 2023], our model is designed to predict salient joints for animation which may vary across instances.

Following DDPM [Ho et al. 2020], the forward diffusion process injects noise $\boldsymbol{\epsilon}_t = \{\boldsymbol{\epsilon}_{k,t}\}_{k=1}^{K}$ independently to the GT joints $\mathcal{J}_0 = \{\mathbf{J}_{k,0}\}_{k=1}^{K}$ at a time step $t \in [1, T]$:

$$\mathbf{J}_{k,t} = \sqrt{\overline{\alpha}_t}\mathbf{J}_{k,0} + \sqrt{1 - \overline{\alpha}_t}\boldsymbol{\epsilon}_{k,t}, \quad \boldsymbol{\epsilon}_{k,t} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}),$$

where $\overline{\alpha}_t$ signifies the noise schedule. We then train a denoiser $\boldsymbol{\epsilon}_\theta(\mathcal{J}_t, t \mid \phi)$ that learns to predict the noise from the noised joints $\mathcal{J}_t = \{\mathbf{J}_{k,t}\}$ conditioned on the shape latent ϕ, using the loss:

$$L_{\text{diff}} = \mathbb{E}_{t, \mathcal{J}_0, \boldsymbol{\epsilon}_t}\left[\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathcal{J}_t, t \mid \phi)\|_2^2\right].$$

We implement the denoiser using a permutation-equivariant transformer architecture with two variations, as illustrated in Fig. 4b:

(1) **Concatenation-based** model inspired by Point-E [Nichol et al. 2022], where the noisy joints $\mathbf{J}_{k,t}$ are concatenated with the shape latent ϕ before passing through a transformer;
(2) **Cross-attention-based** model inspired by Point-Infinity [Huang et al. 2024], where cross-attention and transformer modules iterate across two branches, between the noisy joints $\mathbf{J}_{k,t}$ and the shape latent ϕ.

At inference time, we use a DDIM [Song et al. 2021] sampler to generate a set of joints $\hat{\mathcal{J}}_0$ from Gaussian noise. This transformer-based diffusion architecture allows users to specify an arbitrary number of joints at varying levels of granularity during inference by adjusting the shape of the initial noisy point cloud, as demonstrated in the supp. mat.

Volume-based. The final option we explore is a dense volume-based model, where the task is framed as predicting the probably of a point in space being near a joint. To this end, we experiment with two commonly used 3D field representations, an **implicit neural field** [Mildenhall et al. 2021] and a **tri-plane** representation [Chan et al. 2022], as illustrated in Fig. 4c.

For the implicit neural field, we follow Michelangelo [Zhao et al. 2024] and implement it using a perceiver-based transformer. It takes in a set of 3D coordinates $\mathbf{x}_i$ conditioned on the shape latent ϕ, and predicts a scalar $\sigma(\mathbf{x}_i)$ indicating the probability that a joint is nearby. For direct supervision, we apply a 3D Gaussian kernel to each GT joint with a standard deviation γ (= 2 voxel) and take the maximum response across all joints at each 3D location. At inference time, we first extract a 64^3 voxel grid of joint probability scores from the implicit field and apply a threshold to obtain a set of activated grid points. We then apply the same clustering technique [Ester et al. 1996] to group these points into clusters.

For the tri-plane representation, we follow the implementation of EG3D [Chan et al. 2022] with a StyleGAN2 generator [Karras et al. 2020] conditioned on the shape latent. We then use a similar procedure to extract keypoints from the tri-planes.

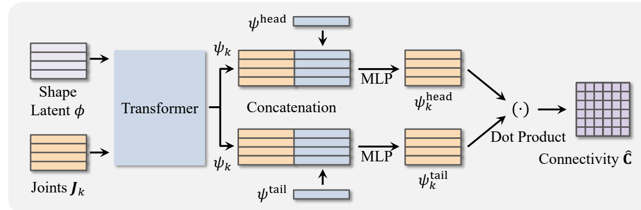
Table 2. **Quantitative Evaluation of Joint Prediction.** Our proposed architectures scale more effectively with larger training data, outperforming existing methods by a significant margin.

Model	CD ↓	EMD ↓	Training Time
Pinocchio [Baran and Popović 2007]	0.198	0.659	-
trained on Anymate-small (14K)			
RigNet [Xu et al. 2020]	0.094	0.131	16h
Mich-Regress	0.099	0.124	20h
Mich-Diff$_{\text{Concat}}$	0.142	0.175	26h
Mich-Diff$_{\text{Cross}}$	0.128	0.160	47h
Mich-Vol$_{\text{Implicit}}$	0.147	0.127	33h
Mich-Vol$_{\text{TriPlane}}$	0.198	0.186	20h
Bert-Regress	**0.091**	**0.113**	11h
Bert-Diff$_{\text{Concat}}$	0.147	0.177	65h
Bert-Diff$_{\text{Cross}}$	0.137	0.167	37h
Bert-Vol$_{\text{Implicit}}$	0.123	0.126	29h
Bert-Vol$_{\text{TriPlane}}$	0.192	0.171	19h
trained on Anymate-train (225K)			
RigNet [Xu et al. 2020]	0.089	0.127	114h
Bert-Regress	**0.077**	**0.098**	79h
Mich-Diff$_{\text{Cross}}$	0.083	0.104	151h

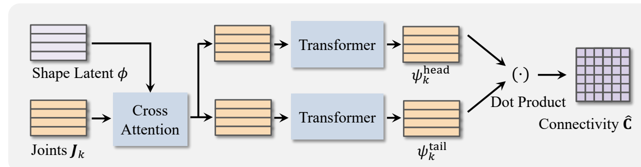
4.2.2 Existing Methods. We also evaluate two existing auto-rigging methods on our dataset, Pinocchio [Baran and Popović 2007] and RigNet [Xu et al. 2020]. Pinocchio relies on predefined template skeletons (humanoid or quadruped), and fit a template to a given mesh via optimization. RigNet proposes a learning-based system similar to ours, but uses primarily Graph Neural Networks. We adopt the official public implementation for both methods, and train the RigNet model on our larger Anymate Dataset. We also compare with two commercial auto-rigging tools, Maya [Autodesk 2024] and Animate Anything [World 2024], qualitatively in section 4.5.

4.2.3 Quantitative Results. For a comprehensive analysis of different architectures, we first train 10 models on the Anymate-small set, exhausting all combinations of the 2 shape encoders and 5 prediction architectures. We then evaluate these models and the existing methods (RigNet and Pinocchio) on the Anymate-test set. For quantitative evaluation, we compute two standard metrics, **Chamfer Distance** (CD) and **Earth Mover's Distance** (EMD), both computed between the predicted and the annotated joints (see supp. mat.).

The results are summarized in Table 2. Among the proposed architectures trained on Anymate-small, Bert-Regress outperforms the two existing methods. We select two architectures for training on the full Anymate-train set: Bert-Regress, and Mich-Diff$_{\text{Cross}}$. The regression-based model performs best on the Anymate-small subset, while the diffusion-based model has the potential for scaling with larger training data. As shown in Table 2, the performance of the model improves as training set increases from 14K to 225K, in particular, from 0.091 to 0.077 for Mich-Diff$_{\text{Cross}}$. Despite comparable performance, compared to the regression-based models, the diffusion-based model offers the flexibility for a user to specify an arbitrary number of target joints, but requires a more time-consuming iterative inference procedure. In comparison, the classic geometric method, Pinocchio [Baran and Popović 2007], performs significantly



(a) Token-conditioned Architecture



(b) Two-branch Architecture

Fig. 5. **Architectures for Connectivity Prediction.** The model predicts a pair-wise connectivity matrix $\hat{\mathbf{C}}$ from a set of joints $\{\mathbf{J}_k\}$, conditioned on the shape latent ϕ (section 4.3.1).

worse. RigNet [Xu et al. 2020] scales poorly with the larger dataset, leading to degraded performance. We also report the training time in terms of GPU hours on a L40S GPU for Anymate-small training and on eight TITANRTX GPUs for Anymate-train training. More analysis is provided in the supp. mat.

4.3 Connectivity Prediction

After obtaining a set of joints $\{\hat{\mathbf{J}}_k\}_{k=1}^K$, the second module f_{conn} predicts connectivity between the joints to establish a bone skeleton. Specifically, it takes the predicted joints and the shape latent ϕ as inputs, and predicts a connectivity matrix $\hat{\mathbf{C}} = f_{\text{conn}}(\hat{\mathcal{J}}, \phi) \in \mathbb{R}^{K\times K}$, where each element $\hat{c}_{i,j}$ indicates the probability that $\hat{\mathbf{J}}_i$ and $\hat{\mathbf{J}}_j$ are connected by a bone, with $\hat{\mathbf{J}}_i$ as the head joint and $\hat{\mathbf{J}}_j$ as the tail joint.

We train the model using the ground-truth joints $\{\mathbf{J}_k\}_{k=1}^K$ as input and compute binary cross entropy between the predicted and ground-truth connectivity matrices, $\hat{\mathbf{C}}$ and $\mathbf{C}$. The GT matrix $\mathbf{C} \in \{0,1\}^{K\times K}$ can be generated from the annotated bones in the training assets. During inference, the trained model is used to infer the connectivity of the predicted joints $\{\hat{\mathbf{J}}_k\}_{k=1}^K$.

In practice, Blender only allows each bone to have *at most one* parent bone. This implies that each joint can only serve as the *tail* joint of either one single bone or none (*i.e.*, a root joint), meaning the sum of each column of the GT connectivity matrix $\mathbf{C}$ is either 1 or 0 (root). Therefore, if a joint $\mathbf{J}_i$ is a root joint, we assign the (i, i) entry of $\mathbf{C}$ to 1, *i.e.*, $c_{i,i} = 1$, while keeping the rest of the column as 0. At inference time, we can exploit this fact and simply take the `argmax` of each column of the predicted connectivity matrix $\hat{\mathbf{C}}$ to identify root joints or retrieve the head joints. This enables us to construct a kinematic chain for animation.

4.3.1 Architectures. We design two architectures to predict this connectivity matrix $\hat{\mathbf{C}}$ given a set of joints $\{\mathbf{J}_k\}_{k=1}^K$ conditioned on the shape latent ϕ: a **token-conditioned** architecture and a **two-branch** architecture, as illustrated in Fig. 5.

Token-conditioned. Given the input joint coordinates, this model first applies positional encoding and extracts embeddings via a

Table 3. **Quantitative Evaluation of Connectivity Prediction.** Despite overfitting on the smaller training set, our proposed architectures scale effectively with larger training data.

Model	Precision ↑	Recall ↑	Training Time
trained on Anymate-small (14K)			
RigNet [Xu et al. 2020]	60.6%	57.9%	8h
Mich-Token	58,0%	55.8 %	21h
Mich-2Branch	60.5%	58.5%	19h
Bert-Token	59.3%	57.2 %	15h
Bert-2Branch	**61.6%**	**59.7%**	12h
trained on Anymate-train (225K)			
RigNet [Xu et al. 2020]	47.9%	50.4%	51h
Bert-2Branch	**84.6%**	**83.5%**	76h

linear layer. These joint embeddings together with the shape latent ϕ go through a transformer to obtain a set of per-joint feature tokens $\{\psi_k\}$. Each token ψ_k is then concatenated with a (shared) learnable 'head' token ψ^{head} and a (shared) learnable 'tail' token ψ^{tail}, and goes through two separate MLPs, generating a pair of head-conditioned and tail-conditioned tokens ψ_k^{head} and ψ_k^{tail}. Finally, we simply compute the dot product between two sets of tokens $\{\psi_k^{\text{head}}\}$ and $\{\psi_k^{\text{tail}}\}$ to obtain the output matrix $\hat{\mathbf{C}}$.

Two-branch. Instead of conditioning on 'head' and 'tail' tokens, The two-branch model processes the joint embeddings through two independent transformer branches, after incorporating the shape latent via cross attention. Similarly, the final matrix $\hat{\mathbf{C}}$ is obtained by taking the dot product between the two sets of output tokens from each branch.

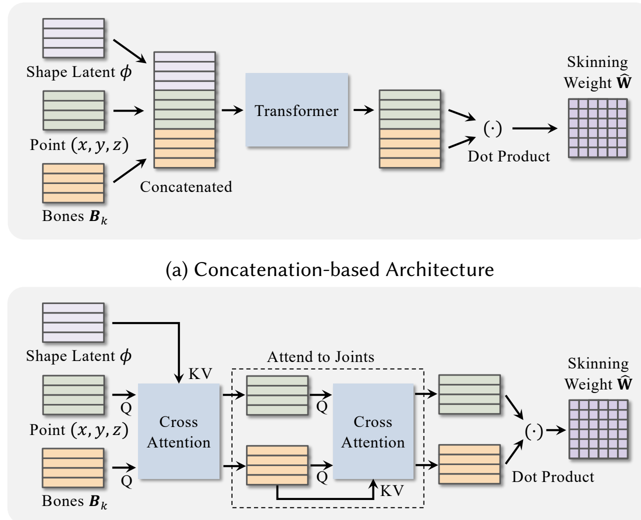
4.3.2 Existing Methods. We compare our models with the bone prediction module of RigNet [Xu et al. 2020] on our dataset. Rather than using the dot product, they directly feed in each pair of joints to an MLP and predict their connectivity, which is less computationally efficient ($O(K^2)$). Furthermore, they use a separate network to predict the probability of a joint being a root. In comparison, our models are more efficient by directly obtaining a connectivity matrix via a dot product.

4.3.3 Quantitative Results. Similar to joint evaluation, we evaluate all architecture combinations on Anymate-small, and train the top performing models on the full Anymate-train set. We report two standard metrics for comparison: **Precision** and **Recall**.

The results are summarized in Table 3. On Anymate-small, we in fact observe overfitting for all of our models, whereas RigNet performs reasonably well. However, on the full Anymate-train set, we observe the opposite: RigNet severely underfits potentially due to a limited model size, whereas our proposed architectures scale well with the larger dataset, outperforming RigNet by a significant margin. Additionally, the two-branch model tends to perform better than the token-conditioned model.

4.4 Skinning Weight Prediction

After obtaining the bones $\mathcal{B}$, the final step is to predict the skinning weights $\hat{\mathbf{W}}_i = f_{\text{skin}}(\mathbf{V}_i \mid \mathcal{B}, \phi) \in \mathbb{R}^B$ of each vertex $\mathbf{V}_i$ with respect



(a) Concatenation-based Architecture

(b) Cross-attention-based Architecture

Fig. 6. **Skinning Weight Prediction Architectures.** The model predicts a pair-wise skinning weight matrix $\hat{\mathbf{W}}$ between each point and each bone $\mathbf{B}_k$, conditioned on the shape latent ϕ (section 4.4.1).

to the bones, where $B = |\mathcal{B}|$ denotes the number of bones. In practice, our model takes in the joint locations $\mathbf{J}$ of the bones, and for efficiency, processes a set of points in a batch using a transformer architecture, producing a skinning weight matrix $\hat{\mathbf{W}} \in \mathbb{R}^{|\mathbf{V}| \times B}$. For training, we use the resampled point clouds explained in section 3.2 together with the annotated bones as inputs, and compute the cosine similarity loss against the ground-truth skinning weights $\mathbf{W}$ (see supp. mat. for details). During inference, we can infer the skinning weights of the original mesh vertices over the predicted bones.

4.4.1 Architectures. We experiment with two architectures illustrated in Fig. 6.

Concatenation-based. This model simply concatenates the point embeddings with the bone embeddings and shape latent ϕ, and mixes these embeddings to generate two set of token features: per-point tokens $\{\phi_i^{\text{point}}\}_{i=1}^{|\mathbf{V}|}$ and per-bone tokens $\{\phi_b^{\text{bone}}\}_{b=1}^{B}$. Reusing the dot product trick in connectivity prediction followed by a `softmax` over all bones, we obtain a skinning weight matrix $\hat{\mathbf{W}}$.

Cross-attention-based. The second variant stacks two consecutive cross-attention layers. The per-point tokens $\{\phi_i^{\text{point}}\}_{i=1}^{|\mathbf{V}|}$ and per-bone tokens $\{\phi_b^{\text{bone}}\}_{b=1}^{B}$ is generated by passing the cross-attention layers. Finally, the skinning weight matrix $\hat{\mathbf{W}}$ is obtained via a dot product followed by a `softmax`.

4.4.2 Existing Methods. We compare against RigNet [Xu et al. 2020] as well as a traditional geometric method, GeoVoxel [Dionne and de Lasa 2013]. RigNet uses a graph neural network based architecture and only predicts the skinning weights with respect to the five nearest bones. In practice, we found this often insufficient, in particular with erroneous joint predictions. We train and evaluate RigNet on our proposed dataset using the official implementation and the

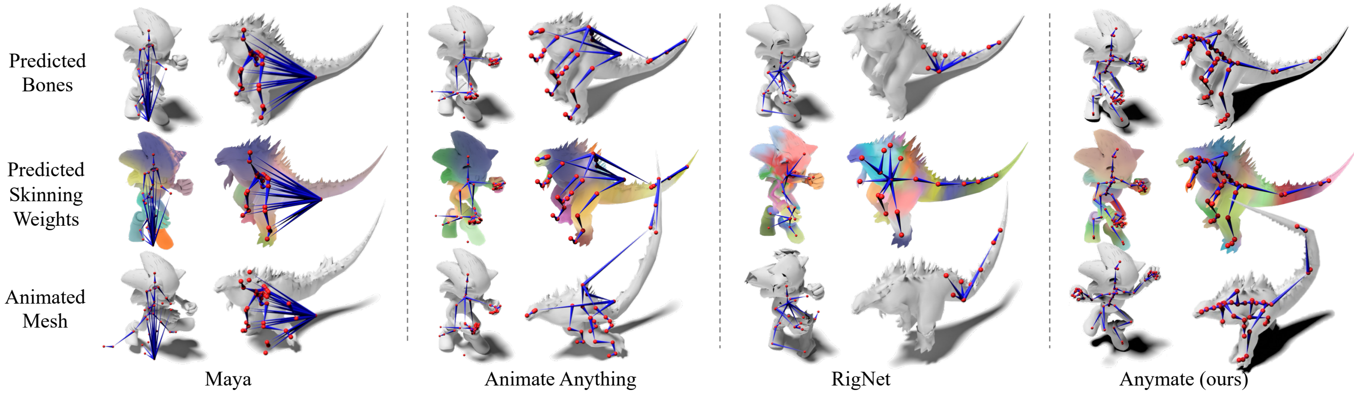


Fig. 7. **Visual Results and Comparisons with Existing Methods.** All methods estimate a bone skeleton and skinning weights from an input 3D mesh. Commercial auto-rigging tools like Maya [Autodesk 2024] and Animate Anything [World 2024] typically rely on template skeletons and generalize poorly well to complex shapes. Despite training on our larger dataset, RigNet [Xu et al. 2020] also struggles to predict reasonable skeleton and skinning weights, resulting in severe distortion in animation. In contrast, our model produces more plausible predictions and animations.

Table 4. **Quantitative Evaluation of Skinning Weight Prediction.** Our proposed architectures scale effectively with larger training data, outperforming GeoVoxel [Dionne and de Lasa 2013] and RigNet [Xu et al. 2020].

Model	CE ↓	Cos ↑	MAE ↓	Train Time
GeoVoxel	3.569	0.407	0.056	-
trained on Anymate-small (14K)				
RigNet	1.573	0.671	0.031	11h
Mich-Concat	1.169	0.754	0.029	41h
Mich-Cross	1.270	0.766	0.026	21h
Bert-Concat	1.148	0.768	0.027	53h
Bert-Cross	**1.134**	**0.801**	**0.025**	32h
trained on Anymate-train (225K)				
RigNet	1.521	0.693	0.028	49h
Bert-Cross	**0.741**	**0.915**	**0.014**	83h

same hyper-parameters. GeoVoxel calculates skinning weights simply based on the geodesic distance after voxelization. We evaluate GeoVoxel on our test set using the implementation from Maya [Autodesk 2024].

4.4.3 Quantitative Results. Similarly, we train the models on both the Anymate-small and Anymate-train sets, and evaluate on the Anymate-test set. We report three metrics: **Cross Entropy** (CE), **Cosine Similarity** (Cos), and **Mean Absolute Error** (MAE).

The results are summarized in Table 4. Overall, GeoVoxel performs poorly on our test set, serving as a purely geometry-based lower-bound. Our proposed architectures scale well with the larger training set, outperforming the baselines by a significant margin. In contrast, little improvement is observed with RigNet as the training data size increases. Among our proposed architectures, Bert-Cross achieves the best results.

4.5 Visual Results

After all individual modules have been trained, we can run inference on any new test mesh and obtain the skeleton and skinning weights automatically in less than a minute. With these predictions, we can easily animate the object by manipulating the predicted skeleton, as visualized in Fig. 8 and Fig. 7 . For all visualizations, we use the best-performing model variants for inference, namely, Mich-Regress (joint), Bert-2Branch (connectivity), and Bert-Cross (skinning).

We compare our results against the previous method, RigNet [Xu et al. 2020], and two commercial auto-rigging tools, Maya [Autodesk 2024] and Animate Anything [World 2024]. Maya only supports fitting a humanoid skeleton to the input mesh and generalizes poorly to non-humanoid objects. Animate Anything, on the other hand, classifies the object into one of a few categories and fits a category-specific skeleton. Both tools produce inaccurate skeletons for complex geometries, resulting in unrealistic animations. For a fair comparison, we retrain the RigNet model on our full dataset. However, it still generalizes poorly to challenging instances. In comparison, our model predicts highly accurate bone skeleton and skinning weights, enabling realistic animations.

5 CONCLUSIONS

We present the Anymate Dataset, the largest object rigging dataset to date, containing 230K 3D assets with rigging and skinning information. Using this dataset, we proposed an effective learning-based auto-rigging framework, with three modules for joint, connectivity, and skinning prediction. For each module, we carefully designed a set of baselines and thoroughly evaluated their performance using the proposed dataset. Experiment results have demonstrated that our proposed baselines scale effectively with the larger training data, outperforming previous methods and existing commercial auto-rigging tools, and provide a reference point for future research in data-driven auto-rigging.

ACKNOWLEDGMENTS

This work is in part supported by ONR YIP N00014-24-1-2117, NSF RI #2211258 and #2338203, the Stanford Institute for Human-Centered AI (HAI), the Stanford Center for Integrated Facility Engineering (CIFE), and Meta.

Fig. 8. **Additional Qualitative Comparison with RigNet [Xu et al. 2020].** Our model consistently produces more accurate predictions. Note that some of the assets do not come with texture, hence marked with 'N/A'.

REFERENCES

Adobe. 2024. Mixamo. https://www.mixamo.com/.

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).

Autodesk. 2024. Maya. https://www.autodesk.com/products/maya/.

Seungbae Bang and Sung-Hee Lee. 2018. Spline interface for intuitive skinning weight editing. *ACM TOG* 37, 5 (2018), 1–14.

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

Eric R Chan, Connor Z Lin, Matthew A Chan, Koki Nagano, Boxiao Pan, Shalini De Mello, Orazio Gallo, Leonidas J Guibas, Jonathan Tremblay, Sameh Khamis, et al. 2022. Efficient geometry-aware 3d generative adversarial networks. In *CVPR*. 16123–16133.

Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. 2015. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012* (2015).

Xu Chen, Yufeng Zheng, Michael J Black, Otmar Hilliges, and Andreas Geiger. 2021. Snarf: Differentiable forward skinning for animating non-rigid neural implicit shapes. In *ICCV*. 11594–11604.

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. *NeurIPS* 36 (2024).

Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. 2022. Objaverse: A Universe of Annotated 3D Objects. *arXiv preprint arXiv:2212.08051* (2022).

Olivier Dionne and Martin de Lasa. 2013. Geodesic voxel binding for production character meshes. In *SIGGRAPH*. 173–180.

Olivier Dionne and Martin De Lasa. 2014. Geodesic binding for degenerate character geometry using sparse voxelization. *IEEE TVCG* 20, 10 (2014), 1367–1378.

Martin Ester, Hans-Peter Kriegel, Jorg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise.. In *KDD*, Vol. 96. 226–231.

Andrew Feng, Dan Casas, and Ari Shapiro. 2015. Avatar reshaping and automatic rigging using a deformable model. In *SIGGRAPH*. 57–64.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. In *NeurIPS*, Vol. 33. 6840–6851.

Zixuan Huang, Justin Johnson, Shoubhik Debnath, James M Rehg, and Chao-Yuan Wu. 2024. PointInfinity: Resolution-Invariant Point Diffusion Models. In *CVPR*. 10050–10060.

Alec Jacobson, Ilya Baran, Jovan Popovic, and Olga Sorkine. 2011. Bounded biharmonic weights for real-time deformation. *ACM TOG* 30, 4 (2011), 78.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. 2021. Perceiver: General perception with iterative attention. In *ICML*. 4651–4664.

Tomas Jakab, Richard Tucker, Ameesh Makadia, Jiajun Wu, Noah Snavely, and Angjoo Kanazawa. 2021. Keypointdeformer: Unsupervised 3d keypoint discovery for shape control. In *CVPR*. 12783–12792.

Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. 2020. Analyzing and Improving the Image Quality of StyleGAN. In *CVPR*.

Ladislav Kavan and Olga Sorkine. 2012. Elasticity-inspired deformers for character articulation. *ACM TOG* 31, 6 (2012), 1–8.

J. P. Lewis, Matt Cordner, and Nickson Fong. 2000. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *SIGGRAPH*.

Ruining Li, Chuanxia Zheng, Christian Rupprecht, and Andrea Vedaldi. 2024. Puppet-Master: Scaling Interactive Video Generation as a Motion Prior for Part-Level Dynamics. *arXiv preprint arXiv:2408.04631* (2024).

Yang Li, Hikari Takehara, Takafumi Taketomi, Bo Zheng, and Matthias Nießner. 2021. 4dcomplete: Non-rigid motion estimation beyond the observable surface. In *ICCV*. 12706–12716.

Zhouyingcheng Liao, Vladislav Golyanik, Marc Habermann, and Christian Theobalt. 2024. VINECS: Video-based Neural Character Skinning. In *CVPR*. 1377–1387.

Isabella Liu, Zhan Xu, Wang Yifan, Hao Tan, Zexiang Xu, Xiaolong Wang, Hao Su, and Zifan Shi. 2025. RigAnything: Template-Free Autoregressive Rigging for Diverse 3D Assets. *arXiv preprint arXiv:2502.09615* (2025).

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 TOG* 38, 4 (2019), 1–12.

Minghua Liu, Ruoxi Shi, Linghao Chen, Zhuoyang Zhang, Chao Xu, Xinyue Wei, Hansheng Chen, Chong Zeng, Jiayuan Gu, and Hao Su. 2024. One-2-3-45++: Fast single image to 3d objects with consistent multi-view generation and 3d diffusion. In *CVPR*. 10072–10083.

Minghua Liu, Minhyuk Sung, Radomir Mech, and Hao Su. 2021. Deepmetahandles: Learning deformation meta-handles of 3d meshes with biharmonic coordinates. In *CVPR*. 12–21.

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. 2015. SMPL: A Skinned Multi-Person Linear Model. *ACM TOG* 34, 6 (Oct. 2015), 248:1–248:16.

Shitong Luo and Wei Hu. 2021. Diffusion probabilistic models for 3d point cloud generation. In *CVPR*. 2837–2845.

Tiange Luo, Chris Rockwell, Honglak Lee, and Justin Johnson. 2024. Scalable 3d captioning with pretrained models. *NeurIPS* 36 (2024).

Zhongjin Luo, Shengcai Cai, Jinguo Dong, Ruibo Ming, Liangdong Qiu, Xiaohang Zhan, and Xiaoguang Han. 2023. RaBit: Parametric Modeling of 3D Biped Cartoon Characters with a Topological-consistent Dataset. In *CVPR*.

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

Thalmann Magnenat, Richard Laperrière, and Daniel Thalmann. 1988. Joint-dependent local deformations for hand animation and object grasping. In *Proceedings of Graphics Interface'88*. Canadian Inf. Process. Soc, 26–33.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Commun. ACM* 65, 1 (2021), 99–106.

Albert Mosella-Montoro and Javier Ruiz-Hidalgo. 2022. Skinningnet: Two-stream graph convolutional neural network for skinning prediction of synthetic characters. In *CVPR*. 18593–18602.

Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. 2022. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751* (2022).

Dafei Qin, Jun Saito, Noam Aigerman, Thibault Groueix, and Taku Komura. 2023. Neural face rigging for animating and retargeting facial meshes in the wild. In *SIGGRAPH*. 1–11.

Ruoxi Shi, Hansheng Chen, Zhuoyang Zhang, Minghua Liu, Chao Xu, Xinyue Wei, Linghao Chen, Chong Zeng, and Hao Su. 2023. Zero123++: a single image to consistent multi-view diffusion base model. *arXiv preprint arXiv:2310.15110* (2023).

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML*. 2256–2265.

Chaoyue Song, Jianfeng Zhang, Xiu Li, Fan Yang, Yiwen Chen, Zhongcong Xu, Jun Hao Liew, Xiaoyang Guo, Fayao Liu, Jiashi Feng, et al. 2025. MagicArticulate: Make Your 3D Models Articulation-Ready. *arXiv preprint arXiv:2502.12135* (2025).

Hongcheng Song, Dmitry Kachkovski, Shaimaa Monem, Abraham Kassauhun Negash, and David IW Levin. 2024. Automatic Skinning using the Mixed Finite Element Method. *arXiv preprint arXiv:2408.04066* (2024).

Jiaming Song, Chenlin Meng, and Stefano Ermon. 2021. Denoising diffusion implicit models. In *ICLR*.

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

Anything World. 2024. Animate Anything. https://app.anything.world/animation-rigging.

Yuefan Wu, Zeyuan Chen, Shaowei Liu, Zhongzheng Ren, and Shenlong Wang. 2022. Casa: Category-agnostic skeletal animal reconstruction. *NeurIPS* 35 (2022), 28559–28574.

Li Xu, Haoxuan Qu, Yujun Cai, and Jun Liu. 2024a. 6d-diff: A keypoint diffusion framework for 6d object pose estimation. In *CVPR*. 9676–9686.

Runsen Xu, Xiaolong Wang, Tai Wang, Yilun Chen, Jiangmiao Pang, and Dahua Lin. 2024b. Pointllm: Empowering large language models to understand point clouds. In *ECCV*. Springer, 131–147.

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

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

Wang Yifan, Noam Aigerman, Vladimir G Kim, Siddhartha Chaudhuri, and Olga Sorkine-Hornung. 2020. Neural cages for detail-preserving 3d deformations. In *CVPR*. 75–83.

Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. 2022. Point-BERT: Pre-training 3d point cloud transformers with masked point modeling. In *CVPR*. 19313–19322.

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. In *NeurIPS*, Vol. 36.

Jieming Zhou, Tong Zhang, Zeeshan Hayder, Lars Petersson, and Mehrtash Harandi. 2023. Diff3DHPE: A Diffusion Model for 3D Human Pose Estimation. In *ICCV*. 2092–2102.