

# Make-It-Animatable: An Efficient Framework for Authoring Animation-Ready 3D Characters

## Supplementary Material

### S1. Formulation of Low-Rank Dynamics

As discussed in Sec. 3.1, modeling the dynamics of an object typically requires a per-timestamp deformation of all the particles (*i.e.*, vertices for meshes, points for point clouds, and splats for 3D Gaussians) that make up its geometry. Suppose that we have an object composed of  $N$  particles and a desired dynamic sequence of  $T$  timestamps. The temporal deformations of all particles can be represented by a matrix  $\mathbf{D} \in \mathbb{R}^{T \times N \times d}$ , where  $d$  is the degrees of freedom (*e.g.*,  $d = 6$  for rigid transformations). Generally,  $\mathbf{D}$  has a lot of redundancy when representing real-world dynamics. Therefore, we would like to seek a low-rank approximation of it. Mathematically, the low-rank decomposition of a matrix like  $\mathbf{D}$  has two different forms expressed by

$$\mathbf{D}^{T \times N \times d} \approx \mathbf{B}_t^{T \times K \times d} \mathbf{W}_s^{K \times N}, \quad (\text{S1})$$

$$\mathbf{D}^{T \times N \times d} \approx \mathbf{B}_s^{K \times N \times d} \mathbf{W}_t^{T \times K}, \quad (\text{S2})$$

where  $K$  is the desired rank. We call the matrix  $\mathbf{B}$  with dimension  $d$  as *basis*, and the other one  $\mathbf{W}$  as *weight*. Eqs. (S1) and (S2) both decouple the temporal and spatial dimensions by splitting them into basis and weight. We then name the ones with temporal dimension ( $T$ ) as *temporal basis*  $\mathbf{B}_t$  and *temporal weight*  $\mathbf{W}_t$ . Correspondingly, the matrices with spatial dimension ( $N$ ) are named as *spatial basis*  $\mathbf{B}_s$  or *spatial weight*  $\mathbf{W}_s$ .

In fact, both decomposition forms have been playing important roles in the applications of dynamic modeling. For example, Eq. (S1) is applied in the sparse control paradigms [4, 6, 19] and the linear blend skinning (LBS) [14, 16] algorithm, where  $\mathbf{B}_t$  is interpreted as the transformations/poses of control nodes/body joints, and  $\mathbf{W}_s$  is the blend weights of the nodes or joints. When applying Eq. (S2),  $\mathbf{B}_s$  and  $\mathbf{W}_t$  are interpreted as the blend shapes and their corresponding weights for the per-timestamp linear combination [3, 7, 8, 23].

In this work, we focus on the low-rank form of Eq. (S1) in modeling dynamic characters. The reason is twofold. First, the temporal weight  $\mathbf{W}_t$  in Eq. (S2) typically requires as much motion data as possible for one object to find a capacious low-rank space, which is prohibitively difficult in practice. In contrast, the spatial weight  $\mathbf{W}_s$  can be supervised much more easily by the well-defined blend weights in existing 3D models. Second, considering the availability of rich motion resources [1, 11] and powerful motion generation methods [2, 5], there is no need to model the temporal basis  $\mathbf{B}_t$  from scratch. What we have to do is

finding a transformation from the input character pose to a pre-defined rest pose, and then any desired animation is within easy reach.

Note that theoretically, joint/bone positions are only proxies or interfaces for the low-rank terms and are not indispensable to the dynamic modeling, as expressed by Eq. (S1). Consequently, some related works [19] choose to model such proxies in an implicit way. However, we still include the bones as one of the desired animation assets in our work for a self-contained and artist-friendly representation compatible with existing animating pipelines. Furthermore, the explicit existence of bones can bring much convenience when applying body priors to assist the optimization, as introduced in Sec. 3.4.

### S2. Implementation Details

#### S2.1. Coarse-to-Fine Shape Representation

To address the limitations of the proposed framework and boost the performance, we introduce an additional design at the input side of the autoencoder, *i.e.*, the coarse-to-fine shape representation (Sec. 3.3). In addition to the overall framework presented in Fig. 2, we include two separate pipelines in Fig. S1 here for a clearer illustration of the coarse and fine training stages. In the following content, we will provide more details of this part, particularly regarding the motivations and implementation choices.

With the particle-based shape autoencoder, our method can already produce fairly good results of blend weights. However, the joint outputs are still unsatisfying in fine-grained regions like the hands. Meanwhile, the pose prediction can hardly converge. We attribute these issues to the ambiguity of the input points and treat the lite training process (taking uniformly sampled points as input and only predicting bone positions, as shown in Fig. S1 upper) as a coarse stage that provides rough but valuable localization information about the input character. To be specific, we exploit some of the coarse joint locations to gain a finer shape representation by applying two different strategies, *i.e.*, the canonical transformation and the hierarchical sampling.

**Canonical transformation.** Although we have normalized the input shapes to align their scales, they still differ in global transformations. Since the poses we want to predict are relative to the fixed origin, the same pose can have huge numerical differences under different coordinate systems. This will lead to a dramatically increased difficulty in pose prediction. Therefore, we move and rotate the entire

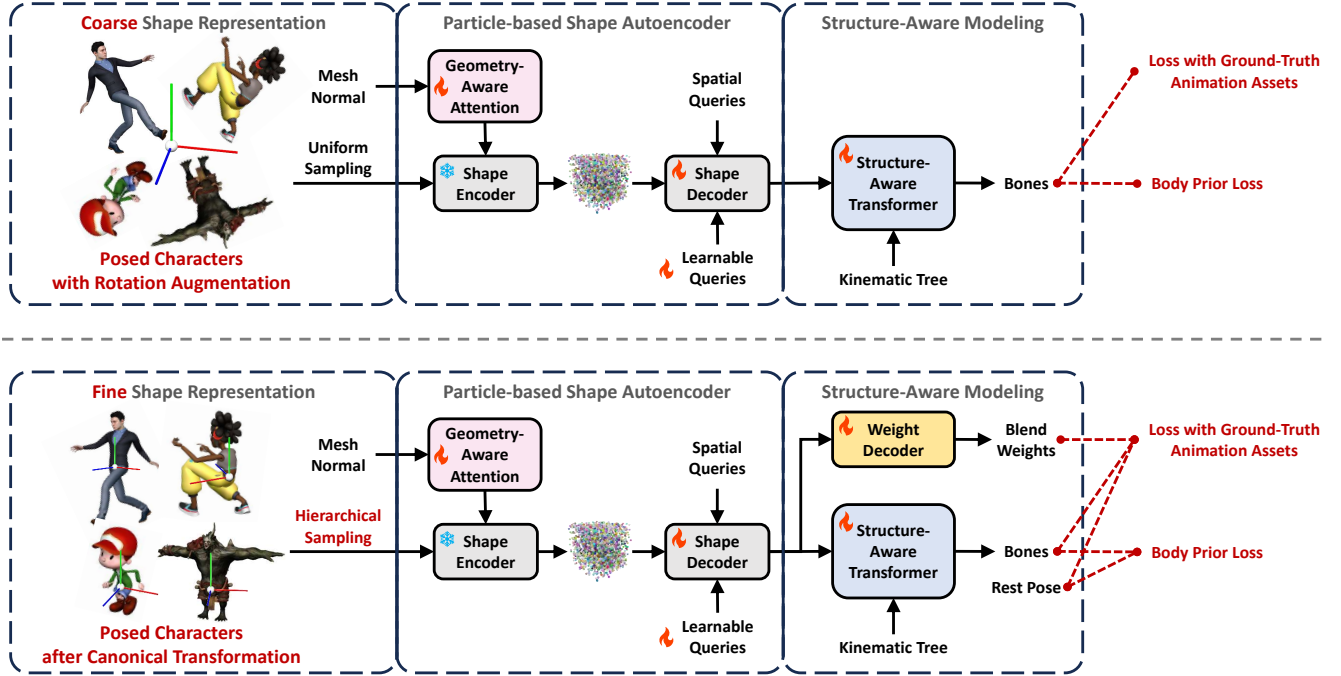


Figure S1. **The coarse (upper) and fine (lower) stages of training our framework.** In the coarse stage, the input shape is uniformly sampled and only the bone positions are predicted. We apply data augmentation to the inputs via random 3D rotations, so that the coarse model is generalizable to global transformations of in-the-wild cases with an acceptable accuracy. In the fine stage, we apply canonical transformation and hierarchical sampling to the shapes in advance based on the ground-truth bone positions. Then during inference, a 3D character is fed into the coarse framework to get its bone positions, which guide the establishment of coarse-to-fine shape representation later in the fine framework. Note that the body prior loss (Sec. 3.4) is directly applied to the bone positions. As for pose prediction, we take the ground-truth bones as a proxy and use the predicted pose to transform them, thereby indirectly affecting the pose optimization.

shape to a canonical position and orientation before sampling. In practice, we choose this transformation based on an empirically selected datum plane (referred to as the *hip plane*) determined by three joint positions, *i.e.*, the hip (root of the kinematic tree) and two upper thighs, which can be accurately located even in the coarse stage. The canonical transformation is applied so that: 1) the hip is located at the origin; 2) the normal of the hip plane is aligned with the z-axis; 3) the vector from the right to the left thigh is parallel to the x-axis. This process simplifies the input shape’s spatial distribution and eliminates the pose representation’s ambiguity. Furthermore, the chosen canonical transformation ensures a consistent upright and front-facing orientation of the input character, which is a common prerequisite of many auto-rigging methods [7, 15, 21]. By integrating the coarse localization, we now automate this preprocessing step, enabling our framework to effectively handle the inputs regardless of their initial spatial configuration (positions, rotations, and scales).

**Hierarchical sampling.** Some parts of the input character, *e.g.*, the hands, present fine-grained details within small regions, which typically demand additional resources for accurate processing. However, the uniform sampling applied

to the input shape, as well as the subsequent farthest point sampling (FPS) algorithm, usually results in sparsely distributed sample points on hands, which are far from enough to describe the geometry of fingers. Theoretically, increasing the number of sampling points  $N$  and the downsampling number  $M$  can bring a larger representation capacity for the entire geometry including the hand regions, but that will significantly add to the computational overhead. For efficiency, we choose to keep the total sampling number unchanged and instead leverage the coarse prediction of hand joints. Specifically, we replace the uniform sampling with a hierarchical approach that ensures a designated proportion of sample points are distributed on both hands. Since the FPS algorithm in the shape encoder disrupts the non-uniformity of samples, we adapt it to a hierarchical algorithm as well.

## S2.2. Networks and Hyperparameters

For the shape autoencoder, we use a point cloud of size  $N = 32768$  as the input. All the points are sampled on the surface of the input mesh. In the fine training or the inference stage equipped with the hierarchical sampling, 50% of the points are uniformly sampled and the other 50% are

sampled near the hand joints. The hyperparameters of the shape autoencoder are consistent with the original setting of 3DShape2VecSet [22], which internally uses shape latents  $\mathbf{F} \in \mathbb{R}^{M \times C}$  with  $M = 512$  and  $C = 512$  for the neural field. Note that in [22],  $\mathbf{F}$  can either be learnable embeddings or initialized by applying farthest point sampling (FPS) on the input point cloud. We choose the latter implementation in this work. During training, the learning rate is linearly increased to  $1e-4$  within the first 1% iterations (warm-up), and then gradually decreased using the cosine decay schedule until reaching the minimum value of  $1e-5$ .

### S2.3. Body Prior Losses

The implementation of the body prior losses (introduced in Sec. 3.4) involves: 1) defining a prior-based bone pair list, and 2) applying  $L_1$ -based loss to their predicted positions or directions. For example, to encourage bone connectivity, we define a list of bone pairs that should be connected based on the predefined topology, select their predicted head-tail positions, and penalize their pair-wise  $L_1$  distances.

## S3. Data Details

### S3.1. Mixamo Dataset

To obtain sufficient training data of 3D characters with high-quality geometry and animation assets, we collect the artist-designed 3D models from Mixamo [1] to form a dataset, which comprises texture meshes of 95 exquisite characters (each has an average of 15000 vertices), along with 2453 diverse motion sequences (each has an average of 200 frames). All the characters are preprocessed to share a standard skeleton structure with  $K = 52$  bones (the leaf bones are removed). For characters with non-standard skeletal structures originally, the blend weights of those bones are transferred to their topologically nearest ancestral bones within the standard skeleton. If some standard bones are missing in a character (e.g., armless person), we mask their corresponding values (weight channels and bone-wise attributes) in loss computing. We use 95% of the data for training and the remaining 5% for validation. During each training iteration, we randomly choose a character-motion pair to get the corresponding animation assets, resulting in an effective dataset size equivalent to over 40 million frames. In Fig. S2, we show some example characters and motions from the Mixamo dataset.

### S3.2. VRoid Dataset

Our training framework can also be extended to different skeleton topologies. To demonstrate this capacity, we use VRoid Studio [17] to manually create 35 different anime characters (30 for training and 5 for validation) with additional accessories including two rabbit-like ears and a fox-like tail, as shown in Fig. S3. These characters differ in



Figure S2. Some samples from the collected Mixamo [1] dataset. The dataset contains bipedal humanoid with different shapes, ranging from realistic humans to cartoon or fantasy creatures. Each character is preprocessed to be animatable by any of the motion sequences. The proposed framework is trained on this dataset.

body shape, clothes, hair style, and the shape of ears and tails. They are all preprocessed to share the same skeleton definition of Mixamo [1] but with extra bones of the accessories. Therefore, they can also be animated with any of the Mixamo motion sequences during the extra-bone training (i.e., fine-tuning the final layer of the weight decoder and the extra learnable queries, based on the standard-skeleton model pretrained on the Mixamo dataset). The experiments show that with our framework, 30 training characters are sufficient to obtain a good model that produces promising predictions for extra bones.

## S4. More Results

### S4.1. Analysis of Efficiency

We compare the inference speed with inputs of different vertex numbers in Fig. S9. Our method maintains sub-second time cost even with 10,000+ vertices, achieving 1000x and 100x speedups over RigNet [21] and TARig [15] respectively. While TARig accelerates RigNet’s iterative process through feed-forward regression, we further introduce a particle-based shape encoder that bypasses mesh connectivity processing while achieving bet-



Figure S3. **Some samples of the anime characters with additional accessories for the extra-bone fine-tuning.** These characters are all manually created using VRoid Studio [17] and then preprocessed to be compatible with the standard skeleton definition of Mixamo [1].

ter performance. This architecture also enables point sub-sampling, making our approach particularly efficient for high-resolution inputs.

## S4.2. Additional Comparison Results

**More comparison cases.** In addition to Figs. 4 and 6, we exhibit more cases of comparison with the baselines, including Meshy [13] and Tripo [18] (Fig. S4), TADA [9] and HumanGaussian [12] (Fig. S5). The results can prove the effectiveness, robustness, and generalizability of the proposed framework.

**Comparison with more auto-rigging methods.** For comparison with existing auto-rigging methods, we include more cases and compare with two more baselines here in Fig. S6 (in addition to Fig. 5), *i.e.*, Neural Blend Shapes [7] and TARig [15]. Neural Blend Shapes only supports T-pose inputs and has quite limited generalizability to shapes different from the SMPL mesh. TARig’s skeleton predictions are better than Neural Blend Shapes and RigNet [21], but its blend weights still cannot meet the standard of practical application, producing spatial unsmoothness when deforming the meshes. Besides, all three baselines are unable to produce fine-grained hand bones, while our method handles the fingers well.

Furthermore, the commercial software, Mixamo [1] and Anything World [10], rely much on the symmetry or pose

simplicity (*e.g.*, T-pose and A-pose) of the inputs and will raise errors when faced with complex ones. Therefore, we compare them separately on some additional cases here in Fig. S7. It can be observed that Anything World produces significant errors when extracting the bones of the left arm for the character with a tail (left case). Meanwhile, Mixamo fails to distinguish the left and right sides of the ninja (right case) and produces a mirrored skeleton. Moreover, when faced with a non-rest input character like this ninja, the predicted pose-to-rest transformations of Mixamo and Anything World both suffer from unnatural deformations, while our results remain good.

**Qualitative comparison on the ModelsResource dataset [20].** We also evaluate the proposed framework on the bipedal-humanoid subset of the “ModelsResource-RigNetv1” dataset [20]. Fig. S8 shows some cases for qualitative comparison with RigNet [21] and TARig [15]. Both of these two baselines are trained on the aforementioned dataset, but our model has never encountered a similar data distribution during training. Despite this, our method still achieves the best quality in rigging and skinning, demonstrating its strong generalizability.

## S4.3. Additional Visualizations

**Tricky cases.** We show more results produced by our method, focusing on the details and some tricky cases. Each sub-figure of Fig. S10 proves some advantages of our methods, which is interpreted one by one as follows.

**(a) Fine-grained control of fingers.** Thanks to the coarse-to-fine shape representation (Sec. 3.3), our method shows remarkable accuracy at the hand regions, which is difficult for most existing approaches.

**(b) Capacity of unusual shapes.** For those characters with an exaggerated body ratio (*e.g.*, extremely large head, short limbs, etc), our method can adaptively change the bone length to fit the shape. This is intractable for template-based human-generating works.

**(c) Complex input poses.** Poses far from the T/A-pose are fully supported. Our model can not only produce the well-fitted posed skeleton, but can also transform it into the T-pose for further animating applications.

**(d) Efficiency for high polygon models.** Benefiting from the particle-based shape autoencoder (Sec. 3.2), our framework is efficient and robust for different input resolutions, ranging from low-poly meshes like (c) to practical game-level 3D models like (d). The Wukong model in (d) has over 1 million triangular faces, but our method can still make it animatable within 3 seconds.

**(e) Support for asymmetric inputs.** While many rigging methods assume symmetric inputs, our method can effectively deal with asymmetric ones. For example, the big gun of this cyborg in (e) is bound well to his arm bones.

**(f) Adaptation to non-existing bones.** This armless statue



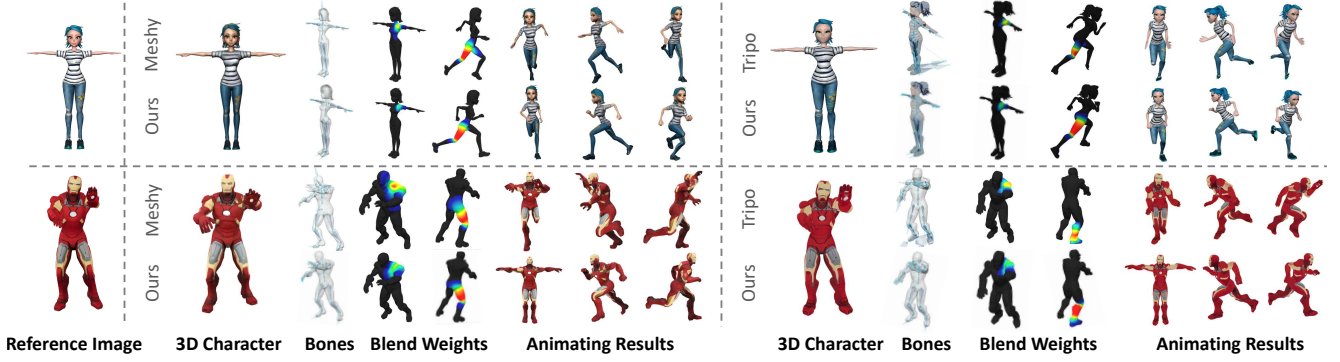


Figure S4. **Additional Comparison with generative 3D methods, i.e., Meshy [13] and Tripo [18].** We feed them the same image as reference and compare the performance based on their generated 3D models respectively. The blend weights of two joints, i.e., Left Shoulder and Right Leg, are visualized. Given that these baselines can only apply preset motions and their rest-pose models cannot be exported, we apply a similar “running” sequence to all the methods for fair comparison. For non-rest cases, the T-pose models predicted by our method are included as the front-view animating results. Zoom in to better view the details.

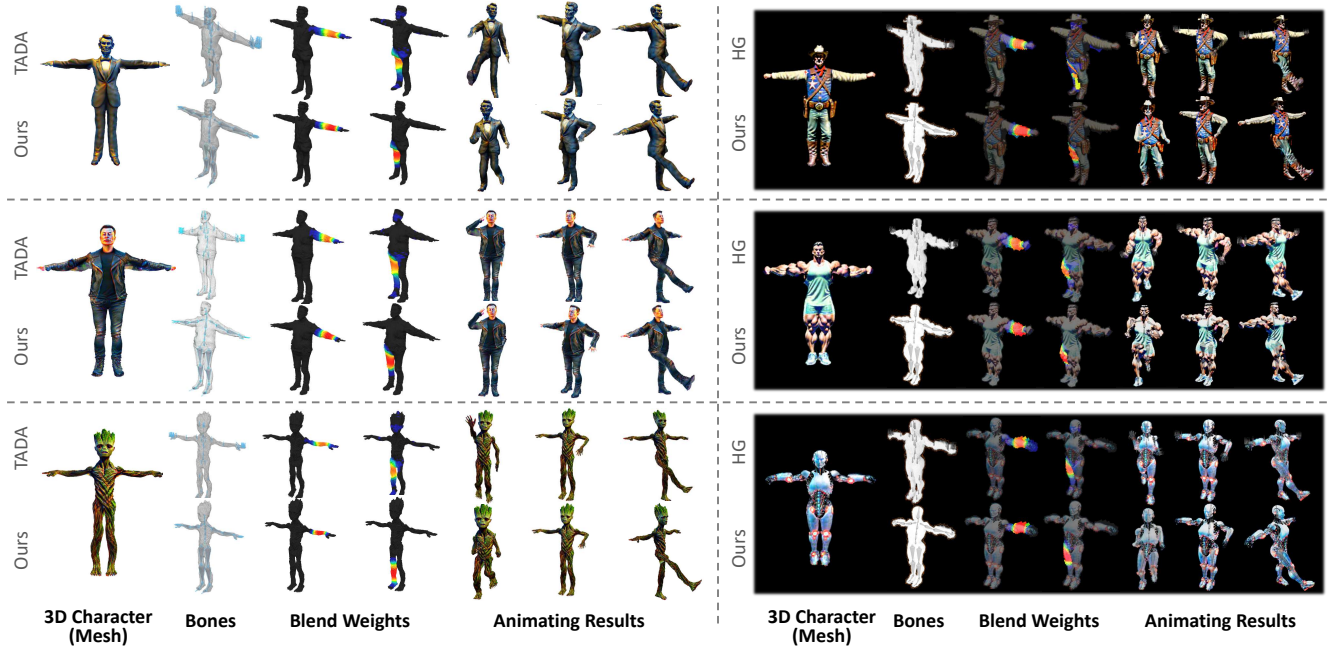


Figure S5. **Additional comparison with template-based avatar generation methods, i.e., TADA [9] and HumanGaussian [12] (HG).** We use the generated meshes from TADA and 3D Gaussians from HG for comparison. Note that the skeletons of these two baselines are identical to the shape-specific SMPL [14] templates (without bone tail), with their weights interpolated from the template meshes. Zoom in to better view the details.

is predicted to have extracorporeal arm bones. However, due to the spatial query mechanism of our framework, those dummy bones have no weights toward the actual vertices and can be simply removed without any side effects.

**(g) & (h) Extension to extra bones.** Once fine-tuned with some extra data (Sec. 3.5), our model is capable of producing positions and weights of non-standard bones. Here we show the long tail of a monkey in (g) and the long ears of a bunny in (h), which are all fully animatable.

**Geometry-awareness: attention vs. adding.** Intuitively, simply concatenating or adding the normals to the coordinates when feeding points into the shape encoder can achieve the same goal as our geometry-aware attention. However, we found in practice that such a vanilla injecting strategy often leads to overfitting on the high-quality training mesh normals. The weight prediction depends so much on normals that some regions are influenced by far-away bones just because they have plausible normals, es-



Figure S6. **Additional comparison with auto-rigging methods, i.e., RigNet [21], TARig [15], and Neural Blend Shapes [7] (Neural BS).** We visualize the blend weights of selected joints and manually deform them to assess the impact of rigging quality on skinning results. \*: Neural Blend Shapes only support T-pose inputs, so for the non-rest cases (lower two), we feed it the T-pose meshes transformed by our pose-to-rest predictions.

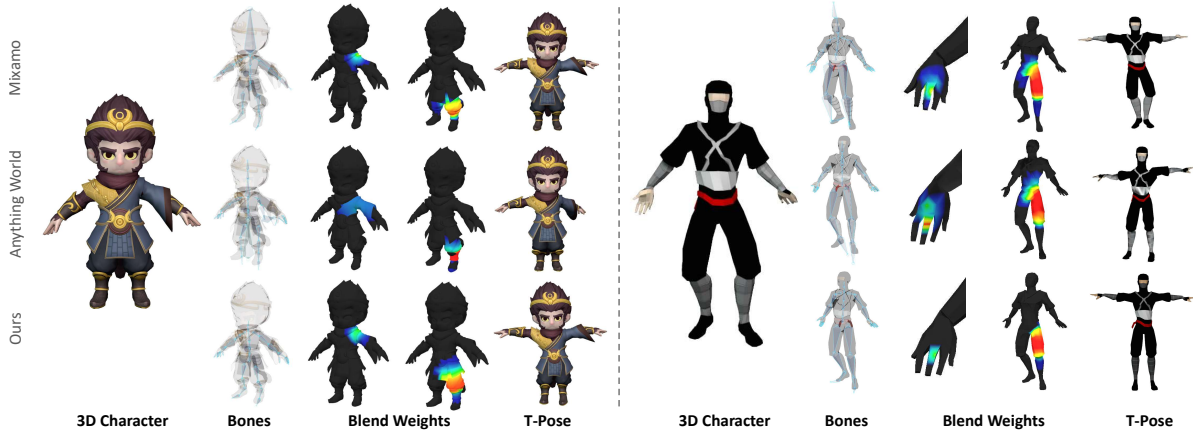


Figure S7. **Comparison with commercial auto-rigging software, i.e., Mixamo [1] and Anything World [10].** Note that these two tools can only deal with simple input poses (T- or A-pose is recommended) and often raise errors when faced with complex ones.

pecially when faced with lower-quality meshes (*e.g.*, produced by generative models). We present a typical case in Fig. S11. While injecting normal information via adding resolves the weight corruption problem in the armpit region, it introduces a new issue of incorrect weights on hands. In contrast, the proposed attention mechanism benefits from

normal information without any side effects.

**Geometry-aware attention score.** In addition to the two exemplar cases in Fig. 7, here we visualize the geometry-aware attention score of more characters in Fig. S12. The distribution of high-attention-score points shows patterns with statistical significance. Specifically, regions with pos-

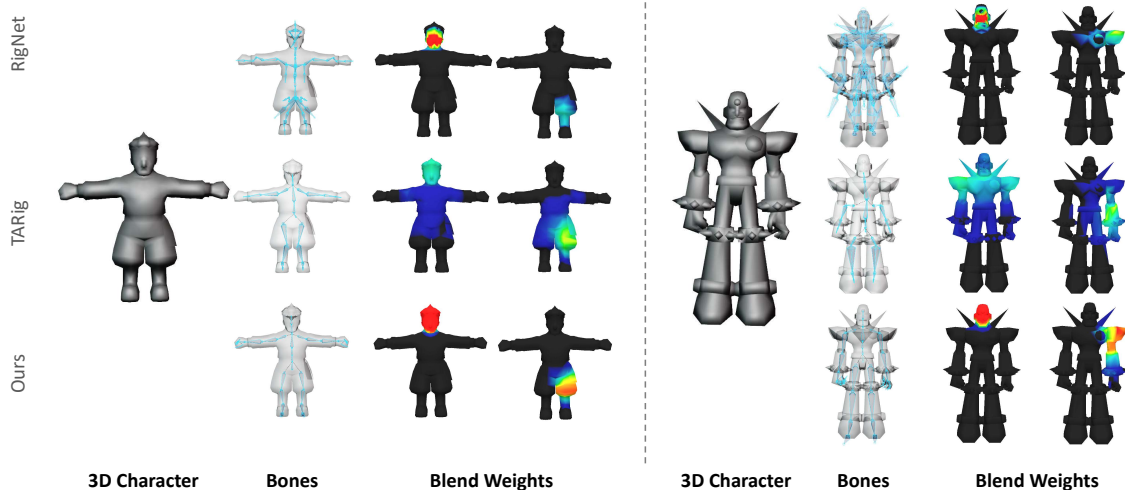


Figure S8. **Qualitative comparison with RigNet [21] and TARig [15] on cases from the test split of “ModelsResource-RigNetv1” dataset [20].** While both baselines are exactly trained on this dataset and ours are not, we still achieve the best performance.

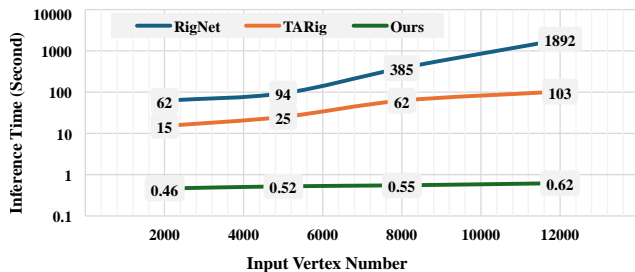


Figure S9. **Comparison of the inference time cost with increasing number of vertices as the input.**

sible spatial ambiguity, *e.g.*, inner thigh, armpit, between ears, *etc.*, are affected more by the normal values. This verifies the effectiveness of our geometry-aware attention, as it works exactly in the desired way to adaptively exploit the normal information.

**Limitations of SMPL-based rigging.** Template-based avatar generation methods [9, 12] benefit a lot from the well-defined SMPL mesh [14, 16], enabling the generation of animatable characters with no additional rigging cost. However, as discussed in Secs. 1 and 4.2, one of the unneglectable limitations of these methods is their inability to depart from realistic human shapes. Although TADA [9] attempts to address this issue by predicting vertex deformations of the template mesh, it remains constrained by the preset body ratio of SMPL. Fig. S13 demonstrates some practical examples of cartoon characters with exaggerated body shapes that the SMPL model can hardly accommodate. As illustrated in the left part of Fig. S13, the large heads of these characters cannot be fitted by the template meshes, and the interpolated blend weights will definitely be incorrect in the head regions. In the right part of Fig. S13, we exhibit the results generated by two template-

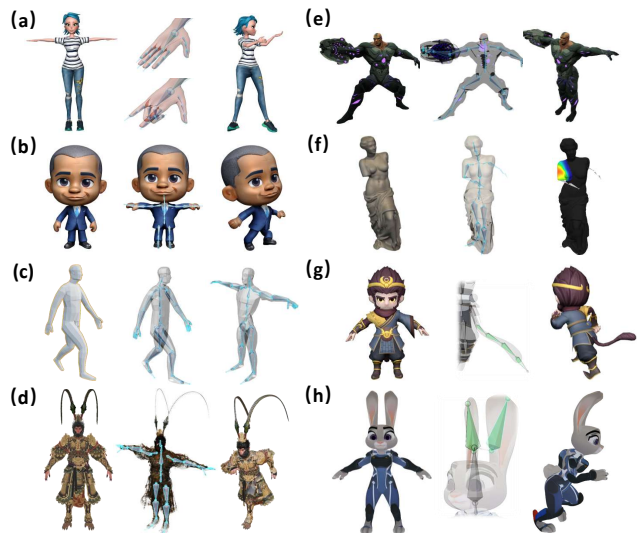


Figure S10. **Results of more tricky cases to demonstrate the advantage of our method.** (a) Fine-grained control of fingers; (b) Capacity of unusual shapes; (c) Complex input poses; (d) Efficiency for high polygon models; (e) Support of asymmetric inputs; (f) Adaptation to non-existing bones; (g) & (h): Extension to extra bones (*e.g.*, long ears and tails).

based methods using the same text prompt “cartoon brown bear with extremely large head and short legs”. Despite the specific prompting, both methods still produce body ratios resembling those of realistic humans, which limits their applicability in practical scenarios. In contrast, our method offers a promising solution by making bipedal characters of any shape ready for animation.

**More versatile pose inputs.** It is worth noting that many of our evaluations above are based on inputs close to rest



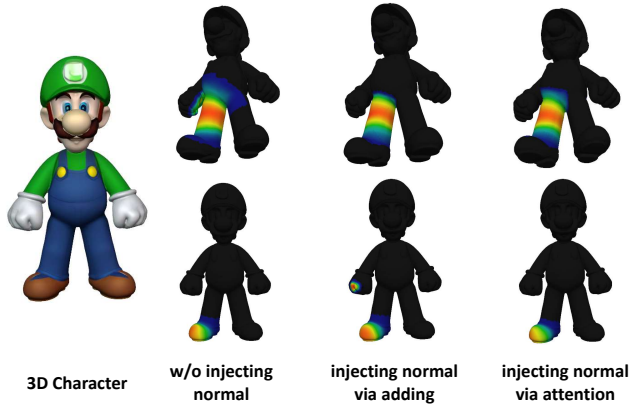


Figure S11. **Qualitative analysis of our geometry-aware attention module and its injecting method.** The proposed attention-based injection can benefit from normal information without any side effects.

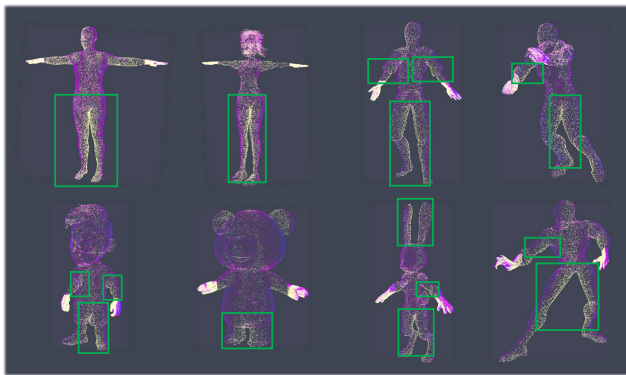


Figure S12. **Visualization of the attention score of our geometry-aware attention module.** These per-sampled-point values are extracted from the first attention head (out of 8 heads in total). The brighter color (yellow) indicates more attention to normals rather than coordinates. We also use green bounding boxes to label some clusters where high-attention-score points are densely distributed. It can be observed that the module adaptively learns to rely more on normals in regions like the inner thigh since coordinates become less discriminative there.

poses because the baselines are not specifically designed to handle versatile pose inputs. Specifically, RigNet [21] and TARig [15] cannot transform arbitrary poses into rest poses. Mixamo [1] and Anything World [10] raise failure without inputs close to A- or T-pose. Meshy [13] and Tripo [18] perform poorly on non-rest poses (see Fig. 4 and Fig. S4). To show our method’s generalizability of poses, we provide more results with non-rest inputs in Fig. S14 (2).

**Failure cases.** Fig. S14 (1) presents some failure cases of our method. Limitations involving challenging poses and out-of-distribution shapes could be resolved by including more targeted training data. As for the topological defects of input meshes, incorporating other methods to refine the

3D geometry might be a promising solution.

## S5. Videos for Dynamic Visualization and Practical Applications

For better visualization, we provide a video file as part of the supplementary material, which includes the following content:

- An illustration of our inference pipeline and the data flow.
- The showcase of 3D characters and their animating results enabled by our framework. Inputs represented by mesh and 3D Gaussian Splats are both included.
- The process of using our web demo to make 3D characters animatable with one click.
- A real-world application that brings a character figure to live by integrating our framework with an off-the-shelf image-to-3D generator.

## References

- [1] Adobe. Mixamo, 2024. <https://www.mixamo.com>. 1, 3, 4, 6, 8
- [2] Xin Chen, Biao Jiang, Wen Liu, Zilong Huang, Bin Fu, Tao Chen, and Gang Yu. Executing your commands via motion diffusion in latent space. In *CVPR*, pages 18000–18010, 2023. 1
- [3] Devikalyan Das, Christopher Wewer, Raza Yunus, Eddy Ilg, and Jan Eric Lenssen. Neural parametric gaussians for monocular non-rigid object reconstruction. In *CVPR*, pages 10715–10725, 2024. 1
- [4] Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. SC-GS: Sparse-controlled gaussian splatting for editable dynamic scenes. In *CVPR*, 2024. 1
- [5] Biao Jiang, Xin Chen, Wen Liu, Jingyi Yu, Gang Yu, and Tao Chen. MotionGPT: Human motion as a foreign language. *NeurIPS*, 36, 2024. 1
- [6] Jiahui Lei, Yijia Weng, Adam Harley, Leonidas Guibas, and Kostas Daniilidis. MoSca: Dynamic gaussian fusion from casual videos via 4D motion scaffolds. *arXiv preprint arXiv:2405.17421*, 2024. 1
- [7] Peizhuo Li, Kfir Aberman, Rana Hanocka, Libin Liu, Olga Sorkine-Hornung, and Baoquan Chen. Learning skeletal articulations with neural blend shapes. *ACM TOG*, 40(4):1–15, 2021. 1, 2, 4, 6
- [8] Tianye Li, Timo Bolkart, Michael J. Black, Hao Li, and Javier Romero. Learning a model of facial shape and expression from 4D scans. *ACM TOG*, 36(6):194:1–194:17, 2017. 1
- [9] Tingting Liao, Hongwei Yi, Yuliang Xiu, Jiaxiang Tang, Yangyi Huang, Justus Thies, and Michael J. Black. TADA! Text to animatable digital avatars. In *3DV*, pages 1508–1519, 2024. 4, 5, 7
- [10] Anything World Limited. 3D animation and automated rigging | Anything World, 2024. <https://anything.world>. 4, 6, 8



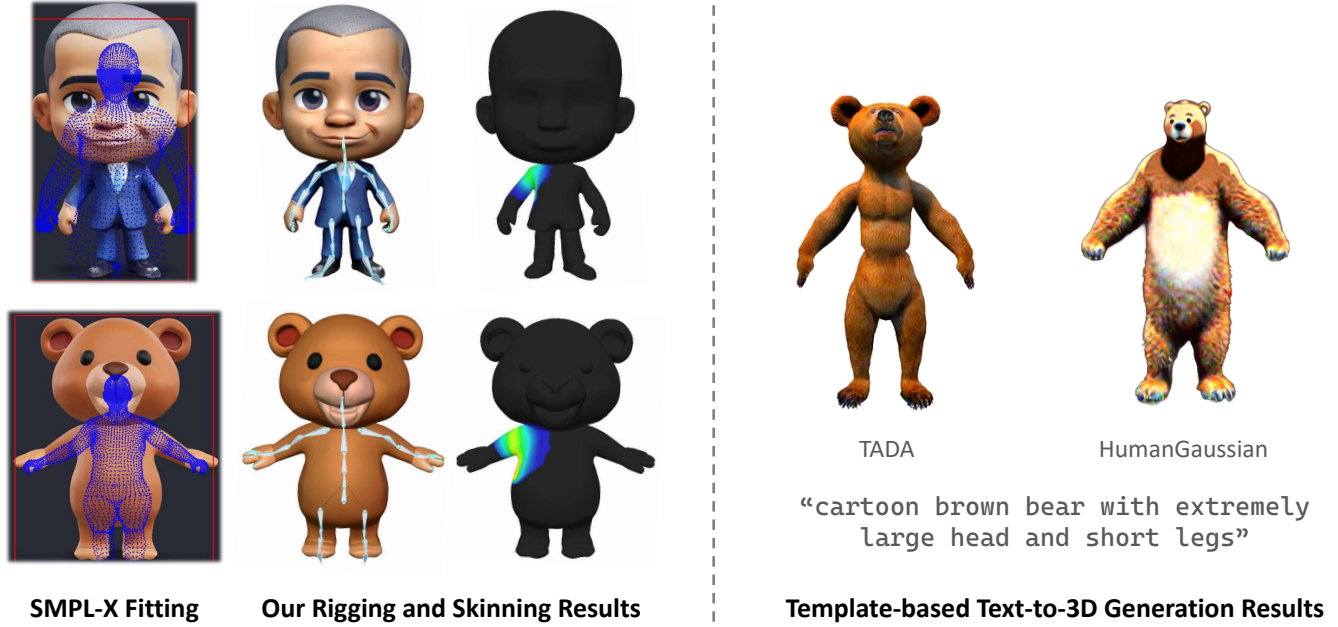


Figure S13. **Illustration of the limitations of SMPL-based rigging.** While SMPL provides a good template for skeletons and weights, it lacks the flexibility to handle exaggerated body shapes.

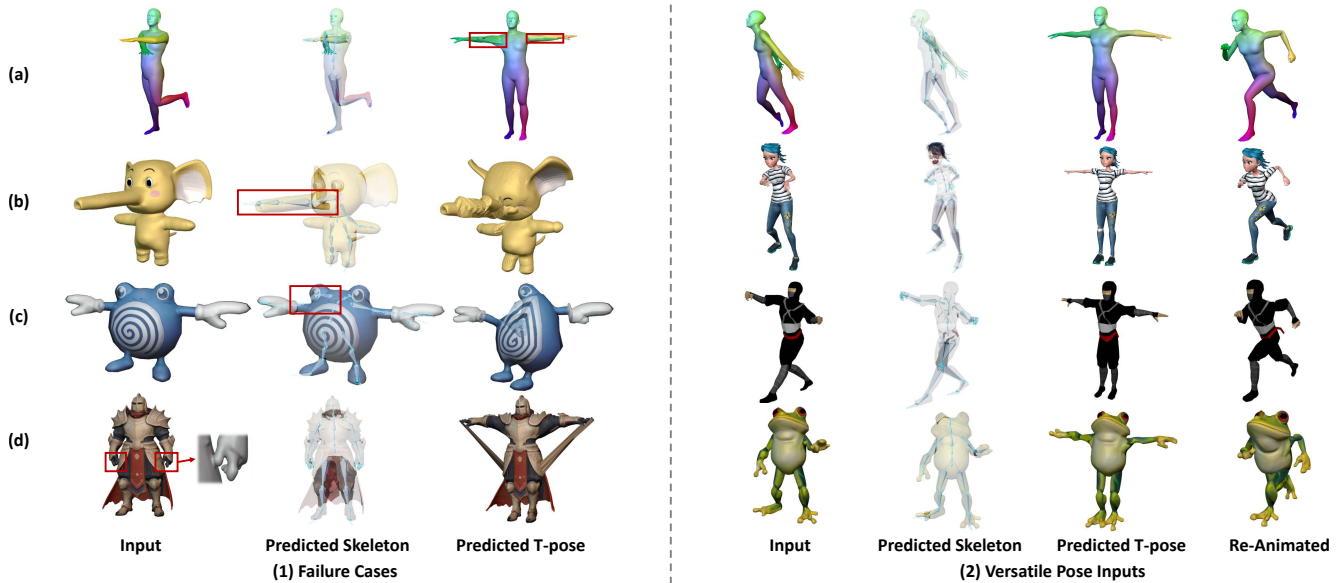


Figure S14. **(1) Failure cases of our method.** (a) Challenging pose with self-intersection; (b) & (c) Out-of-distribution shapes; (d) Topological defects (unexpected mesh connections). **(2) More results produced by our method with non-rest-pose inputs.**

[11] Jing Lin, Ailing Zeng, Shunlin Lu, Yuanhao Cai, Ruimao Zhang, Haoqian Wang, and Lei Zhang. Motion-X: A large-scale 3D expressive whole-body human motion dataset. *NeurIPS*, 2023. 1

[12] Xian Liu, Xiaohang Zhan, Jiayang Tang, Ying Shan, Gang Zeng, Dahua Lin, Xihui Liu, and Ziwei Liu. HumanGaussian: Text-Driven 3D Human Generation with Gaussian Splatting. In *CVPR*, pages 6646–6657, 2024. 4, 5, 7

[13] Meshy LLC. Meshy - convert text and images to 3D models, 2024. <https://www.meshy.ai>. 4, 5, 8

[14] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM TOG*, 34(6):248:1–248:16, 2015. 1, 5, 7

[15] Jing Ma and Dongliang Zhang. TARig: Adaptive template-aware neural rigging for humanoid characters. *Computers &*

*Graphics*, 114:158–167, 2023. [2](#), [3](#), [4](#), [6](#), [7](#), [8](#)

- [16] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3D hands, face, and body from a single image. In *CVPR*, pages 10975–10985, 2019. [1](#), [7](#)
- [17] pixiv Inc. VRoid Studio, 2024. <https://vroid.com/en/studio>. [3](#), [4](#)
- [18] Tripo. Tripo AI, 2024. <https://www.tripo3d.ai>. [4](#), [5](#), [8](#)
- [19] Qianqian Wang, Vickie Ye, Hang Gao, Jake Austin, Zhengqi Li, and Angjoo Kanazawa. Shape of motion: 4D reconstruction from a single video. *arXiv preprint arXiv:2407.13764*, 2024. [1](#)
- [20] Zhan Xu, Yang Zhou, Evangelos Kalogerakis, and Karan Singh. Predicting animation skeletons for 3D articulated models via volumetric nets. In *3DV*, pages 298–307. IEEE, 2019. [4](#), [7](#)
- [21] Zhan Xu, Yang Zhou, Evangelos Kalogerakis, Chris Landreth, and Karan Singh. RigNet: Neural rigging for articulated characters. *ACM TOG*, 39(4):58:58:1–58:58:14, 2020. [2](#), [3](#), [4](#), [6](#), [7](#), [8](#)
- [22] Biao Zhang, Jiapeng Tang, Matthias Nießner, and Peter Wonka. 3DShape2VecSet: A 3D shape representation for neural fields and generative diffusion models. *ACM TOG*, 42(4):92:1–92:16, 2023. [3](#)
- [23] Wojciech Zielonka, Timo Bolkart, Thabo Beeler, and Justus Thies. Gaussian eigen models for human heads. *arXiv preprint arXiv:2407.04545*, 2024. [1](#)