DRiVE: Diffusion-based Rigging Empowers Generation of Versatile and Expressive Characters

Supplementary Material

In this supplementary material, we provide a detailed introduction to the construction of our proposed dataset `AnimeRig`, including the preparation of rigging ground truth of mesh and 3D Gaussian data in Sec. 1. In Sec. 2, we elaborate on the method's details, covering input pre-processing, 3D Gaussian optimization, and the design of the bone networks. Finally, in Sec. 3, we include additional experimental details and results.

1. Dataset Construction Details

To the best of our knowledge, no large-scale open-source rigging dataset exists. Existing large 3D object datasets (e.g., Objaverse [4, 5] and OmniObject3D [20]) do not contain enough 3D anime characters. In addition, Anime3D [11] introduced a large-scale 3D character dataset; however, none of the data includes rigging information. We first introduce `AnimeRig`, a large dataset of anime characters from VRoidHub [1] which contains 9,420 rigged textured meshes. In the following, we explain how to process rigged character mesh data and describe how to obtain the corresponding rigging ground truth data based on the 3D Gaussian:

Rigging Ground Truth of Mesh: The data we collected includes textured meshes and their corresponding rigging information. We process the raw data in the following steps: We first filter out non-human shapes, and for data with issues in skeletons or skinning, we enlist the help of animators for corrections. Then for the extracted part mesh data, we combine the hair, clothing, and torso meshes to obtain a complete mesh. For the skeleton data, we remove the fine finger joints but retain the wrist joints to facilitate learning. Correspondingly, we adjust the skinning by transferring the weights of the removed joints to their respective parent joints, resulting in a new skinning configuration.

Rigging Ground Truth of 3D Gaussian: Since the original mesh data differs in positions and scales from the 3D Gaussian generated by LGM [15], we need to transfer the rigging information obtained from the mesh onto the 3D Gaussian. Therefore, we perform the same positional transformation on the joints, keeping the bone connection unchanged. This lets us obtain the ground truth data for the skeleton corresponding to the 3D Gaussian. We define the skinning of each point on the 3D Gaussian as a weighted average of the k nearest neighbors on the corresponding mesh. This design ensures that the 3D Gaussian deforms smoothly when animated. This way, we can obtain the rigging ground truth corresponding to the 3D Gaussian.

2. Methods Details

2.1. Input Pre-processing Details

Text Input: To accommodate text-based input, we fine-tune Animagine-XL on an anime character dataset. To prepare for fine-tuning, we first render front-view images of our dataset using Blender, followed by human body parsing with SegFormer [21]. We utilize BLIP [9] to generate Visual Question Answering (VQA) descriptions [7], appending the phrase "cartoon style" to create a complete image caption. We use a fine-tuned Animagine-XL model with T-pose images as the Openpose ControlNet to generate standard T-pose anime images from text input. Fig. 2 shows the T-pose anime characters generated by the fine-tuned model under different text inputs. Anime image segmentation model uses anime-segmentation [2]. Using the fine-tuned Animagine-XL model in combination with IP-Adapter and OpenPose as ControlNet, we convert anime characters in arbitrary poses to T-pose anime images. This approach demonstrates strong generalization capabilities on cartoon characters and game avatars. Some input images are sourced from examples used in CharacterGen [11].

Real Human Image Input: To accommodate input from real human image, we employ ControlNet [24], integrating OpenPose [1, 2, 14, 19] and IP-Adapter [23], with text prompts from VQA as optional inputs. The first stage involves converting any posed image into T-pose. We use the Realistic Vision[3] series models as the base models for image generation, with real human photos as input to the IP-Adapter and the T-pose as input to OpenPose ControlNet. In the second stage, our fine-tuned model on Animagine-XL serves as the base model for image generation. The generated real human T-pose image is used as input to the IP-Adapter, with the T-pose provided to OpenPose ControlNet. Fig. 4 demonstrates how our pipeline effectively adapts to various input poses, preserving the details and features of the input portraits.

Anime Image Input: To adapt to anime image inputs, we convert non-T-pose anime images into T-pose anime images using a fine-tuned Animagine-XL model along with the previously mentioned real human to T-pose pipeline. Fig. 3 demonstrates how our pipeline effectively adapts anime im-

[1] https://hub.vroid.com/en

[2] https://github.com/SkyTNT/anime-segmentation/

[3] https://civitai.com/models/152525/realism-engine-sdxl

ages in various input poses to T-pose anime outputs, preserving the details and features of the original portraits.

2.2. 3D Gaussian Refinement Details

In this step, we utilize the T-pose anime image as a conditional input for generating the 3D Gaussian representation. When reconstructing from a full-body image input, the results generated by LGM suffer from severe artifacts, leading to a noticeable loss of detail, especially in the head region.

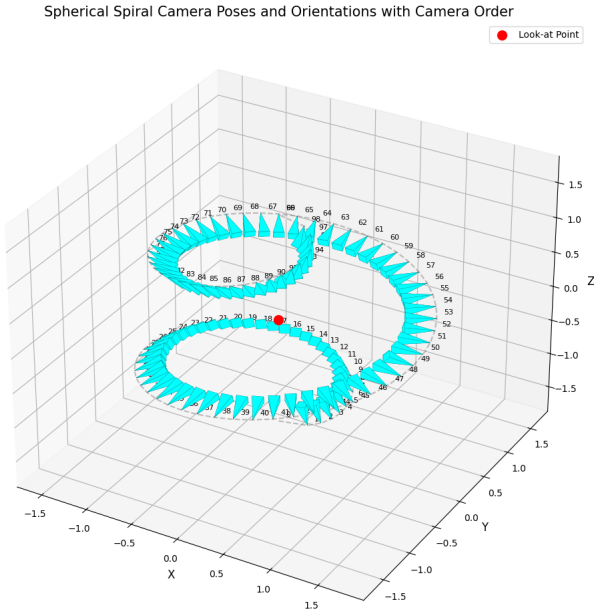

Figure 1. Visualization of cameras arranged in a spherical spiral path, each pointing towards a central look-at point.

To enhance the reconstruction quality of LGM for anime characters, we render 100 full-body T-pose images for each character in the dataset, following a spherical spiral path, and fine-tune LGM with these images. The 100 selected camera viewpoints are shown in Fig. 1.

Subsequently, we decouple the head and body for separate reconstruction, merging them through Gaussian registration to retain more detailed features. To further improve multi-view consistency, we replace the original modules in LGM responsible for generating consistent multi-view images (ImageDream [18] and MVDream [13]) with SV3D [17], as SV3D has demonstrated superior multi-view consistency.

Fig. 5 shows a comparison of the LGM model with and without SV3D as the multi-view generator, as well as before and after fine-tuning.

We crop the head region from the T-pose anime image for separate reconstruction. Using SV3D, we generate continuous new view images at 15° intervals in the horizontal direction, and selected four images at 90° intervals, including the frontal view, as input for LGM. The Gaussian points of the head and the full body obtained from LGM are registered by first applying a known initial alignment based on a fixed cropping frame, followed by refining the registration using the Iterative Closest Point (ICP) algorithm [3].

The registration problem is formulated as an optimization task, where we aim to find the optimal rotation R, translation t, and scaling factors s that minimize the weighted sum of squared distances between the transformed head points and the body points. The initial translation and scaling factors are determined by the size and position of the head crop within the full-body image:

$$E(R,t,s) = \arg\min_{R\in SO(d),t\in\mathbb{R}^d,s\in\mathbb{R}} \sum_{i=1}^{n} w_i \left\| Rsq_i + t - p_i \right\|^2, \tag{1}$$

where p_i and q_i are corresponding points in the body and head points, respectively, and w_i are the associated weights. After the alignment, we merge the head and body points. During this merging step, points from the head Gaussian that may overlap with the body Gaussian are filtered out to ensure a smooth transition between the two parts.

2.3. Bone Connection

In this part, we provide a detailed explanation of how to obtain bone connections based on the predicted joints. The core problem is to estimate the probability of each pair of joints being in a parent-child relationship. Instead of directly learning this probability matrix, we construct a cost matrix based on distance and connection direction dimensions followed by TARig [10]. Since our **GSDiff** model can generate accurate joint positions, we can directly compute the cost matrix and then obtain reasonable bone connectivity using Minimum Spanning Tree (MST) [22].

We use the boneflow $\mathbf{B}$, which consists of normalized direction vectors pointing from each joint to its parent joint, to learn the bone connectivity. We extract the geometric and appearance features of 3D Gaussian following the approach outlined in Sec 4.2 of the main paper and then use an MLP to estimate the boneflow $\hat{\mathbf{B}}$:

$$\hat{\mathbf{B}} = f_b(\mu, \{I_i\}_{i=1}^4; \mathbf{W}_b), \tag{2}$$

where $\mathbf{W}_b$ denotes the learned parameters of the boneflow estimation. The boneflow loss is the MSE between the ground truth boneflow $\mathbf{B}$ and the predicted boneflow $\hat{\mathbf{B}}$.

Based on the boneflow estimated above, we define the following cost function for each pair of predicted joints $\hat{\mathbf{J}}_i$ and $\hat{\mathbf{J}}_j$, where $cost_{ij}$ represents the total cost of traveling from joint i to joint j [10].

$$cost_{i,j} = cost_{euc}(i,j) + \lambda_1 cost_{flow}(i,j),$$
$$= \left|\hat{\mathbf{J}}_j - \hat{\mathbf{J}}_i\right| + \lambda_1 \frac{1}{|\mathbb{C}_i|} \sum_{k \in \mathbb{C}_i} \arccos \hat{\mathbf{B}}_k \cdot \hat{\mathbf{J}}_{ij}, \quad (3)$$

where $\mathbb{C}_i$ represents the set of Gaussian points surrounding joint i and $\hat{\mathbf{J}}_{ij} = (\hat{\mathbf{J}}_j - \hat{\mathbf{J}}_i)/\left|\hat{\mathbf{J}}_j - \hat{\mathbf{J}}_i\right|$. $cost_{euc}(i,j)$ represents the Euclidean distance loss between $\hat{\mathbf{J}}_i$ and $\hat{\mathbf{J}}_j$, while $cost_{flow}(i,j)$ represents the deviation of the bone formed between $\hat{\mathbf{J}}_i$ and $\hat{\mathbf{J}}_j$ from the ground truth direction. A lower cost indicates a higher likelihood of forming a bone between them.

The connections follow a fixed topology for the body joints due to their standardized structure. We then compute the cost matrix between each joint in $\mathbf{J}_o$ and that in $\mathbf{J}$. In our implementation, we use Prim's algorithm [12], where the defined cost matrix serves as the weight matrix. We select the central point along the Y-axis as the root joint of the skeleton and use it as the starting point for Prim's algorithm. This method produces structurally valid skeletons and minimizes the need for post-processing before animation.

3. Experiments

3.1. Implementation details

We provide more detailed experimental specifics for joint prediction. Regarding the training process, the input must be the joints with the same number of points. We do not need to modify the body joints $\mathbf{J}_b$ as their point count is fixed. However, to learn the other joints $\mathbf{J}_o$, we pad the joints to N_1 points to ensure the input has the same number of points. The padding method involves copying noisy points at the $(0,0,0)$ position. During testing, for the generated joints, we treat points that fall within a small range around $(0,0,0)$ as noise points and remove them, using the remaining points as the predicted other joints. We also down-sample the 3D Gaussian using the farthest point sampling (FPS) strategy to the same point number N_2. The training loss is simply the standard diffusion loss followed [8].

Here, we provide the hyperparameters used in the experiments. For joint prediction, we use k-NN search to find $k_1 = 10$ 3D Gaussian points around each joint. We pad the other joint for each shape to $N_1 = 175$, resulting in 200 joints. Meanwhile, we down-sample the 3D Gaussian to $N_2 = 10000$ points. We train our diffusion model for 200 epochs with a batch size of 16. Other parameters related to diffusion training follow the settings in [16]. For bone estimation, we train the DGCCN with a batch size of 16 over 10 epochs. In the cost function, we set $\lambda_1 = 0.1$, and $\mathbb{C}_i$ is composed of the 10 nearest 3D Gaussian points to each joint i. The inclusion of the boneflow cost helps to correct erroneous points caused by proximity. For skinning prediction, we first construct the initial skinning weights $\mathbf{S}_{init}$ as the estimation of the ground truth by k-NN search between the joints $\mathbf{J}$ and the Gaussian points μ. Here we use $k_2 = 2$. Additionally, we train a 3-layer MLP to predict skinning weights and we set the weight of smooth loss $\lambda_2 = 0.01$. Our framework is implemented with PyTorch and on NVIDIA A100 GPU.

3.2. Metircs Descriptions

Here, we present the evaluation metrics used for skeleton and skinning prediction. All the metrics follow the design in RigNet [22]. For skeleton prediction, we use the following metrics:

(a) CD-J2J: The Chamfer distance between joints. We calculate the average symmetric distance between the predicted and the ground truth joints.

(b) CD-J2B: The Chamfer distance between joints and bones. This computes the distance between predicted joints and the nearest bones in the ground truth skeleton, symmetrized in both directions.

(c) CD-B2B: The Chamfer distance between bones, assessing skeleton similarity based on bone placement. Lower values for all these metrics indicate better alignment.

(d) IoU: Intersection over Union is used to evaluate skeleton matching by calculating the proportion of predicted and ground truth joints that fall within a set tolerance, adjusted by the local shape diameter.

(e) Precision & Recall: Precision measures the fraction of predicted joints matching ground truth joints within the tolerance, while recall reflects the fraction of ground truth joints matched to predicted ones.

For skinning evaluation, we compare predicted skinning maps to the ground truth using:

(a) Precision & Recall: Calculated based on the influential bones (bones with skinning weights exceeding a threshold) for each vertex.

(b) L1-norm: The L1 norm of the difference between predicted and ground truth skinning weights across mesh vertices.

3.3. Additional Qualitative Results

In this section, we provide additional visualization results to further demonstrate the robustness of our method. Fig. 2, Fig. 3, and Fig. 4 respectively showcase the results of transforming inputs from text, anime images in arbitrary poses, and real human data in arbitrary poses into T-pose anime images. These results highlight the flexibility of our method in handling multi-modal inputs.

For T-pose anime image inputs, Fig. 5 presents the generation results using the `AnimeRig` dataset to fine-tune LGM, showing significantly higher-quality 3D Gaussians compared to pre-fine-tuned generation. Additionally, Fig. 6

illustrates the results of the 3D Gaussian Refinement introduced in Sec. 2.2. We perform a magnified rendering centered on the head for a more detailed comparison. The results demonstrate that our method achieves a notable enhancement in rendering quality compared to mesh-based methods [11]. Moreover, our 3D Gaussian ICP method effectively improves local rendering quality and can be flexibly adapted to other tasks.

We also provide additional skeleton generation results in Fig. 7 and skinning generation results in Fig. 8, demonstrating that our method can generate reasonable rigging results for characters with different styles. Finally, Fig. 9 and Fig. 10 respectively show the T-pose anime images and rigging results obtained from text and real human data inputs.

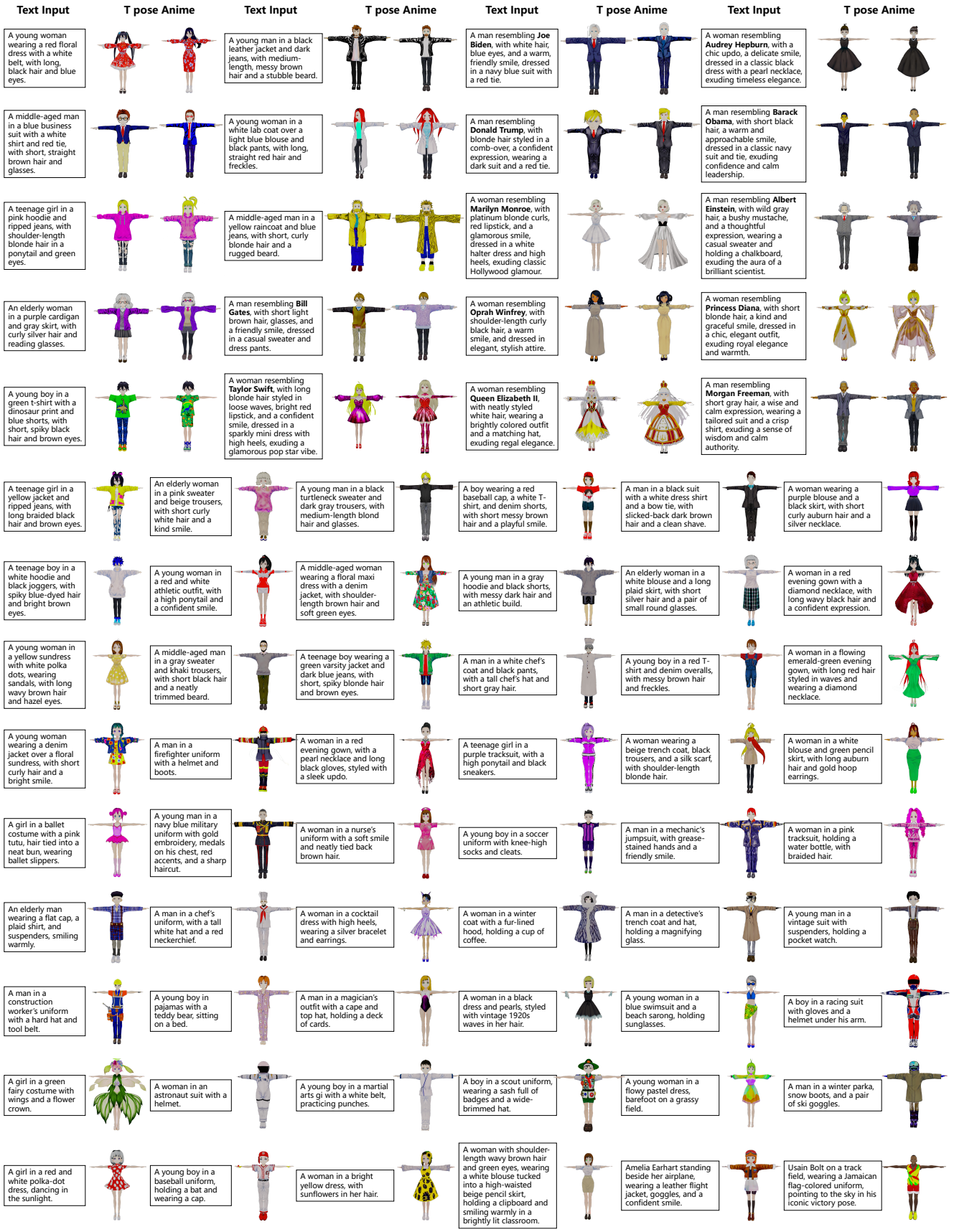

Figure 2. Generating T-pose anime characters from text input also generalizes well when using public figure names as keywords.

Figure 3. Generating T-pose anime characters from any pose anime image input.

Figure 4. Generating any pose real-human image to T-pose anime image.

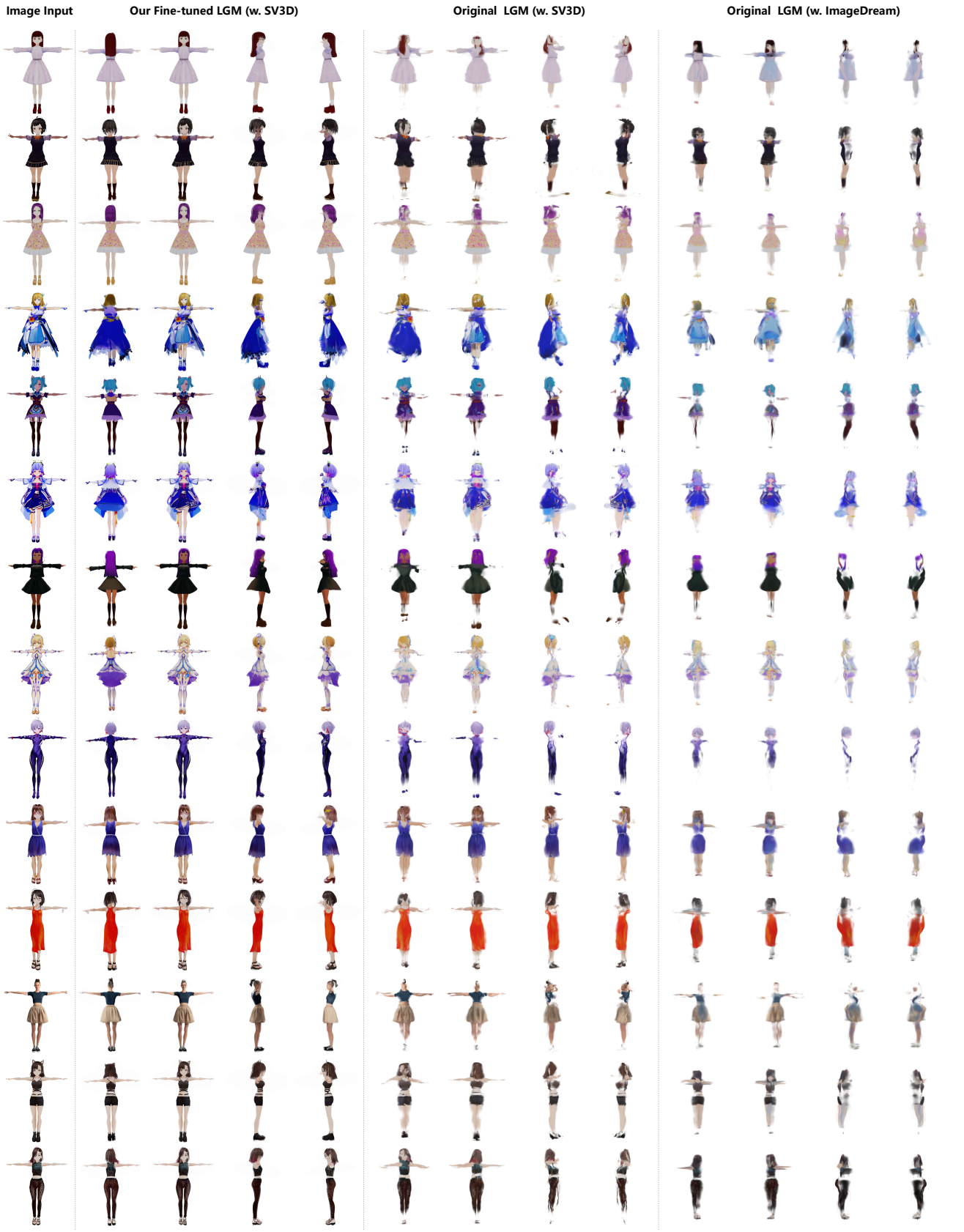

Figure 5. With the same T-pose anime character image input, our fine-tuned LGM model shows a significant improvement in rendering quality.

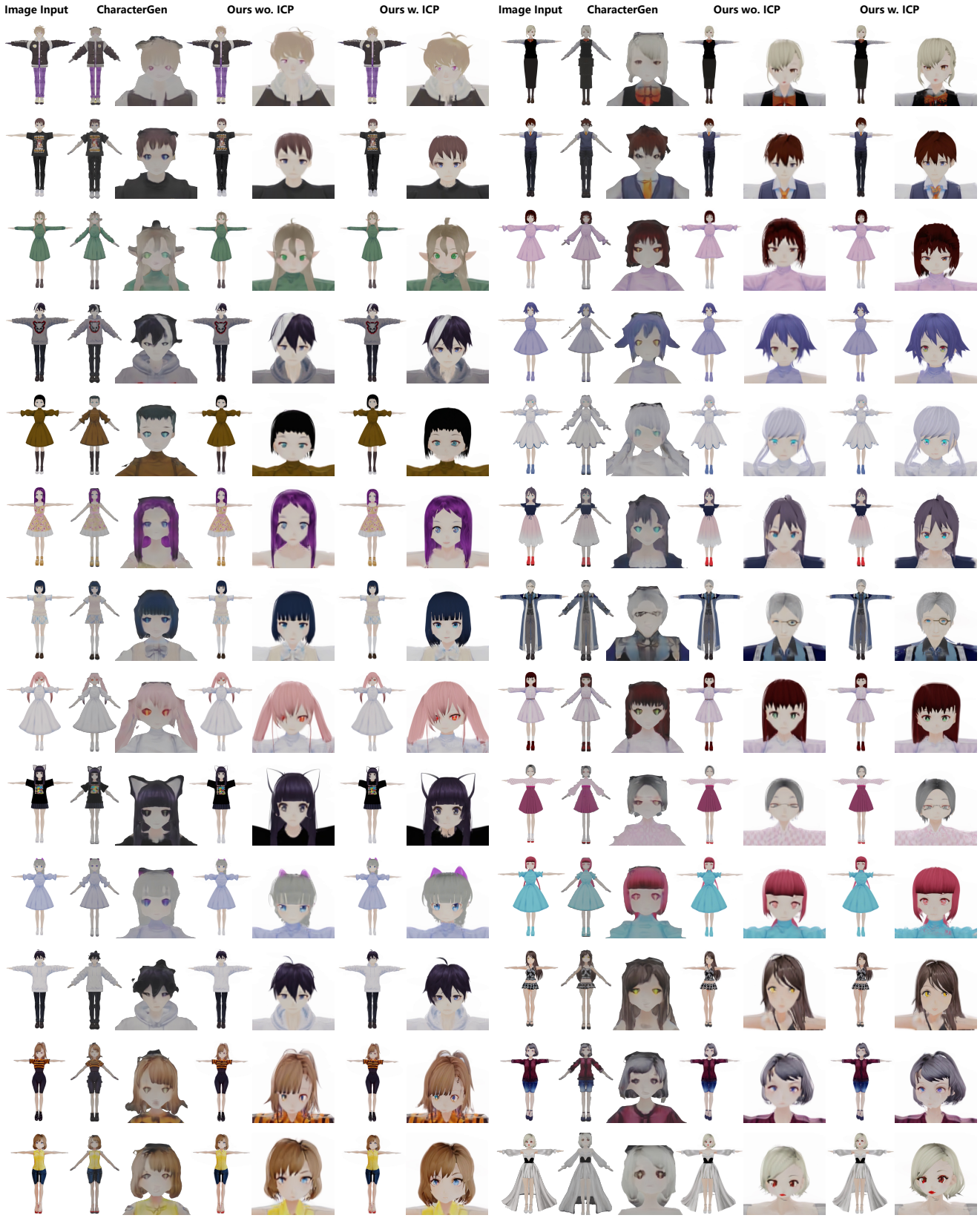

Figure 6. Comparison results with CharacterGen using a single T-pose anime image as input. ”wo. ICP” denotes the results of directly using LGM to reconstruct a single image.

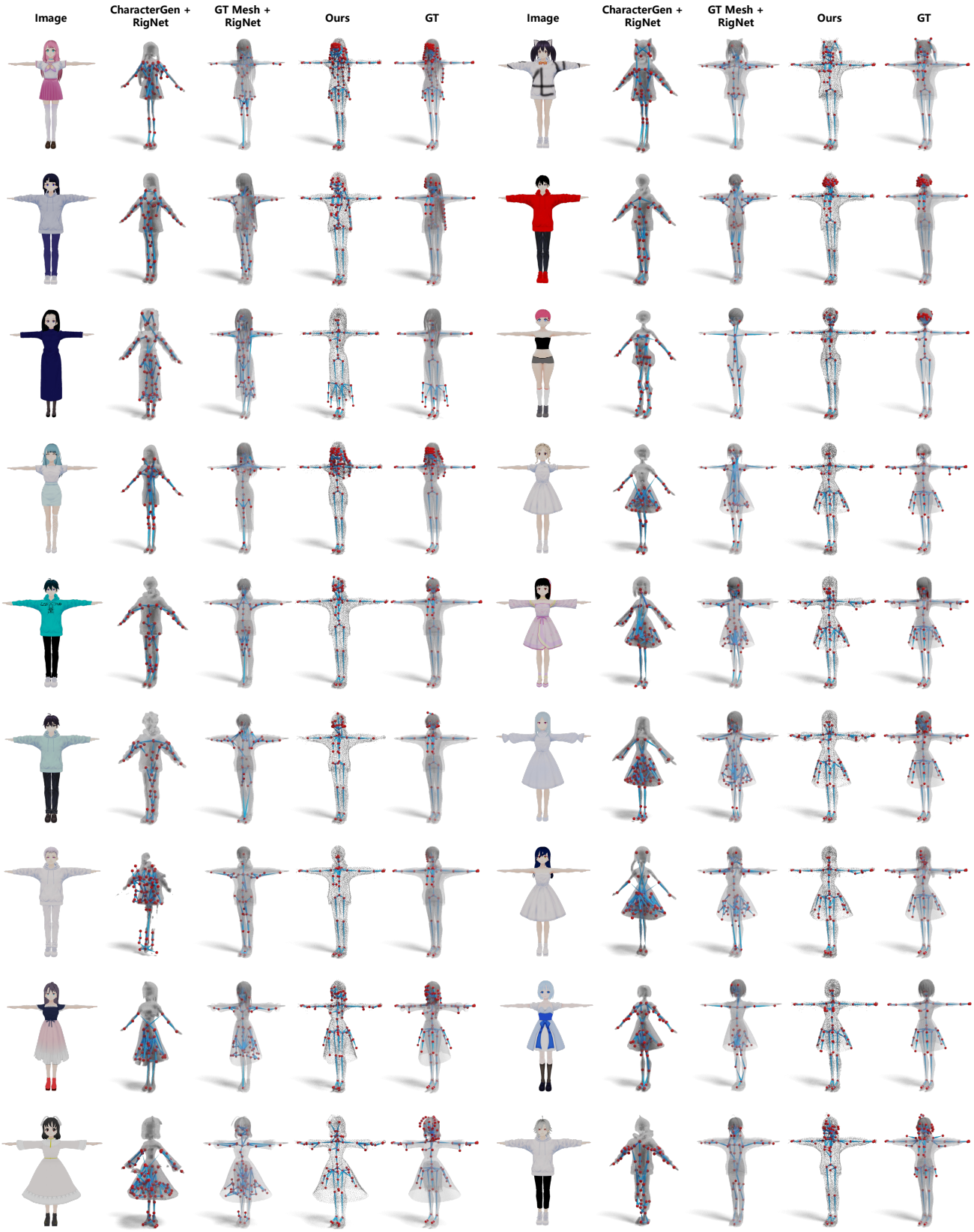

Figure 7. We show more skeleton generation results compared with RigNet [22]. Our model can generate skeletons for clothing and hairstyles of different styles.

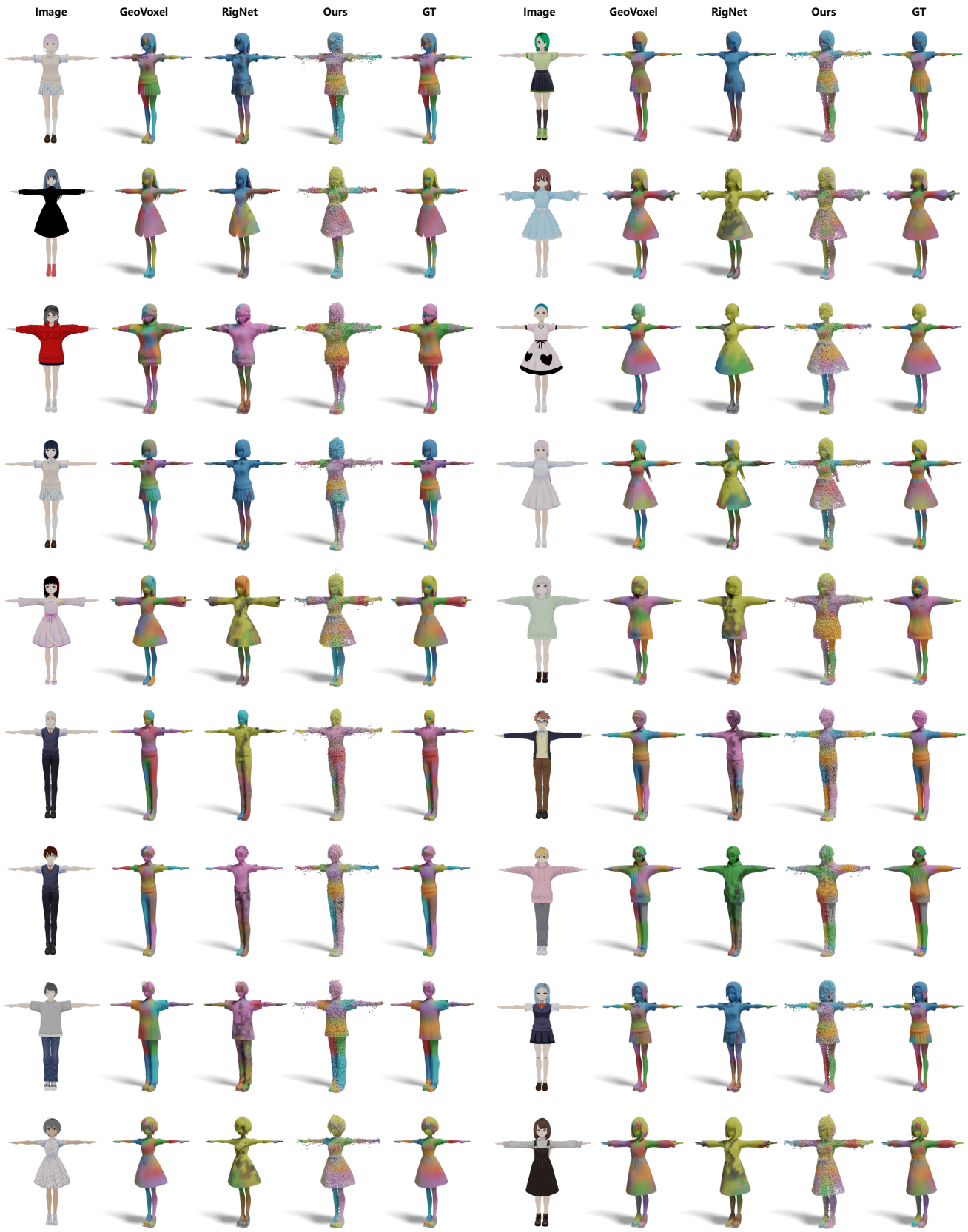

Figure 8. We show more skinning generation results compared with GeoVoxel [6] and RigNet [22].

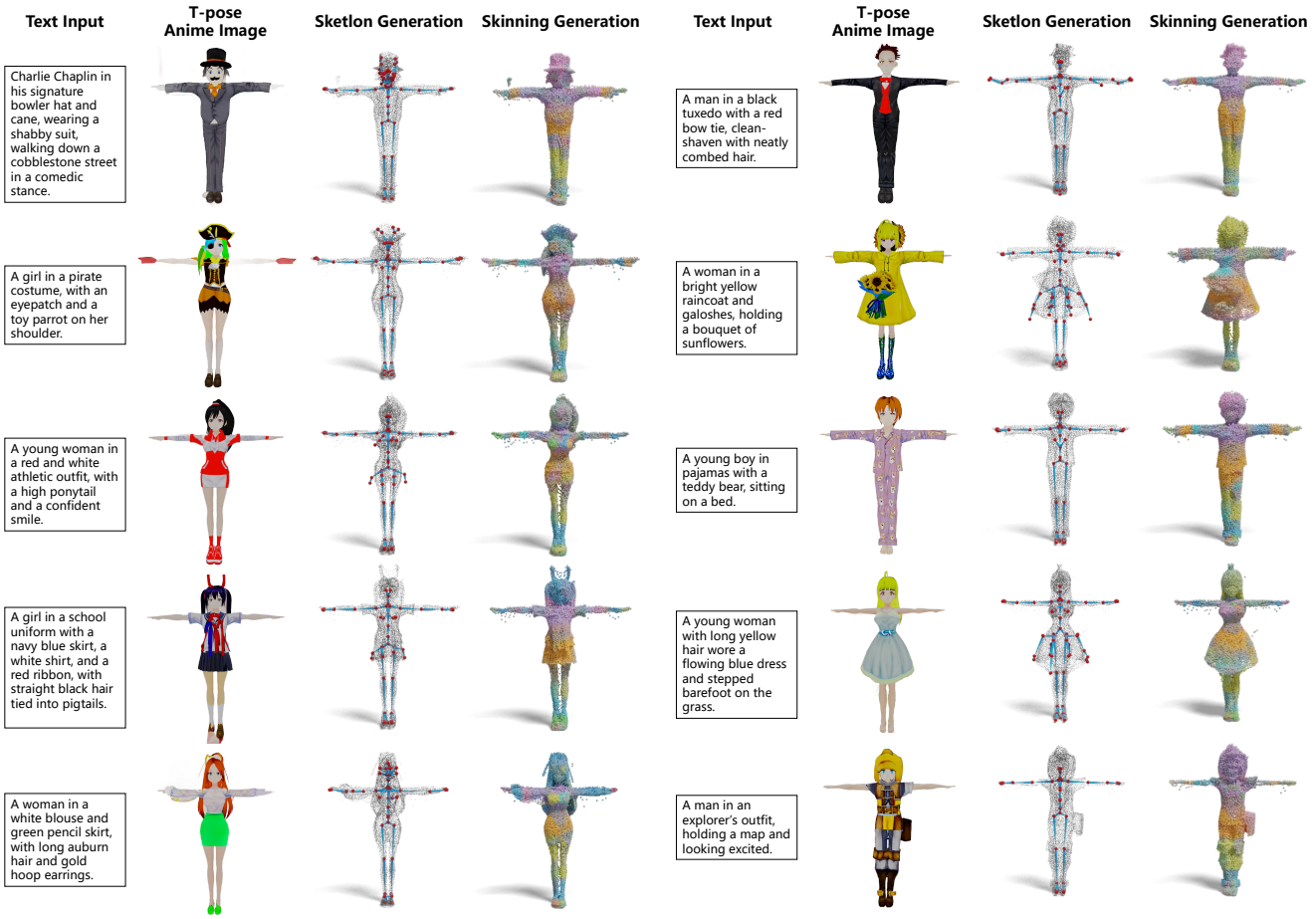

Figure 9. For text input, we first generate a T-pose anime image that is semantically consistent with it, followed by the corresponding 3D Gaussian and rigging results.

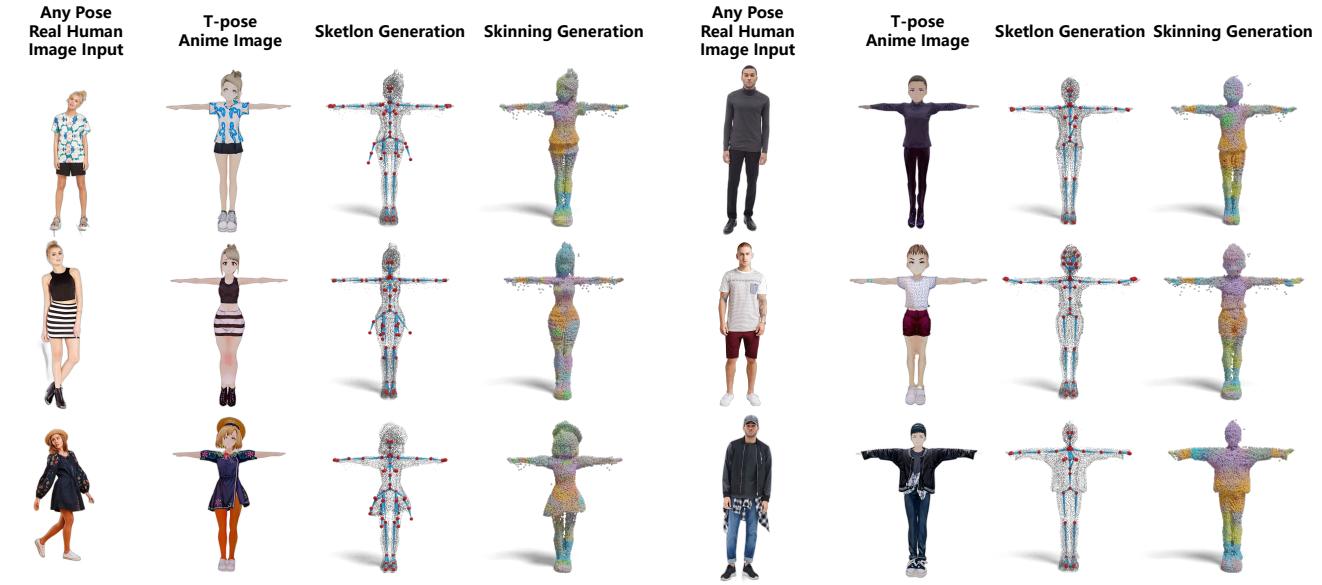

Figure 10. We generate a T-pose anime image for any pose real human image input, and we generate the corresponding 3D Gaussian and rigging results.

References

[1] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, 2017. 1

[2] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. 1

[3] Dmitry Chetverikov, Dmitry Svirko, Dmitry Stepanov, and Pavel Krsek. The trimmed iterative closest point algorithm. In *2002 International Conference on Pattern Recognition*, pages 545–548. IEEE, 2002. 2

[4] Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13142–13153, 2023. 1

[5] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *Advances in Neural Information Processing Systems*, 36, 2024. 1

[6] Olivier Dionne and Martin de Lasa. Geodesic voxel binding for production character meshes. In *Proceedings of the 12th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 173–180, 2013. 11

[7] Yangyi Huang, Hongwei Yi, Yuliang Xiu, Tingting Liao, Jiaxiang Tang, Deng Cai, and Justus Thies. TeCH: Text-guided Reconstruction of Lifelike Clothed Humans. In *International Conference on 3D Vision (3DV)*, 2024. 1

[8] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022. 3

[9] Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In *International conference on machine learning*, pages 12888–12900. PMLR, 2022. 1

[10] Jing Ma and Dongliang Zhang. Tarig: Adaptive template-aware neural rigging for humanoid characters. *Computers & Graphics*, 114:158–167, 2023. 2

[11] Hao-Yang Peng, Jia-Peng Zhang, Meng-Hao Guo, Yan-Pei Cao, and Shi-Min Hu. Charactergen: Efficient 3d character generation from single images with multi-view pose canonicalization. *ACM Transactions on Graphics (TOG)*, 43(4): 1–13, 2024. 1, 4

[12] Robert Clay Prim. Shortest connection networks and some generalizations. *The Bell System Technical Journal*, 36(6): 1389–1401, 1957. 3

[13] Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. Mvdream: Multi-view diffusion for 3d generation. *arXiv preprint arXiv:2308.16512*, 2023. 2

[14] Tomas Simon, Hanbyul Joo, Iain Matthews, and Yaser Sheikh. Hand keypoint detection in single images using multiview bootstrapping. In *CVPR*, 2017. 1

[15] Jiaxiang Tang, Zhaoxi Chen, Xiaokang Chen, Tengfei Wang, Gang Zeng, and Ziwei Liu. Lgm: Large multi-view gaussian model for high-resolution 3d content creation. In *European Conference on Computer Vision*, pages 1–18. Springer, 2024. 1

[16] Michał J Tyszkiewicz, Pascal Fua, and Eduard Trulls. Gecco: Geometrically-conditioned point diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2128–2138, 2023. 3

[17] Vikram Voleti, Chun-Han Yao, Mark Boss, Adam Letts, David Pankratz, Dmitry Tochilkin, Christian Laforte, Robin Rombach, and Varun Jampani. Sv3d: Novel multi-view synthesis and 3d generation from a single image using latent video diffusion. In *European Conference on Computer Vision*, pages 439–457. Springer, 2024. 2

[18] Peng Wang and Yichun Shi. Imagedream: Image-prompt multi-view diffusion for 3d generation. *arXiv preprint arXiv:2312.02201*, 2023. 2

[19] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *CVPR*, 2016. 1

[20] Tong Wu, Jiarui Zhang, Xiao Fu, Yuxin Wang, Liang Pan Jiawei Ren, Wayne Wu, Lei Yang, Jiaqi Wang, Chen Qian, Dahua Lin, and Ziwei Liu. Omniobject3d: Large-vocabulary 3d object dataset for realistic perception, reconstruction and generation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 1

[21] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. In *Neural Information Processing Systems (NeurIPS)*, 2021. 1

[22] Zhan Xu, Yang Zhou, Evangelos Kalogerakis, Chris Landreth, and Karan Singh. Rignet: neural rigging for articulated characters. *ACM Transactions on Graphics (TOG)*, 39 (4):58–1, 2020. 2, 3, 10, 11

[23] Hu Ye, Jun Zhang, Sibo Liu, Xiao Han, and Wei Yang. Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models. *preprint arXiv:2308.06721*, 2023. 1

[24] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3836–3847, 2023. 1