AnimateAnyMesh: A Feed-Forward 4D Foundation Model for Text-Driven Universal Mesh Animation Supplementary File

Anonymous ICCV submission

Paper ID 5509

1. Settings

In this section, we elaborate on the technical details of AnimateAnyMesh. The chapter is structured into three main components: data curation 1.1, implementation details 1.2, and evaluation metric 1.3.

1.1. Data Curation

As described in the main paper, our curated 4D data is sourced from three primary sources: Objaverse [3, 9], AMASS [8], and DT4D [5]. Initially, we filter and extract all `.glb` files containing animation sequences from Objaverse. Using Blender's Python API (bpy), we convert each animation into a mesh sequence. Animations with fewer than 16 frames are discarded, and each sequence is capped at 200 frames. Post-conversion, each animation is encoded into a `.bin` file comprising $D \subset \{F \in \mathbb{R}^{M\times 3}, V \in \mathbb{R}^{T\times N\times 3}\}$, where M denotes the face count, T represents the temporal length of the dynamic mesh sequence, and N indicates the vertex count. Similarly, we develop scripts to convert SMPL [7] models from AMASS and `.anime` files from DT4D into the identical `.bin` format.

Subsequently, we traverse all stored files, implementing vertex merging operations for duplicate vertices while updating the corresponding face information. This serves two purposes: data optimization and, crucially, supporting DyMeshVAE's encoding process, which embeds vertex connectivity information to prevent trajectory inconsistencies during decoding.

The processed dynamic mesh files undergo temporal slicing with window sizes $T = 16$ and $T = 32$. To maximize data utilization, we initiate slicing from both frame 0 and $T//2$, storing T-frame segments sequentially. We also preserve reverse-ordered sequences as independent files, effectively augmenting the dataset by 3-4$\times$. Each new sequence undergoes center normalization, positioning the initial frame at the origin with maximum vertex absolute values normalized to 1.0.

Post-slicing, we implement motion-based filtering, eliminating sequences with inter-frame maximum absolute differences outside the range [0.01, 0.5]. We also filter out the instances whose faces/vertices ratio exceeds 2.5. The cleaned data is then rendered using bpy scripts to generate frontal video sequences. We apply uniform gradient coloring (purple-red) and consistent top-down point lighting, utilizing the Cycles engine for 256×256 resolution rendering on CPU clusters.

For caption generation, we employ Qwen-2.5-VL [1] as our annotation model with the prompt: "Describe the motion of the object in a sentence." The generated captions are stored alongside their corresponding `.bin` files.

Finally, we validate all processed files, removing examples with anomalous vertex or face shapes. The resulting DyMesh dataset is partitioned into subsets based on maximum vertex counts (4,096/8,192/50,000) to facilitate training and testing across different configurations.

1.2. Implementation Details

In the DyMeshVAE architecture, both encoder and decoder utilize attention mechanisms with a hidden dimension of 512. For temporal settings of T=16 and T=32, we employ latent dimensions of 32 and 64 channels respectively for VAE sampling, with both configurations containing approximately 25M parameters. The Shape-Guided Text-to-Trajectory Model consists of 12 stacked transformer blocks as shown in Fig. 4 of the main paper, where each block incorporates 8-head attention layers with features projected to 512 dimensions, totaling approximately 200M parameters. We also conduct scaling experiments with an enhanced architecture of 740M parameters, comprising 24 transformer blocks, 16 attention heads per layer, and a 1024-dimensional latent space. During training, we implement an efficient batchify strategy where each sample's vertices and faces are padded to maintain uniform tensor dimensions across the batch: vertex tensors are padded with zero vectors (0.0, 0.0, 0.0) up to the dataset's maximum vertex count, while face indices are padded with invalid indices

ICCV #5509

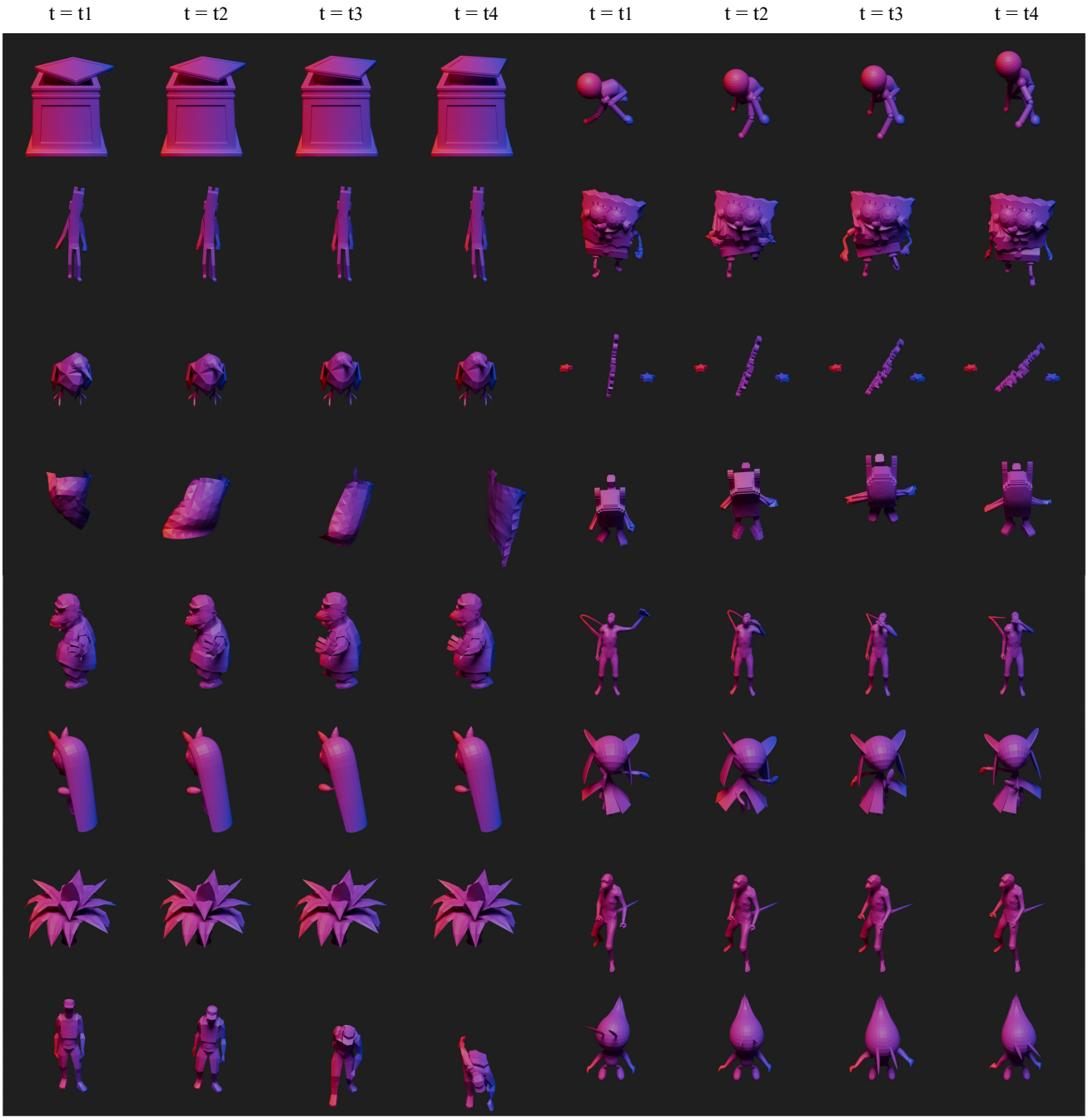

Figure 1. Examples of dynamic mesh sequences in our DyMesh dataset.

(-1, -1, -1) to reach the maximum face count (defined as 2.5 times the maximum vertex count), enabling consistent batch processing while preserving mesh topology integrity. During inference, we perform rectified flow sampling using 64 uniformly sampled timesteps within the [0,1] interval.

1.3. Evaluation Metrics

For quantitative evaluation, we employ three standardized metrics from VBench [4] to assess the performance of comparative methods: I2V Subject Consistency, Motion Smoothness, and Aesthetic Quality. Specifically, I2V Subject Consistency is computed by measuring the frame-wise similarity of DINO [2] features, quantifying the visual co-

num_v	4,096	8,192	16,384	32,768
t (s)	3.95	5.99	10.68	21.86

Table 1. Inference time evaluation. num_v represents the number of mesh vertices. We sample 1/8 number of vertices in the FPS sampling procedure as default. All these testing is conducted on a single Nvidia A800 GPU.

herence between the generated video and the reference image. Motion Smoothness is evaluated through the AMT [6] video interpolation framework, which assesses the temporal continuity and fluidity of the generated motion sequences. The Aesthetic Quality metric leverages the LAION aesthetic predictor to quantify the perceptual quality and artistic value of individual frames from a human-centric perspective. These complementary metrics provide a comprehensive evaluation of both temporal consistency and visual fidelity of the generated results.

For our user study protocol, we recruit 20 participants from diverse backgrounds and age groups to evaluate comparative methods through a controlled assessment. We randomly selected 10 diverse test cases and generated motion sequences for each using all comparative methods based on text prompts, rendering each result from four orthogonal viewpoints and concatenating them temporally ($16\times4 = 64$ frames). The results from all methods were randomly shuffled and horizontally concatenated into 64-frame GIFs, with participants rating each result on a 5-point Likert scale (5: excellent, 1: poor) across three criteria: text-motion alignment, motion plausibility, and shape preservation fidelity. The final evaluation scores were computed by aggregating and de-shuffling ratings across all participants, with failed generations being handled by computing means from successful cases only, ensuring a comprehensive and unbiased assessment of perceptual quality and semantic accuracy. All the User Study source videos can be found in the $User\_Study$ folder of the supplementary materials.

The inference efficiency of our framework scales with both mesh complexity (vertex/face count) and FPS feature sampling density. Our empirical studies indicate that an 8:1∼4:1 ratio between vertex count and FPS samples achieves optimal performance-efficiency trade-off. The corresponding inference times across different mesh resolutions under this sampling configuration are presented in Tab. 1.

2. Animation Results of AnymateAnyMesh

We curated a diverse collection of high-fidelity static meshes from Sketchfab [9], encompassing various categories including humanoid figures, animals, weapons, clothes, and environmental assets. These meshes were animated using our proposed AnimateAnyMesh framework through text-driven synthesis. Fig. 3 demonstrates representative results, showcasing our framework's capability to generate high-fidelity, versatile animations across arbitrary mesh topologies. The qualitative results validate the effectiveness of our approach in achieving generalized mesh animation with exceptional geometric fidelity, motion naturalness, and semantic flexibility.

Moreover, given identical input prompts and initial mesh as condition, our AnimateAnyMesh framework demonstrates robust multi-modal synthesis capabilities through different random seeds, generating diverse yet plausible high-fidelity mesh animations. Fig. 4 illustrates this generative flexibility through exemplar results, highlighting our framework's ability to explore varied motion manifestations while maintaining semantic consistency and geometric integrity.

3. Additional Qualitative Comparison

For a comprehensive comparison of mesh animation approaches, we present additional comparative examples against baseline methods in Fig. 5. The results consistently support our main findings, demonstrating that our approach outperforms existing methods in terms of text-motion alignment, motion naturalness, and shape preservation fidelity.

4. Additional Ablation Studies

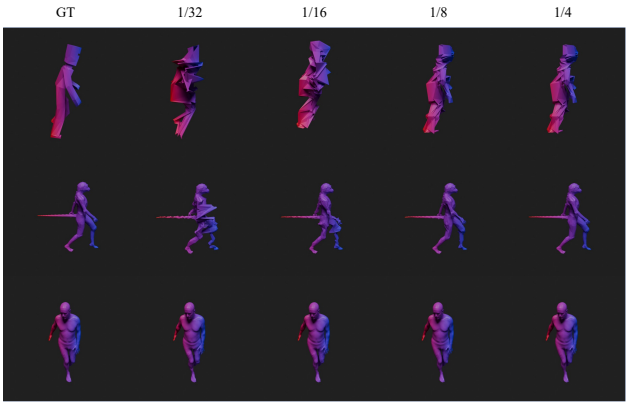

Figure 2. Ablation study on FPS Sampling Ratio. The numerical annotations above each image indicate the FPS ratio employed in the DyMeshVAE Encoder. Please zoom in for a better view.

FPS Sampling Ratio. We conduct ablation study on the sampling ratio in the DyMeshVAE encoding procedure to see how it influences the reconstruction quality. We randomly sampled three dynamic mesh sequences from the DyMesh dataset with vertex counts of 369, 2,567, and 6,890 (top to bottom). For each mesh, we conducted experiments using FPS (Farthest Point Sampling) ratios of 1/32, 1/16, 1/8, and 1/4 for feature sampling and reconstruction.

num_v	FPS Sampling Ratio			
	1/32	1/16	1/8	1/4
369	136.57	58.78	22.45	8.10
2,567	17.74	3.64	1.16	0.54
6,890	1.57	0.66	0.51	0.50

Table 2. Reconstruction error for dynamic mesh sequences with varying vertex counts under different FPS sampling ratios.

The qualitative visualizations and quantitative metrics are presented in Fig. 2 and Tab. 2, respectively, demonstrating the impact of sampling density on reconstruction fidelity.

As demonstrated in Fig. 2, meshes with low vertex counts (row 1_{st}) exhibit poor reconstruction quality even with a 1/4 sampling ratio, while high-vertex-count meshes (row 3_{rd}) maintain satisfactory reconstruction fidelity even at a 1/32 sampling ratio, as validated by quantitative metrics in Table 3. We attribute this phenomenon to spatial distribution characteristics: low-vertex meshes typically exhibit sparse spatial distribution, leading to significant geometric information loss in regions surrounding unsampled vertices, whereas high-vertex-count meshes maintain dense surface coverage even with lower sampling ratios, enabling better preservation of local geometric features during encoding. Based on these observations, we set the feature sampling count to 512 during training to facilitate efficient batch processing while achieving an optimal balance between performance and computational efficiency across meshes with diverse vertex counts. During inference, we adopt an adaptive sampling strategy where n = min(512, $num\_v//8$) for inference, where num_v represents the number of mesh vertices. We empirically find the robust performance across the majority of test cases with this setting.

5. Scaling Experiments

In this section, we conduct scaling experiments across three dimensions: dataset scale, temporal resolution (frame count), and model capacity. We evaluate four configurations, each denoted as $\mathrm{rf\_}<1>\mathrm{v\_}<2>\mathrm{f\_}<3>\mathrm{p}$, where $1, 2, 3$ represents the maximum number of vertices of the dataset, number of frames per instance, and the number of parameters of the Shape-Guided Text-to-Trajectory Model. All models were trained for 600,000 iterations on corresponding DyMesh subsets with a batch size of 2048 and a learning rate of 2e-4. For evaluation, we generate mesh animations using the same 10 mesh-prompt pairs from our qualitative benchmark, maintaining consistent random seeds across all models. Front-view renderings were produced to compute the VBench [4] metrics (I2V, M.Sm, Aest.Q) discussed in the main text. Additionally, given that all four trained models demonstrate the capability to generate high-quality and semantically plausible mesh animations, we incorporate the Dynamic Degree metric (abbreviated as Dy.Dg) from VBench to quantitatively assess motion intensity. The comprehensive results are presented in Tab. 3. The results indicate that: (1) Increasing the maximum number of vertices leads to better performance on most metrics (B vs. A). (2) Increasing the number of frames will improve the output dynamic, improving Dy.Dg while maintaining promising results on other metrics. (C vs. A). (3) Scaling the model's parameter size leads to better performance on all metrics, demonstrating good scalability of our method. (D vs. A).

	Experiment	I2V ↑	M.Sm ↑	Aest.Q ↑	Dy.Dg ↑
A	rf_4096v_16f_200Mp	0.954	0.995	0.539	0.693
B	rf_8192v_16f_200Mp	**0.985**	0.996	**0.550**	0.605
C	rf_4096v_32f_200Mp	0.948	0.993	0.532	**0.737**
D	rf_4096v_16f_740Mp	0.968	**0.997**	0.545	0.705

Table 3. Scaling experiments of AnymateAnyMesh. We name the experiments as $\mathrm{rf\_}<1>\mathrm{v\_}<2>\mathrm{f\_}<3>\mathrm{p}$, where $1, 2, 3$ represents the maximum number of vertices of the dataset, number of frames per instance, and the number of parameters of the Shape-Guided Text-to-Trajectory Model.

6. Limitation

Our work exhibits three limitations: First, regarding dataset scale, while the proposed DyMesh Dataset encompasses over 4M dynamic mesh sequences, the number of unique mesh identities remains below 100k, potentially limiting model generalization across specialized categories. We plan to address this by creating and curating additional high-quality, diverse 4D datasets. Second, concerning annotation quality, we observe that current video captioning models demonstrate suboptimal performance when annotating 3D rendered sequences without natural backgrounds, compared to their performance on natural videos, particularly in motion description granularity. Enhancing caption fidelity for synthetic 3D content remains a key research direction. Third, in terms of model capabilities, the current implementation of AnimateAnyMesh is confined to 16/32-frame sequence generation, and extending the model's capability to generate longer-duration mesh animations represents a significant future research objective.

References

[1] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025. 1

[2] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021. 2

[3] 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

[4] Ziqi Huang, Yinan He, Jiashuo Yu, Fan Zhang, Chenyang Si, Yuming Jiang, Yuanhan Zhang, Tianxing Wu, Qingyang Jin, Nattapol Chanpaisit, et al. Vbench: Comprehensive benchmark suite for video generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21807–21818, 2024. 2, 4

[5] Yang Li, Hikari Takehara, Takafumi Taketomi, Bo Zheng, and Matthias Nießner. 4dcomplete: Non-rigid motion estimation beyond the observable surface. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12706–12716, 2021. 1

[6] Zhen Li, Zuo-Liang Zhu, Ling-Hao Han, Qibin Hou, Chun-Le Guo, and Ming-Ming Cheng. Amt: All-pairs multi-field transforms for efficient frame interpolation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9801–9810, 2023. 3

[7] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. Smpl: A skinned multi-person linear model. *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*, pages 851–866, 2023. 1

[8] Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. AMASS: Archive of motion capture as surface shapes. In *International Conference on Computer Vision*, pages 5442–5451, 2019. 1

[9] Sketchfab. Sketchfab. https://sketchfab.com/, 2024. Accessed: 2024-05-21. 1, 3

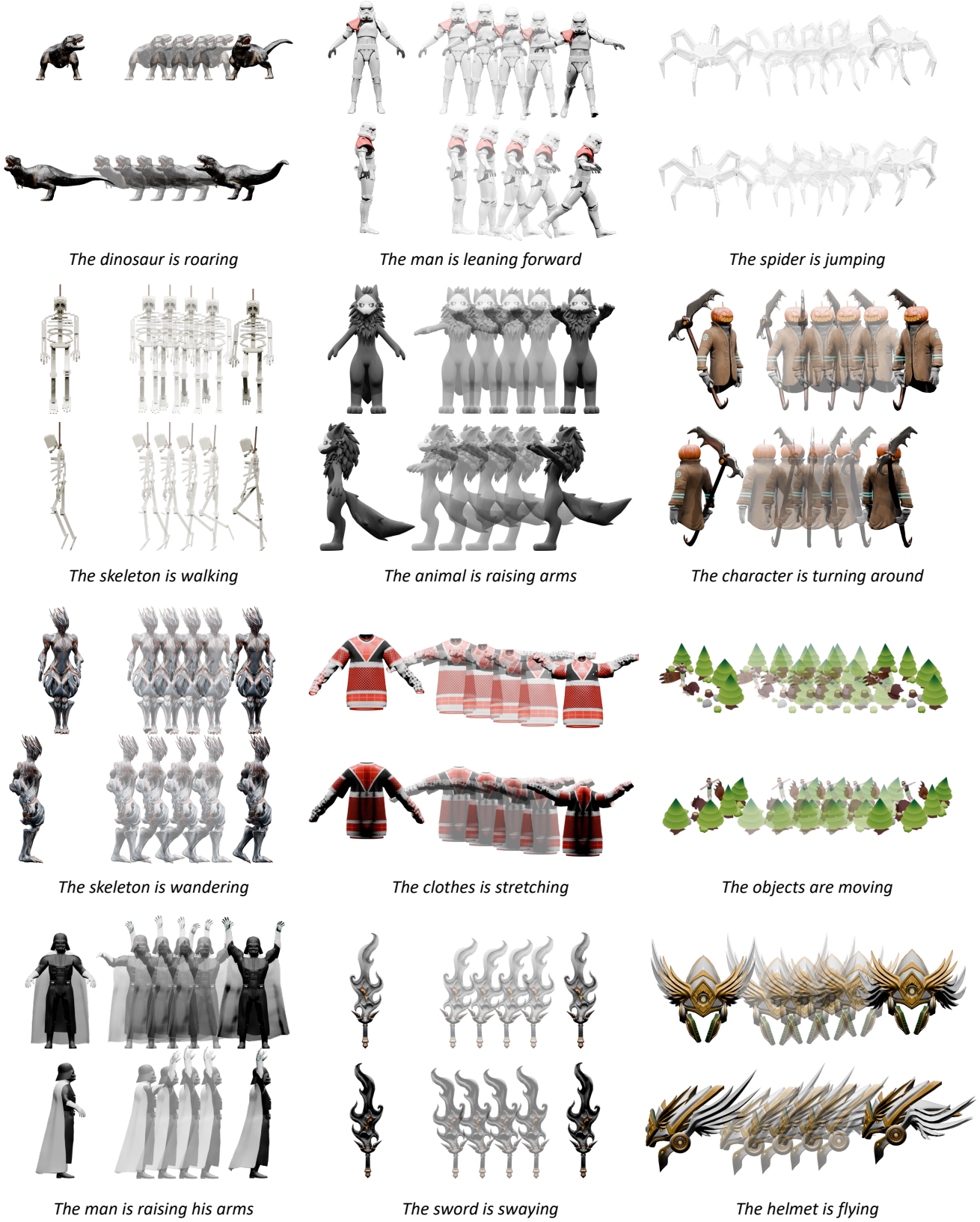

Figure 3. Examples of text-driven mesh animation results of the proposed AnimateAnyMesh. We render two random views for each example. Please zoom in for a better view.

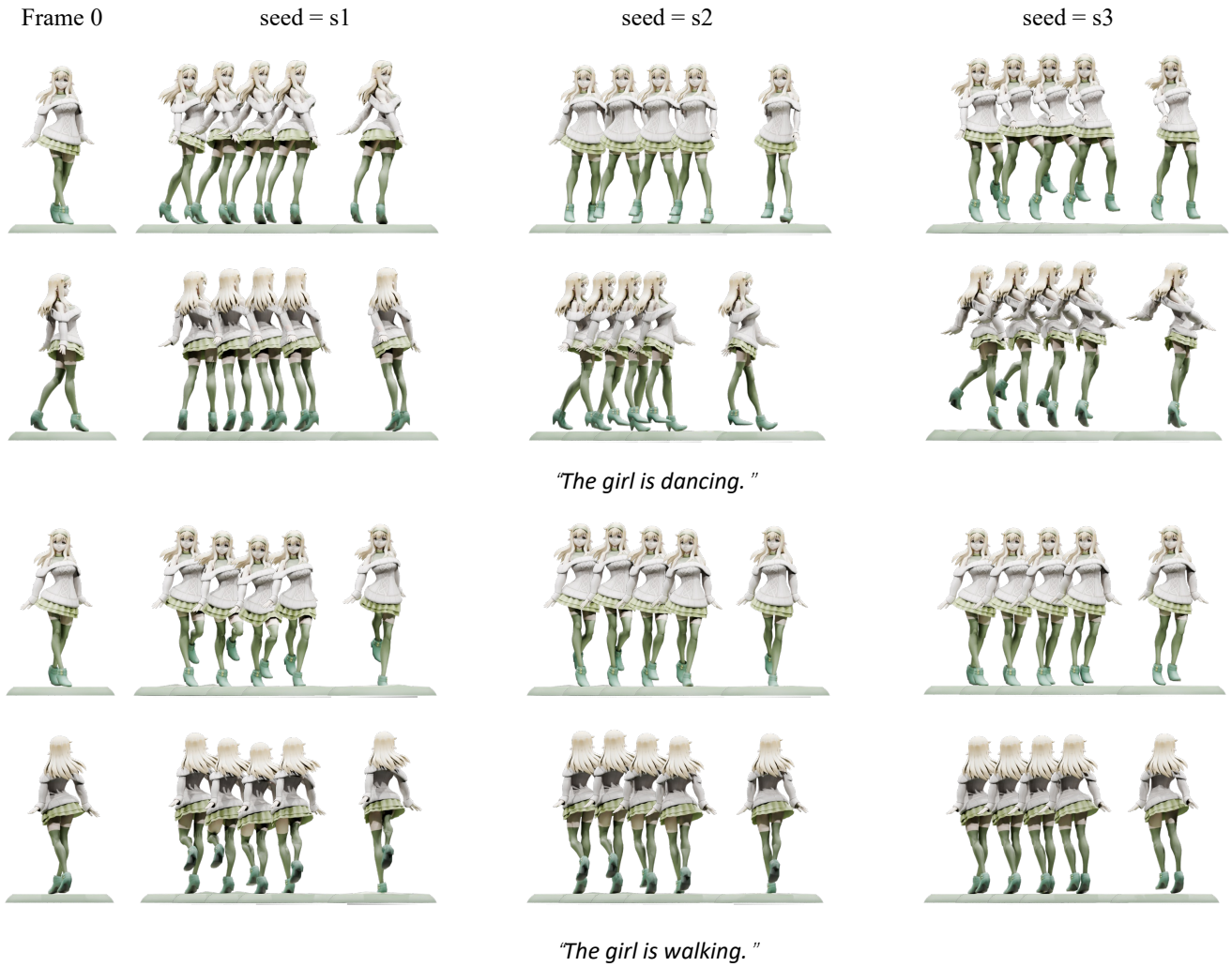

Figure 4. Diversity Demonstration of AnimateAnyMesh Generations. Given identical text-prompt and initial mesh conditions, AnimateAnyMesh demonstrates the capability to generate diverse, high-quality mesh animations through random seed variation.

Figure 5. Additional qualitative comparison with state-of-the-art mesh animation methods.