

BiMotion: B-spline Motion for Text-guided Dynamic 3D Character Generation

Supplementary Material

This supplementary material provides additional implementation details, further experiments and results, outlines the user study design and provides statistics, additional ablations (*e.g.*, extended normal-fusion ablations), and describes our dataset captioning pipeline. We also encourage readers to view the **accompanying video** for demonstrations of the dynamic results.

A. Additional Implementation Details

A.1. Velocity Network

We implement the velocity field $v_{\theta_{\text{vel}}}$ (described in Sec. 3.6 in the paper) based on Diffusion Transformer (DiTs) [14] (see Fig. A1). To promote spatial–motion correspondence, we follow the open-source image-to-video model Wan [19] to concatenate normalized latents along the channel dimension as $\tilde{\mathbf{z}}_{\parallel} = [\tilde{\mathbf{z}}_0, \tilde{\mathbf{z}}_{\mathcal{P}}^{(\tau)}]$. Text prompts are encoded using a pretrained CLIP text encoder [15] to obtain text embeddings. Following [5], we apply separate AdaLN [14] parameters conditioned on the timestep τ to modulate the concatenated latents, the latent $\tilde{\mathbf{z}}_0$, and the text embedding independently. The rescaled $\tilde{\mathbf{z}}_{\parallel}$ first passes through self-attention, then through cross-attention blocks that fuse information from the initial shape and the text embedding. This modality-specific AdaLN dynamically modulates each conditioning signal to vary modality-specific influence across sampling timesteps. After $S' = 12$ such DiT attention blocks, the network decodes the control-point velocities $v_{\theta_{\text{vel}}}(\tilde{\mathbf{z}}_{\mathcal{P}}^{(\tau)} | \tau, \tilde{\mathbf{z}}_0, y)$.

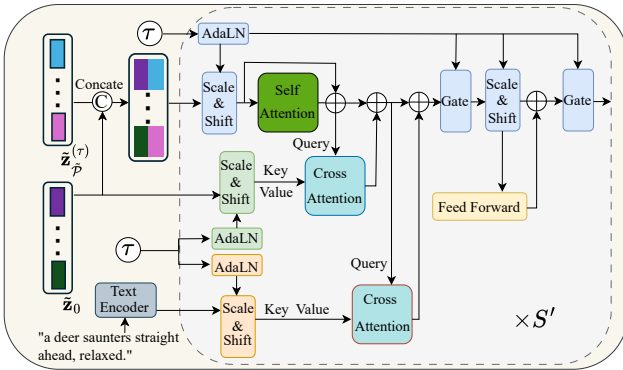


Figure A1. **Velocity Architecture.** The network $v_{\theta_{\text{vel}}}$ comprises S' stacked layers with self- and cross-attention modules.

A.2. Training Details

To reduce I/O cost during training, and to speed up training, each dense point set ($N = 200\text{K}$) is randomly

split into ten files of 20K points stored separately in pre-processing, where all initial shapes are normalized into range $[-0.9, 0.9]$. In our control-point embedding, we adopt a multi-level hierarchical strategy with a decreasing sequence (from finer to coarser) of control-point counts $[k_s, \dots, k_0] = [17, 15, 13, 11, 9, 7, 5, 4]$. When the input control-point sequence has length $k = 16$, we pad it to 17 by replicating the last control point to ensure compatibility.

B. Additional Experiments

B.1. Additional Comparisons

To provide a comprehensive comparison of mesh animation approaches, we include additional qualitative results against three baseline methods in Fig. A2. We further evaluate additional cases against the closest competitor, AnimateAnyMesh [20], using multi-mesh sources from Objaverse [4], Sketchfab [16], and AI-generated assets from Meshy [11] and Hunyuan3D [23]. Additional comparisons with AnimateAnyMesh are presented here in Fig. A3 and in Fig. 1 of the main paper. Across all settings, the results consistently support our main observation that method produces richer text-aligned motion, more natural dynamics, and more expressive animations compared to existing approaches.

For certain walking cases, such as Elephant and Robot in Fig. A3, the motion may appear to be “in-place.” This is because many dynamic 3D assets depict walking using in-place cycles by default. Artists typically add the global displacement (*i.e.*, the translation) only when integrating the character into a specific scene or environment.

B.2. Additional Baselines

Puppeteer. We also evaluate Puppeteer [17], a three-stage pipeline for driving the animation of a given mesh. First, it performs automatic rigging (skeleton and skinning estimation). Second, the static mesh is rendered to produce an initial conditioning frame for video generation. Third, it optimizes the frame-by-frame motion by driving the estimated skeleton, so that the rendered frames match Kling-generated video conditioned on the rendering and prompt [6]. As shown in Fig. A4, Puppeteer is sensitive to rigging quality. For example, an incorrect rig can cause the Spider sequence to remain static. In addition, Puppeteer easily accumulates errors from early frames during the optimization process, leading to severe artifacts in some cases, such as the Robot example. In our experiments, Puppeteer consumed a peak GPU memory of approximately 48 GB and required about 58 minutes of runtime on a single A100 GPU

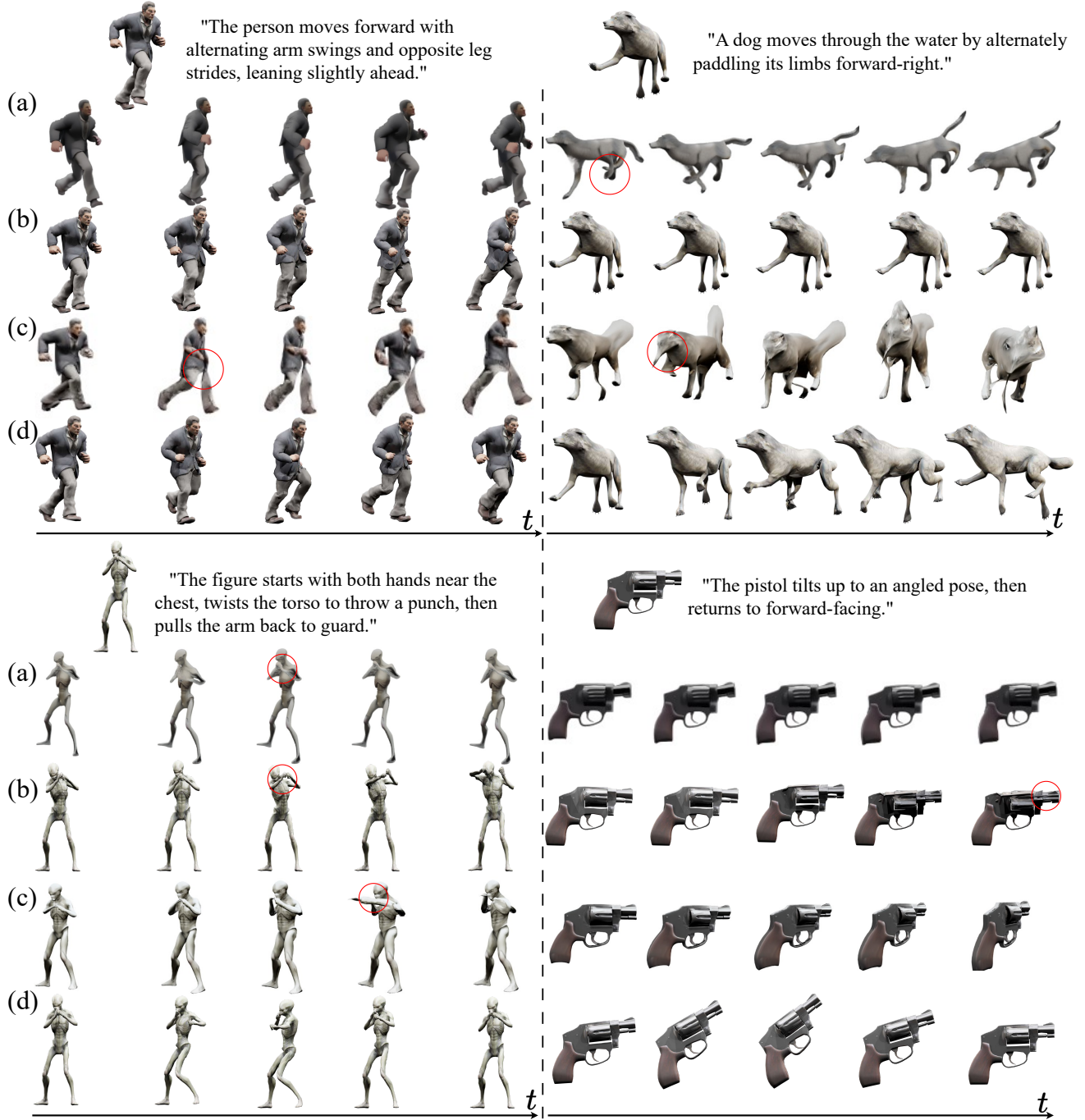


Figure A2. **Additional Qualitative Comparisons.** (a) GVFDiffusion [22], (b) AnimateAnyMesh [20], (c) V2M4 [2], (d) BiMotion (Ours). BiMotion yields more text-aligned, higher-quality motion. Visual artifacts are illustrated in red.

to process a single sequence.

Video Generation Models. For comparison with video-based dynamic 3D generation methods [2, 22], we use Kling 2.1 [6] as the video source, conditioned on the first frame and the text prompts. To validate this choice, we compare Kling with other state-of-the-art video generation

models: Adobe Firefly [1], HunyuanVideo-I2V [7], and Wan-2.5 [19]. As shown in Fig. A5, Adobe Firefly and HunyuanVideo-I2V both struggle to maintain subject consistency and often introduce unrelated or unstable visual content. Wan-2.5 produces motion that aligns with the text prompts but lacks the expressiveness achieved by Kling.

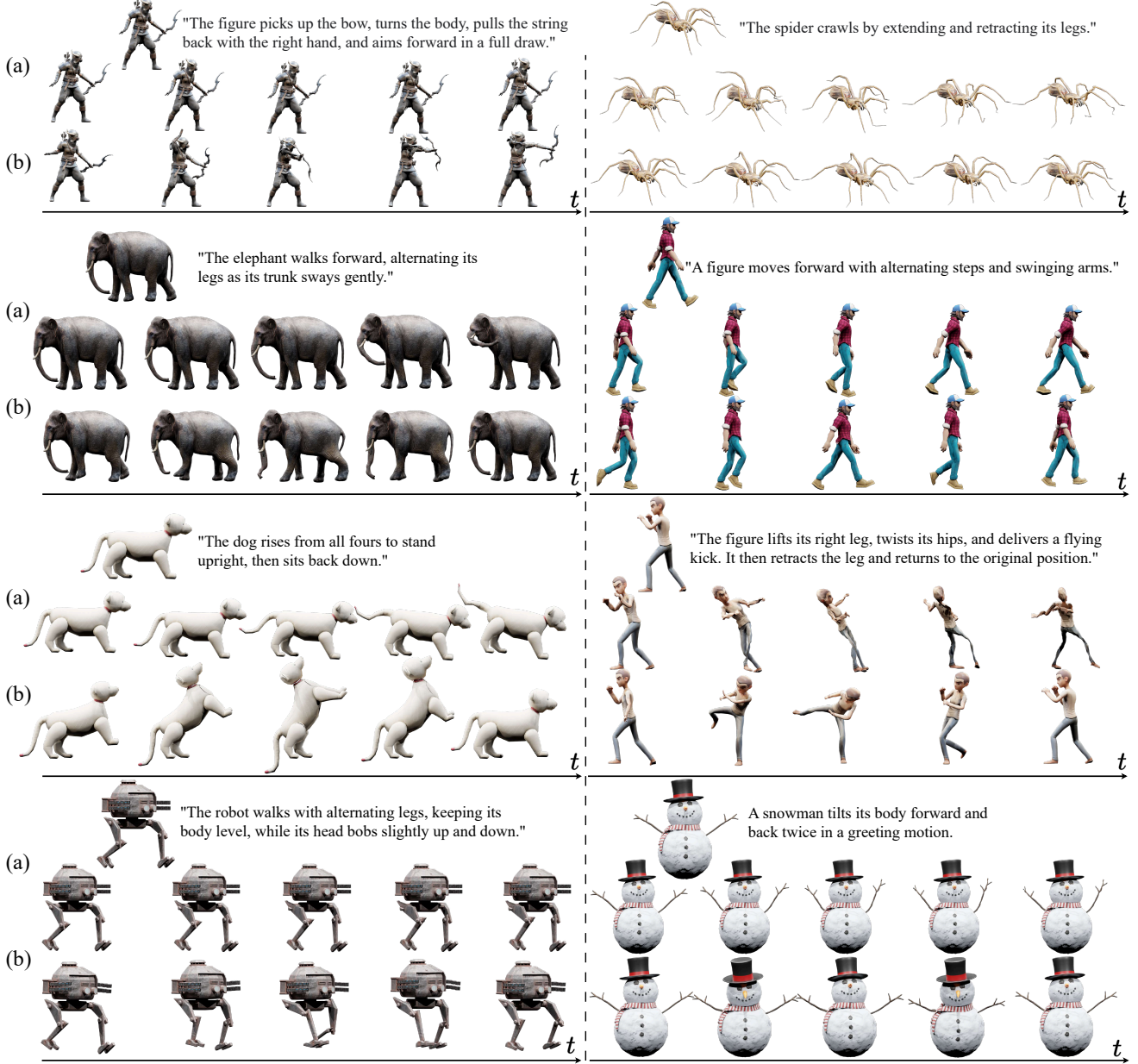


Figure A3. **Additional Qualitative Comparisons with AnimateAnyMesh.** (a) AnimateAnyMesh [20] and (b) BiMotion (Ours).

B.3. Additional Ablations

Variable-Length Sequence Representation. With BiMotion, the B-spline-based VAE robustly handles variable-length sequences, even with a limited number of control points. In addition to the mean statistics reported in Tab. 2 of the main paper, we provide binned reconstruction-error averages for a more fine-grained analysis. In Fig. A6 we report the mean L1 reconstruction error computed per trajectory and averaged per instance within each sequence-length bin. While our method reliably models short- and medium-

length sequences, reconstruction error increases for longer sequences unless more than 16 control points are used. To ensure a fair comparison with AnimateAnyMesh, we fix the number of control points to 16 in all experiments.

Additional Normal Fusion Ablations. We evaluate two alternative designs for incorporating surface normals into the initial shape features. (1) \mathcal{N}_0 **Frequency-PE.** Normals \mathcal{N} are encoded using the same frequency positional encoding (Frequency-PE) applied to the point coordinates [12] and then passed through an MLP to obtain $\mathbf{F}_{\mathcal{N}_0} \in \mathbb{R}^{n \times c}$. The fused feature \mathbf{F}_0 is computed using Eq. 5. (2) \mathcal{N}_0 **Concat.**

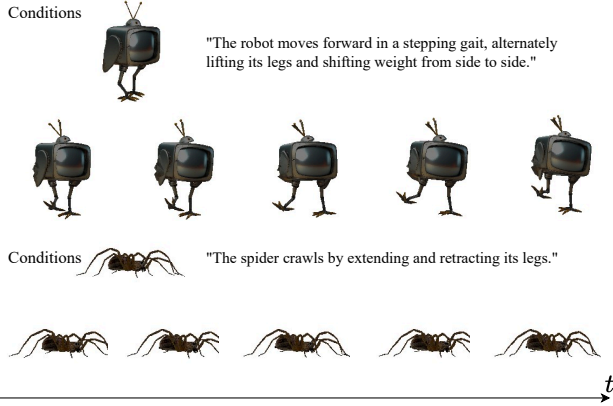


Figure A4. **Puppeteer Qualitative Results.** Results are rendered using the authors’ open-source renderer, whose texture appearance differs from ours and AnimateAnyMesh.

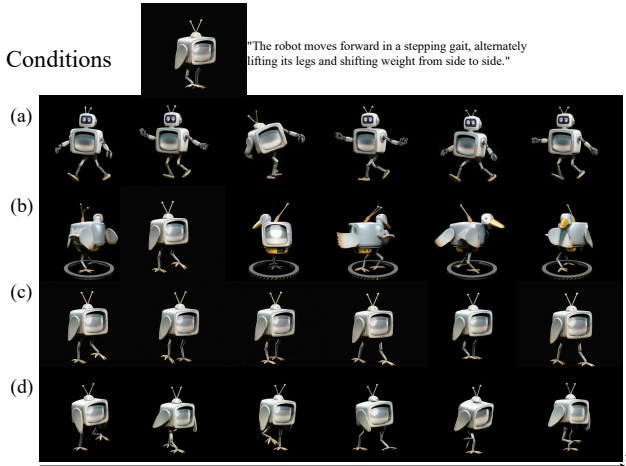


Figure A5. **Video Generation Model Comparison.** Qualitative comparison of video outputs from (a) Adobe Firefly [1], (b) HunyuanVideo-I2V [7], (c) Wan-2.5 [19], and (d) Kling 2.1 [6]. Note: these generation models convert transparent backgrounds to black by default.

As in the main paper, we first extract $\mathbf{F}_{\mathcal{N}_0}$ from \mathcal{N} using an MLP. To fuse normals, we concatenate $\mathbf{F}_{\mathcal{N}_0}$ with $\mathbf{F}_{\mathcal{P}_0}$ along the feature dimension and feed the result into another MLP to obtain \mathbf{F}_0 .

We report the VAE reconstruction error for these variants in Tab. A1. Encoding normals with Frequency-PE disrupts their spherical geometric structure, introducing noise and degrading reconstruction accuracy. In contrast, our cross-similarity fusion aligns the MLP-projected normal features to the same latent space as the 3D position features, enabling more effective fusion and yielding substantially improved reconstruction compared with naive concatenation.

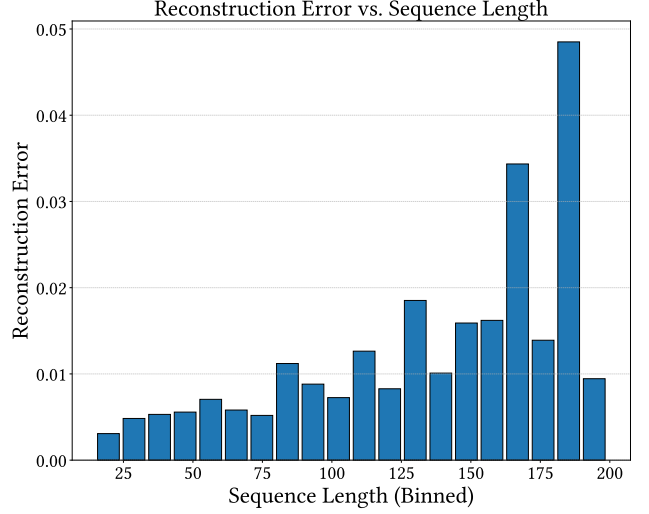


Figure A6. **VAE Reconstruction Error vs. Sequence Length.** Mean L1 reconstruction error for our VAE, computed per trajectory and averaged per instance, reported for each sequence-length bin.

Table A1. **Additional normal-fusion ablations.** Mean L1 reconstruction error ($\times 10^{-2}$), averaged per trajectory and per instance.

w/o \mathcal{N}_0	\mathcal{N}_0 Frequency-PE	\mathcal{N}_0 Concat.	Ours
1.328	1.331	1.294	1.078

B.4. User Study Details

We conducted an anonymous user study to evaluate the perceptual quality of text-driven mesh animations. A total of 20 participants with diverse levels of experience in computer vision and graphics were recruited. Each participant evaluated four generation methods on the same ten randomly selected example sequences and rated three criteria on a 5-point Likert scale (1 = Very Poor, 5 = Excellent). We evaluated the following three questions for each sequence: (i) *Text-to-Motion Agreement* (TA), (ii) *Motion Plausibility* (MP), and (iii) *Motion Expressiveness* (ME). In total, each participant contributed $4 \times 10 \times 3 = 120$ ratings.

Interface and Protocol. As shown in Fig. A7, for each example object, the form displayed a 2×2 grid of videos labeled Video 1–4, each corresponding to one of the four different methods. To avoid bias, the mapping from methods to Video 1–4 was anonymized and randomized per participant and per example. The physical positions of the four videos in the 2×2 grid were also randomly permuted for every page load. Participants could replay the clips freely before submitting their scores for TA/MP/ME (separate questions immediately under the grid). All responses were collected anonymously; participants could withdraw at any time without providing a reason. This study obtained ethics approval from the relevant internal ethics review board.

Results. We report both the mean and standard deviation for each method in Tab. A2. There we observe that users consistently favor BiMotion, which achieves the highest mean score and the lowest standard deviation across all metrics. For detailed data, please refer to the raw survey metadata in the accompanying supplementary material, titled “Anonymous User Study (Responses) – Form responses 1.csv”.

Table A2. **User study results.** Each cell shows the mean \pm standard deviation over all participants and examples.

Methods	TA \uparrow	MP \uparrow	ME \uparrow
GVFDiffusion [22]	2.343 \pm 0.407	2.300 \pm 0.380	2.443 \pm 0.387
AnimateAnyMesh [20]	2.314 \pm 0.394	2.686 \pm 0.464	2.443 \pm 0.357
V2M4 [2]	2.876 \pm 0.368	2.714 \pm 0.354	3.048 \pm 0.348
BiMotion (Ours)	4.095\pm0.334	4.062\pm0.328	4.048\pm0.339

C. Dataset and Preprocessing

C.1. Dataset Curation

To validate our method, we curated **BIMO** (Bspline Motion) dataset. We collect diverse mesh motion sequences from multiple datasets: DeformingThings4D [8], ObjaverseV1 [4], and ObjaverseXL [3]. For DeformingThings4D, we use the official script to extract animal mesh sequences from the raw ‘.anime’ files. For ObjaverseV1 and ObjaverseXL, we extract sequences with more than ten frames, discarding low-motion sequences based on object-center displacement, and bounding-box scale changes. We handle multiple raw formats from Objaverse (e.g., “.glb”, “.gltf”, “.fbx”) and filter out unnecessary elements such as “icosphere”, “sphere”, “proxy”, “env”, “origin”, “pivot”, “camera”, “ground”, “floor”, and “light”. The ObjaverseV1 asset IDs are curated by Diffusion4D [9], while ObjaverseXL meta IDs are filtered by GVFDiffusion [22]. All mesh sequences are capped at 200 frames following AnimateAnyMesh [20]. The distribution of mesh sequence lengths is illustrated in Fig. A9, and the summary statistics of our curated dataset are presented in Tab. A3. The final dataset comprises 38,944 motion sequences with a total of 3,682,790 frames.

Table A3. **Dataset statistics.** Summary statistics of our curated mesh motion dataset **BIMO**.

Dataset Source	Number of Motions	Total Frames
DeformingThings4D [8]	1,770	87,052
ObjaverseV1 [4]	10,550	1,116,134
ObjaverseXL [3]	26,624	2,479,604
Total motions	38,944	3,682,790

C.2. Auto-Captioning Pipeline

To obtain captions for conditioning our model, we use human-annotated motion descriptions provided by OmniMotionGPT [21] for DeformingThings4D, where each motion is annotated with three captions written by trained annotators. For Objaverse, there are no captions available, and so we build an automatic captioning pipeline (see Fig. A8) designed to emulate the style of the DeformingThings4D annotations. To reduce LLM token usage, we adopt a hyper-prompt design that first summarizes human annotations with an LLM to produce three concise prompts for a VLM. These distilled prompts, paired with the rendered frames, are supplied to the VLM as an alternative to requiring expensive to obtain human annotations for each caption. Concretely, we prompt an LLM (GPT-5 [13]) to imitate human annotators and generate three distinct video-captioning prompts. Our initial hyper-prompt that is provided to the LLM is:

Hyper Prompt

The following examples show how real human annotators describe the motion observed in a given sequence of dynamic frames:

Case 1 A bear stands up and then returns to all fours. Another description states that the bear pushes with its front paws, stands on its hind legs, and then settles back down. A third annotator writes that the bear rose to its feet, used its front paws for support, stood upright, and then rested.

Case 2 A bull makes a left turn while remaining in one spot in the water. Another description notes that the bull turns in place, while a third says the bull swims in circles in the water.

Case 3 A wolf retreats in fright. Another annotator writes that the wolf withdraws in fear, while a third notes that the wolf is startled on the left and retreats to the right.

Case 4 A wolf falls into the water and moves forward. Another version describes it as dropping into the water, while a third says it plunges into the water and relaxes.

Case 5 A deer falls from above downward. Another description states that the deer descends from a higher to a lower position, while a third says it drops from high to low.

Case 6 A deer swims to the left while staying in place. Another annotator describes it as paddling left without advancing, while a third

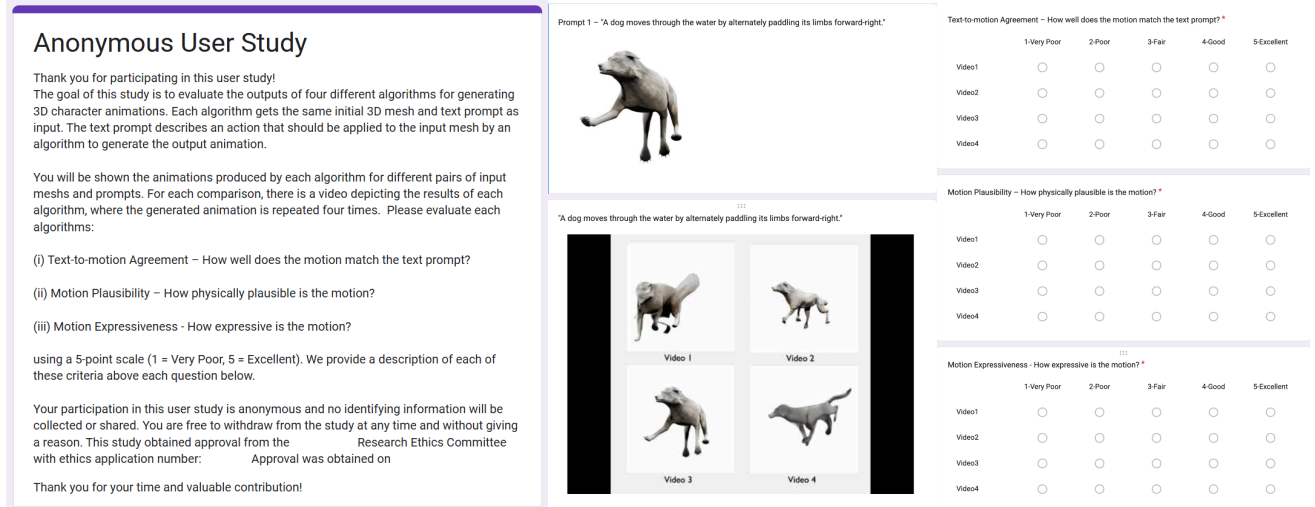


Figure A7. **User Study Interface and Protocol.** **Left:** Instructions page. Screenshot of the questionnaire introduction; personal or institutional information approved by the research ethics committee has been masked. **Middle:** Video comparison interface showing the conditions of both the first frame and the textual prompt, along with four videos generated by four randomly ordered methods. **Right:** Three evaluation questions used to collect user preferences, forming the basis of the subjective evaluation metrics.

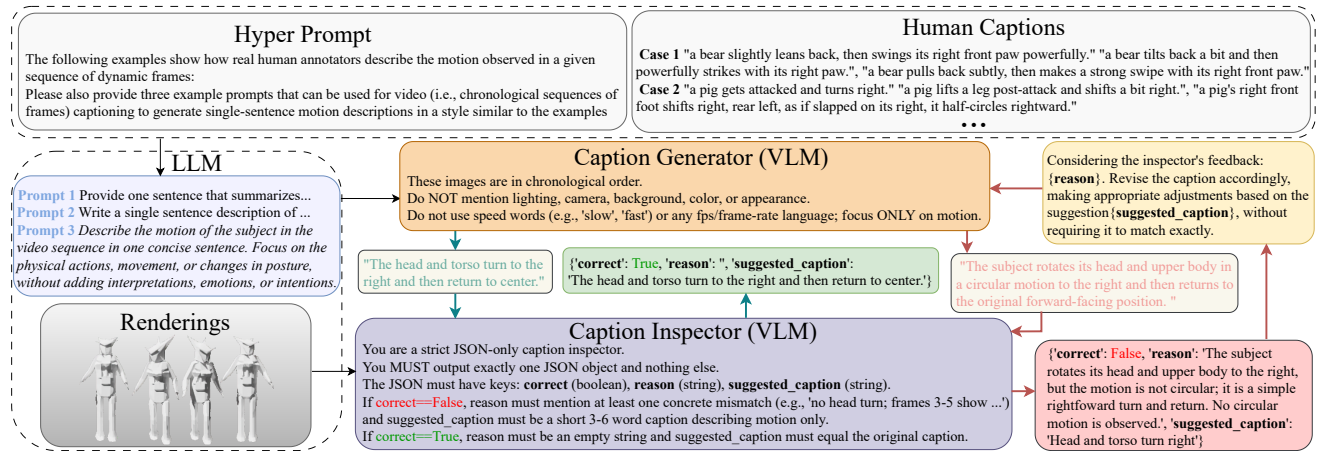
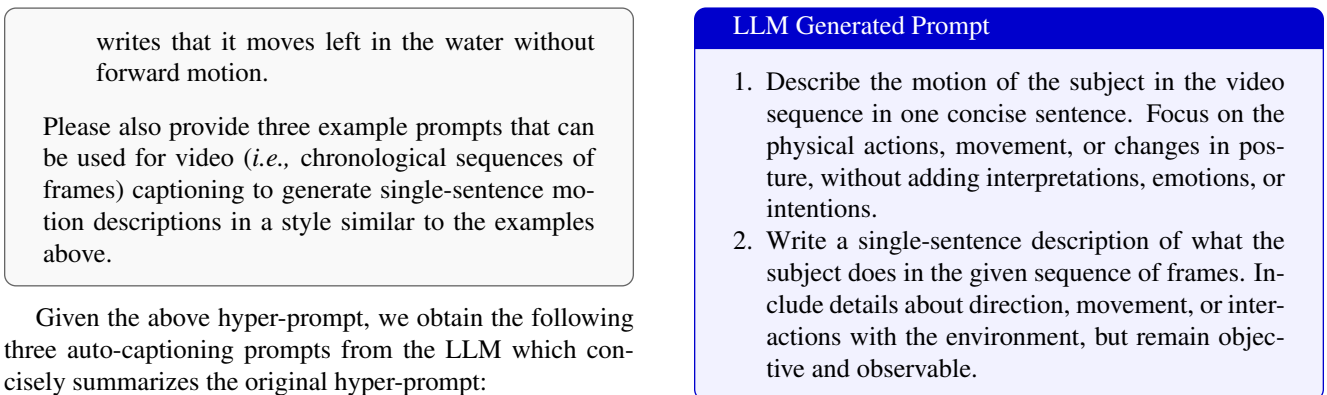


Figure A8. **Auto-captioning pipeline.** We propose an iterative captioning pipeline that incorporates both a caption inspector and a caption generator to produce accurate and consistent captions for dynamic assets.



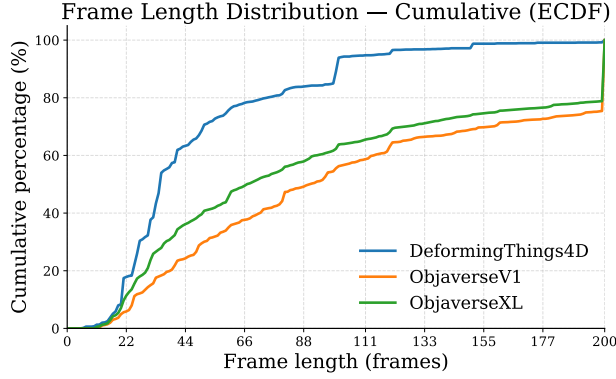


Figure A9. **Frame-length cumulative distribution.** Distribution of mesh sequence lengths (length ≤ 200) from DeformingThings4D [8], ObjaverseV1 [4], and ObjaverseXL [3].

3. Provide one sentence that summarizes the main motion occurring in the video. Describe only what is visually happening, such as movement, turning, rising, falling, or swimming, in chronological order.

Next, we render the Objaverse dynamic assets from a fixed front-view camera using a modified Blender script adapted from Diffusion4D [9]. Rendering is performed on ten A6000 GPUs with the Cycles engine, producing transparent 256×256 frames. We then use a VLM (GPT-5) to generate captions for each rendered sequence. To ensure consistency and reliability, we incorporate an iterative caption-inspection loop using the same model. The caption inspector (implemented with the same VLM GPT-5) receives both the renderings and the captions produced by the caption generator, evaluates their correctness, and provides reasoning and suggested revisions when necessary. The caption-generation agent then updates the caption and repeats the process six times until the inspector approves.

C.3. VLM Ablations

To validate our VLM choice, we ablate GPT-5, Qwen-2.5 [18], and LLaVA-NeXT-Video [10] in Fig. A10. GPT-5 consistently yields motion-focused, non-speculative descriptions that match the DeformingThings4D annotation style. Qwen-2.5 frequently hallucinates global behaviors (*e.g.*, stepping forward, full-body reorientation) and conflates pose with inferred intent. LLaVA-NeXT-Video more often attributes locomotion or complex actions absent from the frames. These errors shift the caption distribution away from true 4D motion and weaken text–motion correspondence during training. Our GPT-5 hyper-prompt pipeline produces stable, grounded, and fine-grained captions that improve the robustness and fidelity of downstream text-

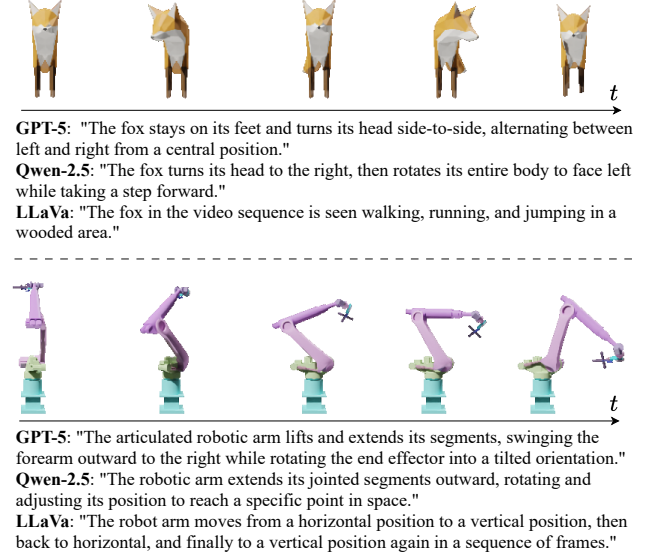


Figure A10. **VLM Captioning Ablation.** We compare captions generated by three video–language models (VLMs). Using GPT-5 produces concise and motion-faithful descriptions closely aligned with the real 4D dynamics.

conditioned animation synthesis.

References

- [1] Adobe. Free ai video generator: Text to video online – adobe firefly, 2025.
- [2] Jianqi Chen, Biao Zhang, Xiangjun Tang, and Peter Wonka. V2m4: 4d mesh animation reconstruction from a single monocular video. In *International Conference on Computer Vision*, 2025.
- [3] 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*, 2023.
- [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 *Computer Vision and pattern recognition*, 2023.
- [5] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *International Conference on Machine Learning*, 2024.
- [6] Kling AI. Kling ai: Next-generation ai creative studio. <https://klingai.com/>, 2024. Accessed: 2024-02-05, 07, 01, 04.
- [7] Weijie Kong, Qi Tian, Zijian Zhang, Rox Min, Zuozhuo Dai, Jin Zhou, Jiangfeng Xiong, Xin Li, Bo Wu, Jianwei Zhang, et al. Hunyuanvideo: A systematic framework for large video generative models. *arXiv:2412.03603*, 2024.
- [8] Yang Li, Hikari Takehara, Takafumi Taketomi, Bo Zheng, and Matthias Nießner. 4dcomplete: Non-rigid motion estimation beyond the observable surface. *International Conference on Computer Vision*, 2021.
- [9] Hanwen Liang, Yuyang Yin, Dejia Xu, Hanxue Liang, Zhangyang Wang, Konstantinos N Plataniotis, Yao Zhao, and Yunchao Wei. Diffusion4d: fast spatial-temporal consistent 4d generation via video diffusion models. In *Advances in Neural Information Processing Systems*, 2024.
- [10] Haotian Liu, Yuheng Li, Zhuowan Zhang, and Chunyuan Li. LLaVA-NeXT-Video: Video Large Multimodal Models, 2024.
- [11] Meshy. Discover — meshy. <https://www.meshy.ai/discover>, 2025. Accessed: 2025-10-27.
- [12] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 2021.
- [13] OpenAI. Introducing GPT-5, 2025. Accessed: 2025-11-03.
- [14] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *International Conference on Computer Vision*, 2023.
- [15] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Computer Vision and Pattern Recognition*, 2022.
- [16] Sketchfab. Sketchfab free 3d models. <https://sketchfab.com/features/free-3d-models>. Accessed: 2025-11-08.
- [17] Chaoyue Song, Xiu Li, Fan Yang, Zhongcong Xu, Jiacheng Wei, Fayao Liu, Jiashi Feng, Guosheng Lin, and Jianfeng Zhang. Puppeteer: Rig and animate your 3d models. In *Advances in Neural Information Processing Systems*, 2025.
- [18] Qwen Team. Qwen2.5: A scalable and open large language model family, 2024.
- [19] Team Wan, Ang Wang, Baole Ai, Bin Wen, Chaojie Mao, Chen-Wei Xie, Di Chen, Feiwei Yu, Haiming Zhao, Jianxiao Yang, et al. Wan: Open and advanced large-scale video generative models. *arXiv:2503.20314*, 2025.
- [20] Zijie Wu, Chaohui Yu, Fan Wang, and Xiang Bai. Animateanymesh: A feed-forward 4d foundation model for text-driven universal mesh animation. In *International Conference on Computer Vision*, 2025.
- [21] Zhangsihao Yang, Mingyuan Zhou, Mengyi Shan, Bingbing Wen, Ziwei Xuan, Mitch Hill, Junjie Bai, Guo-Jun Qi, and Yalin Wang. Omnimotiongpt: Animal motion generation with limited data. In *Computer Vision and Pattern Recognition*, 2024.
- [22] Bowen Zhang, Sicheng Xu, Chuxin Wang, Jiaolong Yang, Feng Zhao, Dong Chen, and Baining Guo. Gaussian variation field diffusion for high-fidelity video-to-4d synthesis. In *International Conference on Computer Vision*, 2025.
- [23] Zibo Zhao, Zeqiang Lai, Qingxiang Lin, Yunfei Zhao, Haolin Liu, Shuhui Yang, Yifei Feng, Mingxin Yang, Sheng Zhang, Xianghui Yang, et al. Hunyuan3d 2.0: Scaling diffusion models for high resolution textured 3d assets generation. *arXiv:2501.12202*, 2025.