TESLA: Native 4D Gaussian Splatting Generation with Temporally Structured Latents

Weiqi Zhang[1*] **Junsheng Zhou**[1*†] **Juntong Fang**[1*]
Kanle Shi[2] **Shenkun Xu**[2] **Yu-Shen Liu**[1†]

[1]School of Software, Tsinghua University, Beijing, China [2]Kuaishou Technology, Beijing, China
{zwq23, zhou-js24, fangjt21}@mails.tsinghua.edu.cn
{shikanle,xushenkun}@kuaishou.com liuyushen@tsinghua.edu.cn



Figure 1: Given a reference image and text description, TESLA generates native 4D Gaussian Splatting with realistic and temporally coherent animations.

Abstract

Diffusion models have achieved remarkable success in image, video, and 3D content creation. However, extending these successes to native 4D model generation remains challenging due to the intricate coupling between temporal geometry and appearance dynamics. Existing approaches typically generate 4D content by synthesizing videos from novel viewpoints, yet they often suffer from poor spatio-temporal consistency across views and time. To address these limitations, we introduce TESLA, a novel feedforward approach for generating native 4D Gaussian Splatting (4DGS) representations. Our key insight is to decouple temporal geometry and appearance generation through a temporally structured latent representation. This decomposition allows us to first establish coarse temporal geometry that captures fundamental structural dynamics, and then synthesize fine-grained appearance details conditioned on this geometric scaffold. To bridge this decoupled representation to the final output, we design a native 4DGS decoder that directly transforms our temporal features into native 4DGS representations, enabling flexible temporal rendering across arbitrary viewpoints and timesteps. Trained on a large collection of curated animated sequences, we extensively evaluate TESLA on the 4DGS generation task, showing superior performance over existing baselines. Project Page: https://weiqi-zhang.github.io/TESLA.

*Equal contribution. †Corresponding authors.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1 Introduction

Dynamic 4D content generation represents a cornerstone technology for next generation digital experiences across AR/VR, gaming, and digital media, enabling the creation of temporally coherent dynamic objects that evolve naturally over time. While recent advances in diffusion models have demonstrated remarkable progress in static 3D object generation [39, 12, 46, 63, 64], extending these achievements to the temporal domain remains fundamentally challenging. The key difficulties lie in modeling spatio-temporal dynamics, maintaining inter-frame geometric and appearance consistency, and managing the exponential growth in computational complexity introduced by temporal dimensions.

Various representation forms exist for modeling 4D dynamic scenes, such as animated mesh and deformable 3DGS. More recently, 4D Gaussian Splatting [47] has emerged as a promising 4D representation that offers unique advantages over existing approaches. While mesh-based representations are constrained by fixed topology and skeletal structures, 4D Gaussian Splatting naturally handles complex deformations, appearance changes, and topological variations through independent Gaussian evolution.

Current 4DGS generation approaches fall into two primary categories. The first employs gradient optimization using powerful 2D diffusion models as supervisory signals [34]. These methods [1, 36, 44, 61] iteratively refine 4D representations by distilling knowledge from pre-trained generative models, but suffer from computational intensity and optimization instabilities that cause temporal flickering and geometric inconsistencies. The second category employs reconstruction pipelines that generate multi-view videos before reconstructing 4D Gaussian representations. These approaches [24, 18, 42] suffer from viewpoint inconsistencies and temporal artifacts that propagate through reconstruction. More fundamentally, native 4DGS generation faces inherent challenges from high-dimensional spatio-temporal data where each Gaussian primitive must maintain temporal coherence across the entire sequence.

To overcome these fundamental challenges, we introduce **TESLA**, a novel feedforward approach that natively generates 4DGS representations without relying on intermediate reconstruction or optimization. The key insight of TESLA lies in a temporally structured latent representation that decouples geometric evolution from appearance synthesis. By first establishing coarse temporal geometry that captures fundamental structural changes and then conditioning fine-grained appearance synthesis on this geometric scaffold, TESLA effectively reduces the complexity of high-dimensional spatio-temporal modeling.

In addition to the decoupled paradigm, TESLA introduces a native 4DGS decoder that directly predicts Gaussian parameters from temporally structured latent representations, thereby eliminating intermediate reconstruction stages. This native generation scheme enables arbitrary temporal sampling during rendering, free from the constraints of discrete video frames. Moreover, the decoder integrates temporal attention mechanisms to enhance temporal coherence and appearance consistency across the generated sequence.

We curate a comprehensive dataset comprising **45,000** animated sequences collected from Deformthings4D [22], Objaverse [8] and Objaverse-XL [7], totaling over **1.08 million** frames across diverse object categories. Each frame is rendered from multiple viewpoints to provide rich multi-view supervision. The entire curation and rendering process requires approximately **50,000** GPU hours, resulting in a large-scale foundational dataset that we expect to benefit future research in 4D content generation. Trained on this dataset, TESLA achieves significant improvements over existing methods, with extensive evaluation demonstrating superior animation quality and computational efficiency compared to prior approaches.

Our primary contributions include:

- We introduce TESLA, a novel feedforward framework for generating 4D content as native 4D Gaussian Splatting representation. TESLA is designed based on temporally structured latents that effectively decouples temporal geometry and appearance dynamics, simplifying the inherent complexity of 4D data modeling.
- We develop a native 4D Gaussian Splatting decoder that directly predicts Gaussian parameters from temporal latents, enabling flexible and efficient 4DGS generation.

- We contribute a meticulously curated dataset of **45,000** animated sequences requiring about **50,000** GPU hours of rendering, with comprehensive evaluations demonstrating TESLA's superior performance in generating high-quality 4DGS representations.

2 Related Work

2.1 Native 3D Generation

The field of 3D content generation has evolved significantly, transitioning from direct manipulation of raw 3D data to sophisticated latent-based approaches. Initial methodologies operated directly on explicit 3D representations including point clouds [28], volumetric grids [11], and implicit neural representations such as Neural Radiance Fields [27, 3, 10]. While these direct approaches were conceptually straightforward, they struggled with scalability and poor generalization due to limited datasets and computational constraints.

This field has increasingly adopted latent-based denoising approaches for 3D content creation [43, 21, 62, 23], utilizing variational autoencoders to encode high-dimensional geometric information into compact latent representations that facilitate efficient generative modeling. Early works [52, 4] pioneered vector-based latent encoding strategies for geometric data compression and reconstruction. CLAY [55] established the viability of merging VAE architectures with transformer-driven diffusion processes, achieving competitive performance on large-scale datasets with flexible input modalities. TripoSG [23] advanced the field by implementing rectified flow techniques [26] alongside mixture-of-experts designs to accelerate generation workflows while maintaining quality. Trellis [46] introduced hierarchical sparse encoding schemes coupled with adaptive decoding modules to enhance flow-model training efficiency and representation quality. While these latent-based 3D generation methods, together with methods that generate 3D Gaussians [63, 59, 58] and unsigned distance fields [64, 65], have achieved remarkable success in creating static geometric content, extending these achievements to temporal dynamics and 4D generation presents fundamental challenges. The addition of temporal dimension introduces complexities in modeling coherent motion patterns, maintaining inter-frame consistency, and handling the exponential growth in data dimensionality across time-varying scenes. Reconstruction methods built on Gaussian Splatting [56, 20] or signed distance functions [29, 30] likewise target static scenes and shapes.

2.2 4D Generation

4D content generation represents a frontier challenge in generative modeling, requiring the synthesis of both geometric structure and temporal dynamics within a unified framework. Early approaches have predominantly employed distillation techniques that leverage knowledge from established 2D, 3D, or video generation models [1, 15, 25, 36, 61] to guide spatio-temporal synthesis. These distillation-based methods, however, impose significant computational burdens and necessitate lengthy optimization procedures for individual scenes, often resulting in temporal artifacts and spatial inconsistencies that degrade generation quality. More recent work has investigated fine-tuning strategies applied to existing 3D and video generation architectures using available 4D datasets, enabling the production of multi-view dynamic sequences [24, 18, 42, 49, 32, 16, 51] with reduced computational requirements and improved temporal stability. Nevertheless, these approaches remain limited by their reliance on per-scene reconstruction workflows and exhibit spatial discrepancies due to the scarcity of high-quality 4D training data. Feed-forward 4D reconstruction models [37, 60, 9, 5] avoid per-scene optimization, but they recover dynamic content from an input video rather than generating it. Moving-region segmentation under camera motion [17] likewise analyzes motion already present in a video. Recently, AnimateAnyMesh [45] pioneered feed-forward mesh animation through trajectory decomposition and topology-aware encoding. GVDiffusion [53] proposed a Gaussian Variation Field approach that efficiently compresses 4D mesh animations into compact latent spaces and enables video-to-4D generation through diffusion modeling. However, these methods primarily require intermediate conversions, and cannot directly generate native 4D Gaussian Splatting with flexible rendering. Concurrently, AEON [57] decodes video diffusion latents into 4D Gaussians for scene-level world modeling, whereas TESLA generates object-level animations natively from temporally structured latents.

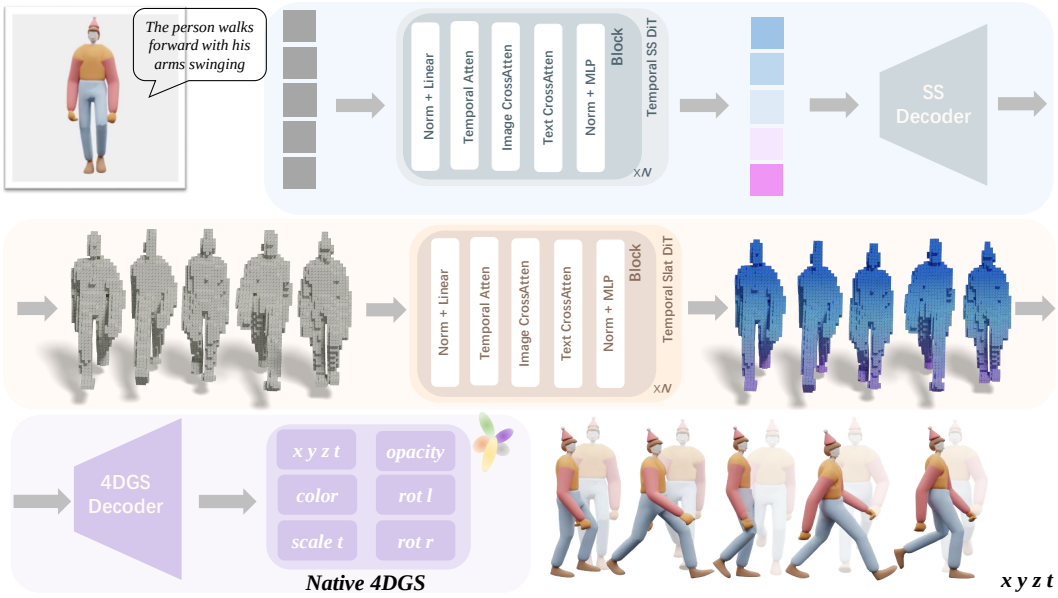


Figure 2: Overview of TESLA for native 4D Gaussian Splatting generation. Unlike methods that require intermediate video reconstruction or per-scene optimization, TESLA directly generates native 4DGS in a feedforward manner. Given a reference image and text prompt, the framework consists of three stages: (1) The inputs are encoded into multi-modal conditioning features. (2) A Temporal Slat DiT with parallel attention architecture generates structured temporal latents, where Temporal SS DiT first produces sparse geometric structures, followed by dense appearance synthesis. (3) The 4DGS Decoder directly transforms temporal latents into native 4D Gaussian parameters, enabling flexible temporal rendering without being constrained to discrete video frames. The parallel attention mechanism processes image and text conditions simultaneously to ensure effective multi-modal guidance while maintaining temporal coherence.

3 Method

We introduce TESLA, a novel framework that generates native 4D Gaussian Splatting representations through a feedforward approach that decouples temporal geometry and appearance dynamics. Given a reference image I and descriptive text prompt P, our goal is to generate temporally coherent 4DGS representations G. The overview of TESLA is shown in Fig. 2. We begin by previewing 4D Gaussian Splatting and introducing data preparation in Sec. 3.1. In Sec.3.2, we present the encoding and decoding framework that establishes a structured latent and enables native 4DGS generation. We further detail the temporal generation approach that synthesizes temporally structured latents through diffusion transformers [33] conditioned on multi-modal inputs in Sec. 3.3.

3.1 Preliminary Knowledge

Preview 4D Gaussian Splatting. 4D Gaussian Splatting (4DGS) [47] extends 3D Gaussian Splatting to dynamic scenes by representing the underlying spatio-temporal 4D volume as a collection of 4D Gaussians that capture both temporal geometry evolution and time-varying view-dependent appearances. For a 4DGS representation $G = \{gi\}_{i=1}^{N}$, each Gaussian g_i models the spatio-temporal distribution through a 4D covariance matrix $\Sigma_i \in R^{4\times4}$ and center $\mu_i \in R^4$, formulated as:

$$g_i(x, y, z, t) = \exp\left[-\tfrac{1}{2}(\xi - \mu_i)^{\mathrm{T}}\Sigma_i^{-1}(\xi - \mu_i)\right], \tag{1}$$

where $\xi = [x, y, z, t]^\top$ denotes the 4D spatio-temporal coordinates. Following the factorization strategy of 3DGS, the covariance matrix is decomposed as

$$\Sigma_i = R_i\, S_i S_i^\top R_i^\top, S_i = \mathrm{diag}(s_x, s_y, s_z, s_t), \tag{2}$$

where R_i represents a 4D rotation matrix and S_i encodes anisotropic scaling across spatial and temporal dimensions. Crucially, the 4D rotation R_i is parameterized using left and right isoclinic

rotations via quaternion pairs $(\mathbf{q}_\ell, \mathbf{q}_r)$, which enables the Gaussian to undergo arbitrary rotations in 4D space and naturally capture complex temporal dynamics.

During rendering at a specific time t, each 4D Gaussian undergoes probabilistic decomposition into a conditional 3D Gaussian $p(x, y, z \mid t)$ that governs spatial geometry and a marginal 1D temporal Gaussian $p(t)$ that provides time-dependent weighting. This decomposition allows the rendering pipeline to leverage efficient 3D splatting while incorporating temporal coherence. For appearance modeling, each Gaussian maintains an opacity value $\alpha_i \in \mathbb{R}$ and employs time-evolving view-dependent color $\mathbf{c}_i(\mathbf{d}, t) \in \mathbb{R}^3$, where $\mathbf{d}$ denotes the viewing direction.

3.2 Encoding and Decoding for 4DGS

Temporal Latent Representation. 4D Gaussian Splatting generation faces inherent complexity due to the intricate coupling between geometric motion and appearance variations across time. The established paradigms in image-to-video generation fundamentally rely on temporalizing spatial representations, which proves inadequate for handling such complex spatiotemporal dynamics. Inspired by the success of sparse structural approaches in 3D generation [46], we extend the structured latent paradigm to the temporal domain to effectively address this limitation. Our approach introduces a temporal structured latent representation that organizes 4D information across both spatial and temporal dimensions. We construct a temporal sequence of structured latents $S = \{S_t\}_{t=1}^{T}$, where each $S_t = \{(z_{i,t}, p_{i,t})\}_{i=1}^{L_t}$ represents the structured latent at time step t. Here, $p_{i,t} \in 0, 1, \ldots, (N-1)^3$ denotes the positional indices of active voxels within the 3D grid occupied by the object at time t, capturing geometric structure evolution, while $z_{i,t} \in \mathbb{R}^C$ represents the local latent features attached to these voxels, encoding detailed appearance information that varies across space and time.

Decoding For 4DGS. Given the temporal structured latent representation $S = \{S_t\}_{t=1}^{T}$, we design a native 4DGS decoder to transform these temporally structured latents directly into native 4D Gaussian primitives without intermediate reconstruction stages. Built upon Transformer [41], the decoder employs temporal self-attention layers to capture motion dynamics and ensure temporal consistency across frames while modeling the evolution of Gaussian attributes including 4D positions, scales, rotations, colors, and opacities. Since the temporal latents inherently encode sequential information across time steps, our decoder preserves the temporal information of each Gaussian while additionally learning a temporal offset parameter $\Delta t_i^{(t)}$ that captures the dynamic motion patterns of individual Gaussians. The decoder output can be formulated as:

$$\mathcal{G} = \{G_t\}_{t=1}^{T} = \text{Decoder}(S), \tag{3}$$

$$G_t = (\mu_i^{(t)}, \mathbf{s}_i^{(t)}, \mathbf{q}_{\ell,i}^{(t)}, \mathbf{q}_{r,i}^{(t)}, \mathbf{c}_i^{(t)}, \alpha_i^{(t)}, \Delta t_i^{(t)})_{i=1}^{N}. \tag{4}$$

where $\mu_i^{(t)} \in \mathbb{R}^4$ denotes the 4D center position (x, y, z, t), $\mathbf{s}_i^{(t)} \in \mathbb{R}^4$ represents the 4D scale parameters (s_x, s_y, s_z, s_t), $\mathbf{q}_{\ell,i}^{(t)}, \mathbf{q}_{r,i}^{(t)} \in \mathbb{R}^4$ are the left and right quaternion rotation pairs, $\mathbf{c}_i^{(t)} \in \mathbb{R}^3$ indicates the color, $\alpha_i^{(t)} \in \mathbb{R}$ represents the opacity, and $\Delta t_i^{(t)} \in \mathbb{R}$ denotes the learned temporal offset that enables flexible motion modeling for the i-th Gaussian at time t. Each temporal latent S_t is processed through multiple transformer blocks that incorporate positional embeddings to maintain spatial-temporal correspondence, enabling the direct synthesis of coherent 4D Gaussian Splatting sequences with enhanced motion representation capabilities.

Training Loss. The design of training objective focuses on two fundamental challenges in 4D generation: achieving high-quality visual synthesis and maintaining motion consistency. We design a multi-component loss function that systematically tackles these objectives.

For visual quality, we employ a comprehensive reconstruction loss that captures multiple aspects of image fidelity:

$$\mathcal{L}_{\text{recon}} = \mathcal{L}_1 + 0.2(1 - \text{SSIM}) + 0.2\mathcal{L}_{\text{LPIPS}}. \tag{5}$$

For motion consistency, we introduce a motion regularization loss that directly constrains Gaussian velocity to prevent erratic movements:

$$\mathcal{L}_{\text{motion}} = \frac{1}{N} \sum_{i=1}^{N} |v_i|_2. \tag{6}$$

Additionally, we include two auxiliary regularization terms to maintain stable Gaussian properties. The volume regularization loss is defined as:

$$\mathcal{L}_{\text{vol}} = \frac{1}{LK} \sum_{i=1}^{L} \sum_{k=1}^{K} s_k^i, \tag{7}$$

which prevents excessive Gaussian concentration. The opacity regularization loss is:

$$\mathcal{L}_{\alpha} = \frac{1}{LK} \sum_{i=1}^{L} \sum_{k=1}^{K} (1 - \alpha_k^i)^2, \tag{8}$$

which constrains Gaussians from being excessively transparent. The complete training objective is:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{recon}} + \lambda_{\text{motion}} \mathcal{L}_{\text{motion}} + \lambda_{\text{vol}} \mathcal{L}_{\text{vol}} + \lambda_{\alpha} \mathcal{L}_{\alpha}. \tag{9}$$

3.3 Temporal Latent Generation

Rectified Flow Model. To generate temporal slat representations, we employ a Rectified Flow model [26] as our core generative framework. Unlike traditional diffusion models such as DDPM [13] that rely on complex noise schedules, Rectified Flow learns straight-line trajectories from noise to data distributions, offering superior efficiency and training stability.

Given target temporal latent representations $S = \{S_t\}_{t=1}^{T}$, our objective is to train a neural network $v_\theta(x, t, I, P)$ that predicts the velocity field for transforming noise $x_0 \sim \mathcal{N}(0, I)$ to target data $x_1 = S$, where I represents the reference image and P denotes the text prompt describing the desired animation. The Rectified Flow training objective is formulated as:

$$\mathcal{L}_{RF} = \mathbb{E}_{t \sim \mathcal{U}[0,1], x_0, x_1} \| v_\theta(x_t, t, I, P) - (x_1 - x_0) \|^2, \tag{10}$$

where $x_t = t x_1 + (1 - t) x_0$ represents linear interpolation along the optimal transport path. This formulation encourages direct mapping between noise and structured temporal representations conditioned on both visual and textual information, significantly reducing computational overhead during inference while ensuring temporal coherence guided by the reference image appearance and text-described motion dynamics.

Model Architecture. Our temporal latent generation model builds upon the Diffusion Transformer architecture [33], adapted for dual-conditioned temporal synthesis. The model generates temporal latent embeddings conditioned on reference images and text prompts through three key architectural innovations designed for effective 4D generation.

Temporal Attention Mechanism is designed to maintain temporal coherence in 4DGS generation by introducing dedicated temporal self-attention layers that operate across the time dimension. By treating spatial dimensions as independent sequences, this mechanism focuses specifically on temporal relationships and computes attention weights between different timesteps. This design enables the model to capture long-range temporal dependencies and ensure smooth motion transitions throughout the generation process.

Parallel Attention Architecture. To enable effective multi-modal conditioning, a parallel attention mechanism is adopted, in which image and text guidance are processed simultaneously through separate cross-attention branches. The image cross-attention branch preserves appearance consistency based on reference visuals, while the text cross-attention branch captures motion semantics conveyed by the textual descriptions.

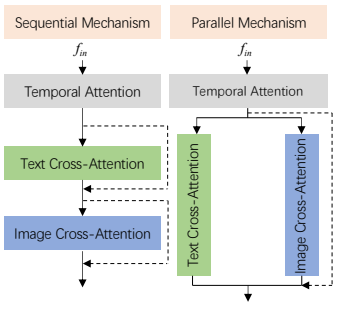


Figure 3: Attention mechanisms for multi-modal conditioning.

In contrast to sequential conditioning approaches, which process modalities in order and may incur information bottlenecks due to inter-modal dependencies, this architecture adopts a parallel structure. Temporal attention is applied to model motion dynamics:

$$f_{\text{temp}} = \text{TemporalAttn}(f_{\text{in}}) + f_{\text{in}}, \tag{11}$$

Subsequently, both text and image conditioning are applied in parallel:

$$f_{\text{out}} = \text{TextCrossAttn}(f_{\text{temp}}, f_{\text{text}}) + \text{ImageCrossAttn}(f_{\text{temp}}, f_{\text{ref}}) + f_{\text{temp}}, \tag{12}$$

Table 1: Quantitative comparison on animation generation. Objective metrics are from VBench; user-study scores (1–5) are averaged over 20 participants on motion–text alignment (M.Match), realism (M.Real.), and geometric consistency (G.Cons.).

	Objective Metrics				User Study (1–5)		
Method	I2V ↑	M.Sm. ↑	Sub.Cons. ↑	Aest.Q ↑	M.Match ↑	M.Real. ↑	G.Cons. ↑
Animate3D [14]	0.663	0.990	0.727	0.420	1.850	1.717	1.917
AnimateAnyMesh [45]	0.850	0.993	0.911	0.532	3.217	3.483	3.350
Ours	**0.861**	**0.994**	**0.941**	**0.533**	**3.850**	**3.775**	**3.867**

where f_{ref} and f_{text} denote the reference image features and text embeddings, respectively. This parallel formulation allows each modality to attend independently to the temporally enriched features, thereby avoiding sequential dependencies and facilitating more effective multi-modal fusion for temporal generation. We further validate this parallel design against a sequential conditioning baseline in Tab. 5.

Spatio-Temporal Position Encoding employs a hybrid $xyzt$ position embedding scheme that captures both spatial coordinates within each timestep and temporal progression across the sequence. This enables the attention layers to understand spatio-temporal correlations, which is essential for coherent 4D generation. The full two-stage inference procedure is detailed in Appendix A.5.

4 Experiments

Implementation Details. We train TESLA using NVIDIA A800 GPUs across three stages. The 4DGS decoder is trained on 8 GPUs for 5 days, while the temporal SS flow model and slat flow model are trained on 32 GPUs for 8 days and 10 days respectively. For data preparation, we employ Qwen2.5-VL [2] to generate detailed captions for each animated sequence. This automatic captioning process ensures comprehensive text-image-motion alignment across our dataset of 45,000 sequences. For the loss function, we set $\lambda_{\text{motion}} = 0.1$, $\lambda_{\text{vol}} = 10000.0$, and $\lambda_{\alpha} = 0.1$. The temporally structured latent representation uses a grid resolution of $R = 64$, generating sequences of $T = 24$ frames during training. Each frame is rendered from 36 or 48 uniformly sampled viewpoints to provide multi-view supervision.

4.1 Animation Generation

Evaluation and Metrics. Following the evaluation protocol established in Animate3D [14] and AnimateAnyMesh [45], we conduct both objective measurements and subjective assessments. For objective evaluation, we utilize three perceptual metrics from the VBench suite [54], consistent with the assessment framework in Animate3D [14]: I2V Subject Similarity, Aesthetic Quality, Motion Smoothness and Subject Consistency. These metrics are computed on view-aligned renderings by rendering the generated 4D Gaussian Splatting from predetermined camera viewpoints.

To supplement our quantitative evaluation, we conducted a perceptual study involving 20 participants. Each participant assessed the generated animations using a five-point rating system, with higher scores reflecting superior quality. The evaluation focused on three key aspects: how well the motion matched the textual description, the realism of the movements, and the consistency of geometric structure throughout the animation.

Baselines. Following the data setting of [14, 45], we collect 30 test cases that span different object categories and motion complexities for evaluation. We compare TESLA against two representative animation generation methods using a reference image and text prompt as input. Since AnimateAnyMesh [45] requires mesh input rather than images, we first generate a 3D mesh from the reference image using Trellis [46], followed by mesh simplification [19] to satisfy the vertex count constraints imposed by their architecture. The method then produces animated meshes through trajectory decomposition and topology aware encoding. Animate3D [14] similarly leverages Trellis generated 3DGS as initialization, employing a video diffusion model finetuned on 3D data to synthesize multi-view consistent dynamic sequences, subsequently reconstructing 4D representations through optimization.

Comparison. As shown in Tab. 1, TESLA consistently outperforms all baselines across both objective metrics and subjective evaluations. Unlike mesh-based methods that must move vertices while

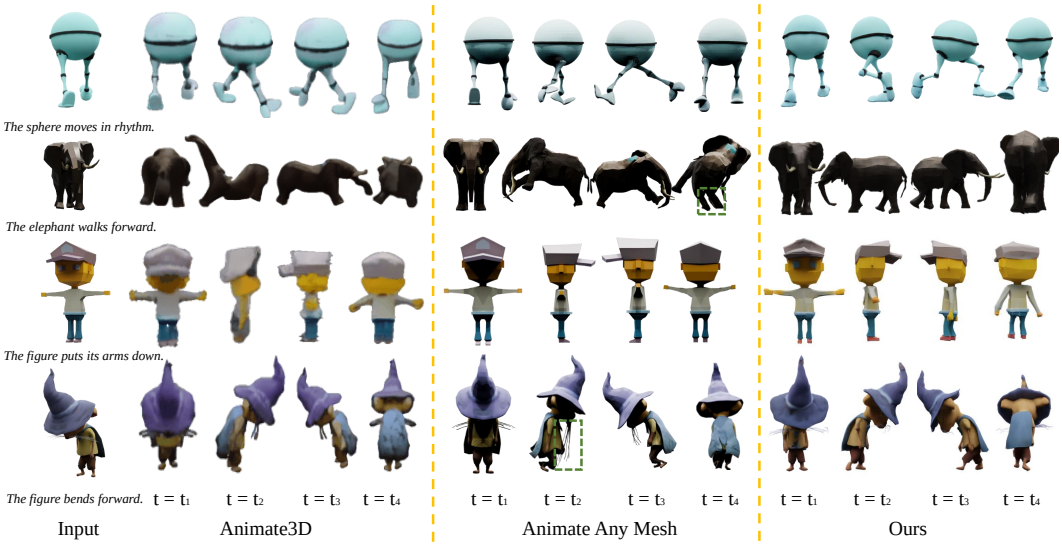


Figure 4: Qualitative comparison with baseline methods across diverse animation scenarios. TESLA produces smoother motions with better geometric stability. In contrast, AnimateAnyMesh shows limited deformation due to topology preservation challenges in vertex-based mesh animation, while Animate3D exhibits temporal flickering.

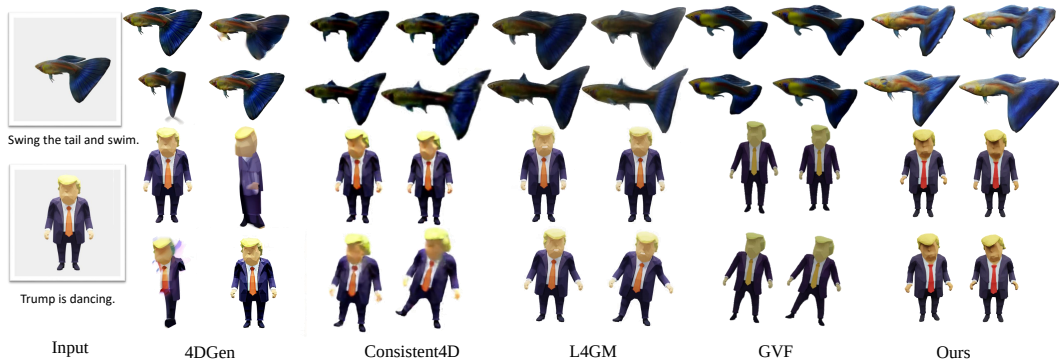


Figure 5: Qualitative comparison of 4DGS generation on Consistent4D dataset from text prompts across different methods.

preserving topology, our approach naturally supports flexible deformation without such constraints. This enables smoother and more realistic animations, particularly for complex articulated motions. Fig. 4 illustrates this advantage. TESLA produces smooth animations with stable geometry, while Animate3D suffers from temporal flickering and AnimateAnyMesh exhibits limited deformation. The walking elephant example shows TESLA generating realistic gait patterns, whereas baselines display artifacts or stiffness. Complementary motion-centric metrics, including motion stability, prompt-conditioned diversity, and inter-frame consistency, are reported in Appendix C.5.

4.2 4DGS Generation

Datasets and Metrics. We follow the evaluation protocol from Consistent4D [15]. For evaluation, we adopt three complementary metrics: CLIP Score [35] measures semantic alignment between rendered animations and text prompts, FVD [40] evaluates the realism and quality of generated motion by comparing feature distributions with real sequences, and generation time quantifies computational efficiency on a single NVIDIA 4090 GPU.

Baselines. We compare TESLA against four state-of-the-art 4D generation methods: 4DGen [50], Consistent4D [15], L4GM [37], and GVF [53]. Since TESLA natively accepts image and text inputs, we can directly evaluate 4DGen using the same input modality. However, the remaining baselines require video inputs for 4D reconstruction. To ensure fair comparison, we first generate multi-view video sequences from the reference image and text prompt using Wan2.1 [38], a state-of-the-art video

Config.	PSNR ↑	SSIM ↑	LPIPS ↓
w/o $\mathcal{L}_{\text{mot}}$	29.41	0.966	0.0417
w/o $\mathcal{L}_{\text{vol}}$	30.17	0.972	0.0393
Full	**31.82**	**0.979**	**0.0285**

Table 3: VAE Loss Function Components.



Figure 6: Visualizations on ablation study.

diffusion model. These generated videos are then fed into Consistent4D, L4GM, and GVF to produce their corresponding 4D representations.

Comparison. As shown in Tab. 2, TESLA achieves superior performance across all metrics while maintaining competitive generation speed. Our method demonstrates better semantic alignment with text prompts and generates more realistic motion with enhanced temporal coherence. Moreover, TESLA achieves competitive generation speed comparable to feed-forward methods while substantially faster than optimization-based approaches [50, 15]. Fig. 5 illustrates qualitative comparisons. TESLA produces smoother animations with stable geometry, while baselines suffer from temporal flickering or geometric artifacts. Additional qualitative comparisons against state-of-the-art baselines are provided in the appendix (Fig. 10).

Table 2: Quantitative comparison on 4DGS generation. * indicates methods that require video inputs, which we generate using Wan2.1 [38].

Method	CLIP ↑	FVD ↓	Time ↓
4DGen [50]	25.06	1713.2	∼1.5h
Consistent4D* [15]	25.50	992.49	∼1.6h
L4GM* [37]	25.72	<u>876.25</u>	**4.5s**
GVF* [53]	<u>25.75</u>	891.38	15.5s
Ours	**25.98**	**778.43**	<u>9.2s</u>

4.3 Ablations and Further Analysis

VAE Architecture Ablations. We systematically evaluate the contribution of each regularization term in our VAE training objective to understand their individual and combined effects on reconstruction quality. As demonstrated in Tab. 3, each component plays a distinct role in ensuring high-quality 4DGS generation. Specifically, motion regularization $\mathcal{L}_{\text{motion}}$ maintains temporal stability by directly constraining Gaussian velocities, preventing erratic movements across frames. Without this constraint, individual Gaussians may exhibit unpredictable motion patterns that break temporal coherence and lead to visual artifacts. Volume regularization $\mathcal{L}_{\text{vol}}$ prevents excessive expansion of Gaussian primitives by constraining their scales, thereby maintaining reasonable sizes throughout the animation sequence. Fig. 6 visualizes these effects: without $\mathcal{L}_{\text{motion}}$, the generated animations exhibit significant temporal jittering and inconsistent frame-to-frame transitions, whereas our full model produces smooth, coherent animations with stable geometric structures.

Generalization. To evaluate TESLA's generalization beyond the curated training distribution, we examine two complementary aspects: motion diversity and in-the-wild applicability. Fig. 7 visualizes multi-motion generation from the same reference image conditioned on different text prompts, alongside test cases drawn from real-world sources outside the training data. TESLA produces temporally coherent and semantically aligned animations across both settings, demonstrating that the temporally structured latent space captures a rich distribution of motion patterns rather than collapsing to repetitive behaviors. We further quantify motion diversity in Appendix C.5 to verify that TESLA learns diverse motions despite the limited scale of available 4D animation data. These results suggest that the decoupled geometry and appearance representation has internalized generalizable spatiotemporal priors that extend well beyond the training distribution.

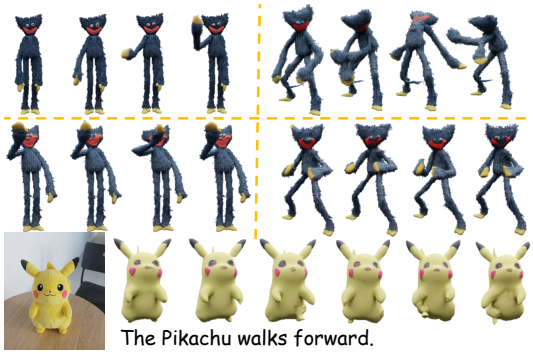


Figure 7: Motion diversity and in-the-wild generalization.

5 Conclusion

In this work, we present TESLA, a novel feedforward approach for native 4D Gaussian Splatting generation. We leverage a diffusion model to learn distributions of temporally structured latent representations that decouple geometry evolution from appearance dynamics, enabling effective spatio-temporal modeling. Our native 4DGS decoder directly transforms temporal latents into Gaussian primitives. Evaluations on animation generation and 4DGS synthesis benchmarks demonstrate our superior performance over state-of-the-art methods in generating high-quality dynamic content with enhanced temporal coherence. We discuss limitations and broader impact in Appendix E.

Acknowledgments and Disclosure of Funding

This work was partially supported by Deep Earth Probe and Mineral Resources Exploration – National Science and Technology Major Project (2024ZD1003405) and the National Natural Science Foundation of China (62272263), and in part by Kuaishou.

References

[1] Sherwin Bahmani, Ivan Skorokhodov, Victor Rong, Gordon Wetzstein, Leonidas Guibas, Peter Wonka, Sergey Tulyakov, Jeong Joon Park, Andrea Tagliasacchi, and David B. Lindell. 4D-fy: Text-to-4D generation using hybrid score distillation sampling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[2] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-VL technical report. *arXiv preprint arXiv:2502.13923*, 2025.

[3] Ziang Cao, Fangzhou Hong, Tong Wu, Liang Pan, and Ziwei Liu. Large-vocabulary 3D diffusion model with transformer. In *International Conference on Learning Representations (ICLR)*, 2024.

[4] Rui Chen, Jianfeng Zhang, Yixun Liang, Guan Luo, Weiyu Li, Jiarui Liu, Xiu Li, Xiaoxiao Long, Jiashi Feng, and Ping Tan. Dora: Sampling and benchmarking for 3D shape variational auto-encoders. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[5] Tiancheng Chen, Sheng Tang, Wenhua Jin, Weiqi Zhang, Juntong Fang, Junsheng Zhou, and Zesong Li. UniQuery4R: Unified 4D scene reconstruction from a single query. *arXiv preprint arXiv:2608.17283*, 2026.

[6] Cecilia Curreli, Dominik Muhle, Abhishek Saroha, Zhenzhang Ye, Riccardo Marin, and Daniel Cremers. Nonisotropic Gaussian diffusion for realistic 3D human motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[7] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhak Gadre, Eli VanderBilt, Aniruddha Kembhavi, Carl Vondrick, Georgia Gkioxari, Kiana Ehsani, Ludwig Schmidt, and Ali Farhadi. Objaverse-XL: A universe of 10M+ 3D objects. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[8] 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 (CVPR)*, 2023.

[9] Juntong Fang, Zequn Chen, Weiqi Zhang, Donglin Di, Xuancheng Zhang, Chengmin Yang, and Yu-Shen Liu. MoRe: Motion-aware feed-forward 4D reconstruction transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

[10] Jun Gao, Tianchang Shen, Zian Wang, Wenzheng Chen, Kangxue Yin, Daiqing Li, Or Litany, Zan Gojcic, and Sanja Fidler. GET3D: A generative model of high quality 3D textured shapes learned from images. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[11] Anchit Gupta, Wenhan Xiong, Yixin Nie, Ian Jones, and Barlas Oğuz. 3DGen: Triplane latent diffusion for textured mesh generation. *arXiv preprint arXiv:2303.05371*, 2023.

[12] Xianglong He, Zi-Xin Zou, Chia-Hao Chen, Yuan-Chen Guo, Ding Liang, Chun Yuan, Wanli Ouyang, Yan-Pei Cao, and Yangguang Li. SparseFlex: High-resolution and arbitrary-topology 3D shape modeling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

[13] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[14] Yanqin Jiang, Chaohui Yu, Chenjie Cao, Fan Wang, Weiming Hu, and Jin Gao. Animate3D: Animating any 3D model with multi-view video diffusion. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[15] Yanqin Jiang, Li Zhang, Jin Gao, Weiming Hu, and Yao Yao. Consistent4D: Consistent 360° dynamic object generation from monocular video. In *International Conference on Learning Representations (ICLR)*, 2024.

[16] Zeren Jiang, Chuanxia Zheng, Iro Laina, Diane Larlus, and Andrea Vedaldi. Geo4D: Leveraging video generators for geometric 4D scene reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

[17] Wenhua Jin, Tiancheng Chen, Sheng Tang, Jiahao Li, Yupeng Liu, Weiqi Zhang, Juntong Fang, and Zesong Li. Moving regions, not moving objects: Segmenting what actually moves under camera motion. *arXiv preprint arXiv:2610.31209*, 2026.

[18] Yudong Jin, Sida Peng, Xuan Wang, Tao Xie, Zhen Xu, Yifan Yang, Yujun Shen, Hujun Bao, and Xiaowei Zhou. Diffuman4D: 4D consistent human view synthesis from sparse-view videos with spatio-temporal diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

[19] Julian Knodt. Single edge collapse quad-dominant mesh reduction. *ACM Transactions on Graphics (TOG)*, 44(4):145:1–145:23, 2025.

[20] Shujuan Li, Yu-Shen Liu, and Zhizhong Han. GaussianUDF: Inferring unsigned distance functions through 3D Gaussian Splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[21] Weiyu Li, Jiarui Liu, Hongyu Yan, Rui Chen, Yixun Liang, Xuelin Chen, Ping Tan, and Xiaoxiao Long. CraftsMan3D: High-fidelity mesh generation with 3D native diffusion and interactive geometry refiner. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[22] 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 (ICCV)*, 2021.

[23] Yangguang Li, Zi-Xin Zou, Zexiang Liu, Dehu Wang, Yuan Liang, Zhipeng Yu, Xingchao Liu, Yuan-Chen Guo, Ding Liang, Wanli Ouyang, and Yan-Pei Cao. TripoSG: High-fidelity 3D shape synthesis using large-scale rectified flow models. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2025.

[24] 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 (NeurIPS)*, 2024.

[25] Huan Ling, Seung Wook Kim, Antonio Torralba, Sanja Fidler, and Karsten Kreis. Align Your Gaussians: Text-to-4D with dynamic 3D Gaussians and composed diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[26] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. In *International Conference on Learning Representations (ICLR)*, 2023.

[27] 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. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[28] Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. Point-E: A system for generating 3D point clouds from complex prompts. *arXiv preprint arXiv:2212.08751*, 2022.

[29] Takeshi Noda, Chao Chen, Weiqi Zhang, Xinhai Liu, Yu-Shen Liu, and Zhizhong Han. MultiP-ull: Detailing signed distance functions by pulling multi-level queries at multi-step. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[30] Takeshi Noda, Chao Chen, Junsheng Zhou, Weiqi Zhang, Yu-Shen Liu, and Zhizhong Han. Learning bijective surface parameterization for inferring signed distance functions from sparse point clouds with grid deformation. In *Proceedings of the IEEE/CVF Conference on Computer*

Vision and Pattern Recognition (CVPR), 2025.

[31] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research (TMLR)*, 2024.

[32] Panwang Pan, Chenguo Lin, Jingjing Zhao, Chenxin Li, Yuchen Lin, Haopeng Li, Honglei Yan, Kairun Wen, Yunlong Lin, Yixuan Yuan, and Yadong Mu. Diff4Splat: Controllable 4D scene generation with latent dynamic reconstruction models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.

[33] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[34] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. DreamFusion: Text-to-3D using 2D diffusion. In *International Conference on Learning Representations (ICLR)*, 2023.

[35] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, 2021.

[36] Jiawei Ren, Liang Pan, Jiaxiang Tang, Chi Zhang, Ang Cao, Gang Zeng, and Ziwei Liu. DreamGaussian4D: Generative 4D Gaussian Splatting. *arXiv preprint arXiv:2312.17142*, 2023.

[37] Jiawei Ren, Kevin Xie, Ashkan Mirzaei, Hanxue Liang, Xiaohui Zeng, Karsten Kreis, Ziwei Liu, Antonio Torralba, Sanja Fidler, Seung Wook Kim, and Huan Ling. L4GM: Large 4D Gaussian reconstruction model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[38] Team Wan. Wan: Open and advanced large-scale video generative models. *arXiv preprint arXiv:2503.20314*, 2025.

[39] Tencent Hunyuan3D Team. Hunyuan3D 2.0: Scaling diffusion models for high resolution textured 3D assets generation. *arXiv preprint arXiv:2501.12202*, 2025.

[40] Thomas Unterthiner, Sjoerd van Steenkiste, Karol Kurach, Raphaël Marinier, Marcin Michalski, and Sylvain Gelly. FVD: A new metric for video generation. In *International Conference on Learning Representations Workshops (ICLRW)*, 2019.

[41] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[42] Rundi Wu, Ruiqi Gao, Ben Poole, Alex Trevithick, Changxi Zheng, Jonathan T. Barron, and Aleksander Holynski. CAT4D: Create anything in 4D with multi-view video diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[43] Shuang Wu, Youtian Lin, Feihu Zhang, Yifei Zeng, Yikang Yang, Yajie Bao, Jiachen Qian, Siyu Zhu, Xun Cao, Philip H. S. Torr, and Yao Yao. Direct3D-S2: Gigascale 3D generation made easy with spatial sparse attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[44] Zijie Wu, Chaohui Yu, Yanqin Jiang, Chenjie Cao, Fan Wang, and Xiang Bai. SC4D: Sparse-controlled video-to-4D generation and motion transfer. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024.

[45] Zijie Wu, Chaohui Yu, Fan Wang, and Xiang Bai. AnimateAnyMesh: A feed-forward 4D foundation model for text-driven universal mesh animation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

[46] Jianfeng Xiang, Zelong Lv, Sicheng Xu, Yu Deng, Ruicheng Wang, Bowen Zhang, Dong Chen, Xin Tong, and Jiaolong Yang. Structured 3D latents for scalable and versatile 3D generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[47] Zeyu Yang, Hongye Yang, Zijie Pan, and Li Zhang. Real-time photorealistic dynamic scene representation and rendering with 4D Gaussian Splatting. In *International Conference on Learning Representations (ICLR)*, 2024.

[48] Jiraphon Yenphraphai, Jianqi Chen, Jian Wang, Gordon Qian, Sergey Tulyakov, Rameen Abdal, Raymond A. Yeh, Peter Wonka, and Chaoyang Wang. Helix4D: Complex 4D mesh generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2026.

[49] Minghao Yin, Yukang Cao, Songyou Peng, and Kai Han. Splat4D: Diffusion-enhanced 4D Gaussian Splatting for temporally and spatially consistent content creation. In *ACM SIGGRAPH Conference Papers*, 2025.

[50] Yuyang Yin, Dejia Xu, Zhangyang Wang, Yao Zhao, and Yunchao Wei. 4DGen: Grounded 4D content generation with spatial-temporal consistency. *arXiv preprint arXiv:2312.17225*, 2023.
[51] Mark Yu, Wenbo Hu, Jinbo Xing, and Ying Shan. TrajectoryCrafter: Redirecting camera trajectory for monocular videos via diffusion models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.
[52] Biao Zhang, Jiapeng Tang, Matthias Nießner, and Peter Wonka. 3DShape2VecSet: A 3D shape representation for neural fields and generative diffusion models. *ACM Transactions on Graphics (TOG)*, 42(4):92:1–92:16, 2023.
[53] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.
[54] Fan Zhang, Shulin Tian, Ziqi Huang, Yu Qiao, and Ziwei Liu. Evaluation Agent: Efficient and promptable evaluation framework for visual generative models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025.
[55] Longwen Zhang, Ziyu Wang, Qixuan Zhang, Qiwei Qiu, Anqi Pang, Haoran Jiang, Wei Yang, Lan Xu, and Jingyi Yu. CLAY: A controllable large-scale generative model for creating high-quality 3D assets. *ACM Transactions on Graphics (TOG)*, 43(4):120:1–120:20, 2024.
[56] Wenyuan Zhang, Jimin Tang, Weiqi Zhang, Yi Fang, Yu-Shen Liu, and Zhizhong Han. MaterialRefGS: Reflective Gaussian Splatting with multi-view consistent material inference. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.
[57] Weiqi Zhang, Wenyuan Zhang, Yu-Shen Liu, and Junsheng Zhou. AEON: Unifying video and 3D world models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2026.
[58] Weiqi Zhang, Junsheng Zhou, Haotian Geng, Kanle Shi, Shenkun Xu, Yi Fang, and Yu-Shen Liu. GaussianGrow: Geometry-aware Gaussian growing from 3D point clouds with text guidance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026.
[59] Weiqi Zhang, Junsheng Zhou, Haotian Geng, Wenyuan Zhang, and Yu-Shen Liu. GAP: Gaussianize any point clouds with text guidance. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.
[60] Weiqi Zhang, Junsheng Zhou, Xuancheng Zhang, Juntong Fang, Zequn Chen, Donglin Di, and Yu-Shen Liu. GAMMA: Scalable 4D Gaussian reconstruction model for novel view synthesis of monocular videos. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2026.
[61] Yuyang Zhao, Zhiwen Yan, Enze Xie, Lanqing Hong, Zhenguo Li, and Gim Hee Lee. Animate124: Animating one image to 4D dynamic scene. *arXiv preprint arXiv:2311.14603*, 2023.
[62] Zibo Zhao, Wen Liu, Xin Chen, Xianfang Zeng, Rui Wang, Pei Cheng, Bin Fu, Tao Chen, Gang Yu, and Shenghua Gao. Michelangelo: Conditional 3D shape generation based on shape-image-text aligned latent representation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.
[63] Junsheng Zhou, Weiqi Zhang, and Yu-Shen Liu. DiffGS: Functional Gaussian Splatting diffusion. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.
[64] Junsheng Zhou, Weiqi Zhang, Baorui Ma, Kanle Shi, Yu-Shen Liu, and Zhizhong Han. UDiFF: Generating conditional unsigned distance fields with optimal wavelet diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.
[65] Junsheng Zhou, Weiqi Zhang, Baorui Ma, Kanle Shi, Yu-Shen Liu, and Zhizhong Han. UDFStudio: A unified framework of datasets, benchmarks and generative models for unsigned distance functions. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2026.

A More Implementation Details

A.1 Training Data Preparation

To train TESLA effectively, we curate a comprehensive dataset of animated 4D sequences that provides diverse temporal dynamics and geometric variations. Our dataset construction begins with collecting animated meshes from three primary sources: Deformthings4D [22], Objaverse [8], and Objaverse-XL [7]. We filter sequences based on animation quality and temporal consistency, resulting in 45,000 animated sequences spanning various object categories.

For each filtered sequence, we sample 24 uniformly distributed frames to capture complete temporal evolution while maintaining computational feasibility. At each temporal sample, we render the object from 48 or 36 distinct camera viewpoints. Text descriptions are generated using Qwen2.5-VL [2] to caption the rendered animation sequences, providing accurate semantic descriptions of temporal dynamics and motion patterns. This comprehensive rendering pipeline requires approximately 50,000 GPU hours, resulting in a dataset where each training sample consists of reference images, text descriptions, and multi-view rendered frames across the temporal sequence. Fig. 8 illustrates representative examples from our curated dataset, showcasing the diversity of object categories and motion patterns.

A.2 Temporal Latent Encoding

We directly inherit the encoder of TRELLIS [46] without modification and keep it frozen throughout training. Since the ground-truth sequences are temporally consistent and free of temporal artifacts, we apply this encoder independently at each timestep. At timestep t, we voxelize the frame onto the same R^3 grid used by the latent representation, which gives the active voxel positions $\{p_{i,t}\}$, and render the frame from the 36 or 48 viewpoints described in Sec. A.1. DINOv2 [31] features extracted from each rendered view are projected onto the active voxels and averaged across views, and a sparse VAE encoder then compresses the resulting voxel features into the latents $\{z_{i,t}\}$. Repeating this procedure for $t = 1, \ldots, T$ yields the target latent sequence $S = \{S_t\}_{t=1}^{T}$, which supervises the decoder pre-training and both flow models. Only the 4DGS decoder, the SS flow model, and the slat flow model are optimized (Tab. 4).

Because the encoder is inherited, we make no novelty claim about its design. Encoding frames independently still preserves sufficient information, since decoding the encoded ground-truth latents reaches a PSNR of 31.82, an SSIM of 0.979, and an LPIPS of 0.0285 (Tab. 3). As the encoder has no temporal mechanism of its own, the temporal coherence of generated sequences must come from the downstream models, and removing temporal attention indeed lowers I2V by 0.049 (Tab. 5).

Our 4DGS decoder, unlike the encoder, differs substantially from its TRELLIS counterpart. TRELLIS decodes a static latent into 3D Gaussians, whereas our decoder maps the temporal latent sequence into 4D Gaussians with 4D centers, 4D scales, isoclinic quaternion pairs, and a learned temporal offset Δt, so the output can be rendered at arbitrary continuous timesteps rather than only at discrete frames. Each active voxel is decoded into K Gaussians, indexed by k in Eqs. (7) and (8), whose 4D centers are predicted in continuous space.

A.3 Training Details

We train TESLA in three stages following the pipeline described in the main paper. In all stages, we use AdamW as the optimizer with a learning rate of 1×10^{-4}, $\beta_1 = 0.9$, $\beta_2 = 0.999$, and weight decay of 0.01. The learning rate is scheduled with a cosine decay and a short linear warm-up at the beginning of training. Unless otherwise noted, we sample sequences of 24 frames and render each frame from 36 or 48 views, forming mini-batches over multiple sequences and views to fully utilize GPU memory. We adopt mixed-precision training and gradient clipping to stabilize optimization. The overall training objective is a weighted sum of per-pixel reconstruction and regularization losses: we combine an ℓ_1 loss with SSIM and LPIPS losses for image reconstruction, and add motion, volume, and opacity regularization terms to encourage temporally smooth trajectories and compact, non-degenerate Gaussian sets, using $\lambda_{\text{motion}} = 0.1$, $\lambda_{\text{vol}} = 10{,}000.0$, and $\lambda_{\alpha} = 0.1$ in all experiments. The 4DGS decoder is first pre-trained on ground-truth multi-view supervision, after which we train the rectified flow model for sparse temporal structure prediction (SS flow), and finally the rectified flow model for latent prediction (slat flow). All large-scale experiments are conducted on NVIDIA

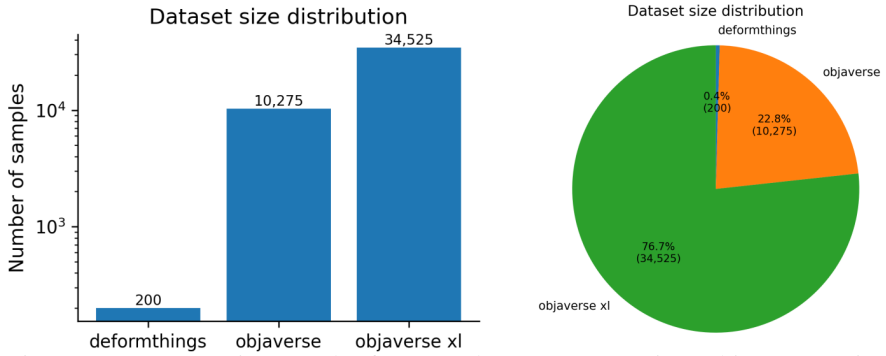


Figure 8: Representative samples from our dataset across various object categories.

A800 GPUs; training the full TESLA model requires several days on 32 GPUs, while the decoder pre-training stage is significantly cheaper and can be completed on a single node.

A.4 Model Architecture

Our model consists of three main components: the 4DGS decoder, the temporal SS flow model, and the temporal slat flow model. The 4DGS decoder employs a transformer-based architecture with temporal attention mechanisms to directly predict Gaussian parameters from structured latents. Both flow models utilize Diffusion Transformer architectures with parallel attention mechanisms for multi-modal conditioning. Tab. 4 provides detailed architectural specifications for each component.

A.5 Inference Pipeline

During inference, TESLA generates native 4D Gaussian Splatting representations through a two-stage process. Given a reference image I and text prompt P, we encode them into conditioning features and initialize with random noise. Our rectified flow model first generates sparse temporal voxel structures by predicting positional indices $p_{i,t} \in \{0, 1, ..., (N-1)\}^3$ for active voxels at each timestep t, establishing the geometric scaffold. Conditioned on this structure, the model then synthesizes latent features $z_{i,t} \in \mathbb{R}^C$ for each active voxel position, completing the temporally structured latent representation $S = \{S_t\}_{t=1}^T$ where $S_t = \{(z_{i,t}, p_{i,t})\}_{i=1}^{L_t}$. The latents are directly transformed into native 4D Gaussian primitives through the 4DGS decoder, enabling flexible rendering at arbitrary viewpoints and timesteps.

B Evaluation Metrics

We evaluate TESLA on both text and image-conditioned 4D generation tasks and compare it against a set of strong baselines under a unified evaluation protocol. Unless otherwise specified, all methods generate sequences of 24 frames with shared camera trajectories.

B.1 Quantitative metrics

We use a combination of distributional, semantic, and low-level metrics. For distributional similarity, we compute the Fréchet Video Distance (FVD) between generated videos $\mathcal{G}$ and real videos $\mathcal{R}$. Let (μ_g, Σ_g) and (μ_r, Σ_r) be the mean and covariance of feature embeddings extracted from a pretrained video recognition network for $\mathcal{G}$ and $\mathcal{R}$, respectively. FVD is given by

$$\mathrm{FVD}(\mathcal{G}, \mathcal{R}) = \|\mu_g - \mu_r\|_2^2 + \mathrm{Tr}\Big(\Sigma_g + \Sigma_r - 2(\Sigma_g^{1/2}\Sigma_r\Sigma_g^{1/2})^{1/2}\Big),$$

where lower values indicate better alignment with the real data distribution. To measure semantic alignment for text-conditioned generation, we compute CLIP-based similarities. Let $f_{\text{text}}(x_i)$ be the CLIP text embedding and $f_{\text{img}}(\hat{I}^M_{i,t,v})$ the image embedding of a generated frame. We calculate

$$s^M_{\text{text}}(i) = \frac{1}{TV}\sum_{t=1}^{T}\sum_{v=1}^{V}\frac{\langle f_{\text{text}}(x_i),\, f_{\text{img}}(\hat{I}^M_{i,t,v})\rangle}{\|f_{\text{text}}(x_i)\|_2\,\|f_{\text{img}}(\hat{I}^M_{i,t,v})\|_2},$$

Table 4: More details about Model Architectures.

Network	#Layer	#Dim.	#Head	Block Arch.	Special Modules	#Param.
$\mathcal{D}_{4DGS}$	12	768	12	SW-MSA + FFN	Swin Attn.	85.52M
SS Flow	24	1024	16	Parallel Attn + Temporal Attn	QK Norm. + Sp. Conv.	643.84M
Slat Flow	24	1024	16	Parallel Attn + Temporal Attn	QK Norm. + Sp. Conv.	688.61M

Table 5: Ablation studies on key architectural components.

Variant	I2V ↑	M.Sm. ↑	Sub.Cons. ↑
w/o Temporal Attn	0.812	0.991	0.914
Sequential Attn	0.851	0.992	0.927
Spatial only (xyz)	0.811	0.986	0.907
Full	**0.861**	**0.994**	**0.941**

and report the average $\bar{s}^M_{\text{text}} = \frac{1}{N}\sum_{i=1}^{N} s^M_{\text{text}}(i)$. For image-conditioned generation, we instead compare frames to the reference image x_i^{ref} using the same formulation.

B.2 VBench evaluation

To obtain fine-grained assessments of motion quality and temporal behavior, we additionally employ VBench [54]. For each generated sequence, we render the motion as a 24-frame video and feed the full sequence into the VBench pipeline using the default configuration. VBench produces multiple dimension-wise scores such as subject fidelity, motion smoothness, temporal consistency, and aesthetic quality. We evaluate all methods under identical settings and report the averaged score over all prompts and sequences for each dimension.

B.3 Baseline implementations

For animation generation, we follow the evaluation protocol of Animate3D [14] and AnimateAnyMesh [45]. Since AnimateAnyMesh operates on meshes, we first reconstruct 3D meshes from reference images using Trellis [46] and apply mesh simplification [19] to meet vertex-count constraints. Similarly, Animate3D initializes with Trellis-generated 3DGS and employs a video diffusion model fine-tuned on 3D data to synthesize multi-view dynamic sequences for 4D reconstruction.

For 4DGS generation, we follow Consistent4D [15]'s protocol and compare against four state-of-the-art methods: 4DGen [50], Consistent4D [15], L4GM [37], and GVF [53]. TESLA and 4DGen directly accept image-text inputs for evaluation. Since Consistent4D, L4GM, and GVF require video inputs, we first generate multi-view video sequences from the reference images and text prompts using Wan2.1 [38], then feed these videos into their official implementations. All methods are evaluated on a single NVIDIA 4090 GPU using CLIP score, FVD, and generation time to ensure consistent comparison of quality and efficiency.

For these video inputs, we use Wan2.1-I2V-14B-720P [38]. We choose the image-to-video variant rather than the text-to-video variant because it conditions on the same reference image that every method receives, which keeps the inputs aligned across the comparison. We keep the official default inference settings, namely a classifier-free guidance scale of 5.0, a flow-matching shift of 5.0, a maximum resolution of 1280×720 with the aspect ratio determined by the reference image, the default negative prompt of the I2V configuration, and disabled prompt extension. Each generated video contains 81 frames at 16 fps, from which we uniformly sample the 24 frames used in our evaluation. The same generated videos are given to all three video-conditioned baselines, whereas TESLA receives only the reference image and the text prompt and never accesses these videos.

C More Experiments

C.1 Additional Ablation Studies

We conduct additional ablation studies to validate the effectiveness of our key architectural designs. Table 5 presents a comprehensive analysis of three critical components: temporal attention mechanism, parallel attention architecture, and position encoding strategy.

Table 7: Ablation studies on sequence length.

	I2V ↑	M.Sm. ↑	Sub.Cons. ↑	Aest.Q ↑
16 frames	0.842	**0.994**	0.922	**0.534**
24 frames	**0.861**	**0.994**	**0.941**	0.533

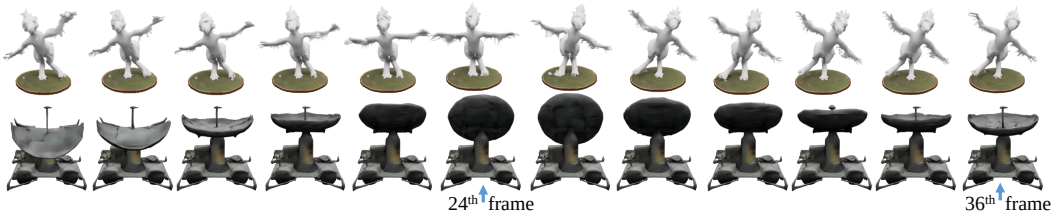


Figure 9: TESLA generalizes to extended-length sequences via sliding window inference.

The results demonstrate that temporal attention plays a crucial role in capturing motion dynamics, as removing it leads to significant performance degradation. For multi-modal conditioning, our parallel architecture consistently outperforms sequential fusion alternatives, validating its effectiveness in integrating image and text guidance. Finally, we find that using only spatial or temporal encoding alone yields degraded performance, while our hybrid spatio-temporal encoding explicitly disentangles these two dimensions, enabling the network to better distinguish and model geometric structure and motion dynamics separately for coherent 4D generation.

C.2 Geometric Accuracy

The latent grid resolution in TESLA is carefully designed to strike a balance between representation efficiency and geometric fidelity. The chosen resolution demonstrates sufficient capacity to handle geometrically challenging structures while maintaining practical efficiency. To further substantiate this with quantitative evidence, we evaluate geometric accuracy using the Uni3D metric in Tab. 6, where TESLA achieves competitive performance against prior methods and confirms that the chosen grid resolution does not compromise structural precision.

Table 6: Geometric Accuracy.

	Uni3D ↑
4DGen [50]	24.25
C4D [15]	10.55
L4GM [37]	33.81
GVF [53]	35.66
Ours	**35.89**

C.3 Variable Length

Although TESLA is trained on fixed-length sequences, the temporally structured latent representation is inherently capable of accommodating different sequence lengths without architectural modifications. To validate this flexibility, we conduct an ablation study comparing models trained on sequences of different lengths. As reported in Tab. 7, models trained on shorter sequences achieve comparable performance across all evaluation metrics, confirming that TESLA generalizes well to variable temporal granularities and does not rely on a fixed sequence length to maintain generation quality.

To extend generation beyond the training-time horizon, we adopt a sliding window strategy at inference, drawing from inpainting techniques widely used in diffusion models. Starting from an initial generation, the final half of the generated frames serve as inpainting conditions for the subsequent window, establishing a temporal overlap that guarantees continuity across window boundaries. Through this iterative process, sequences of arbitrary length can be produced while preserving consistent geometry and appearance. As demonstrated in Fig. 9, sequences generated through this approach exhibit smooth and coherent animations throughout window transitions, further validating TESLA's capability to produce long-horizon dynamic content that remains temporally stable beyond the training-time sequence length.

C.4 Additional Qualitative Comparisons

We provide additional qualitative comparisons against state-of-the-art 4DGS generation baselines in Fig. 10, covering diverse object categories and motion patterns drawn from the Consistent4D [15] evaluation protocol. Across these cases, optimization-based approaches such as 4DGen [50] and Consistent4D [15] tend to produce over-smoothed surfaces and noticeable temporal flickering, while feed-forward baselines L4GM [37] and GVF [53] inherit residual artifacts from the intermediate multi-view video synthesis stage and exhibit blurred appearance under large motion. In contrast, by directly generating 4D Gaussian primitives from temporally structured latents, TESLA bypasses these intermediate failure modes and yields more stable geometry, sharper appearance details, and smoother motion across the full sequence. These observations are consistent with the quantitative results reported in Tab. 2, further supporting the effectiveness of native 4DGS generation.

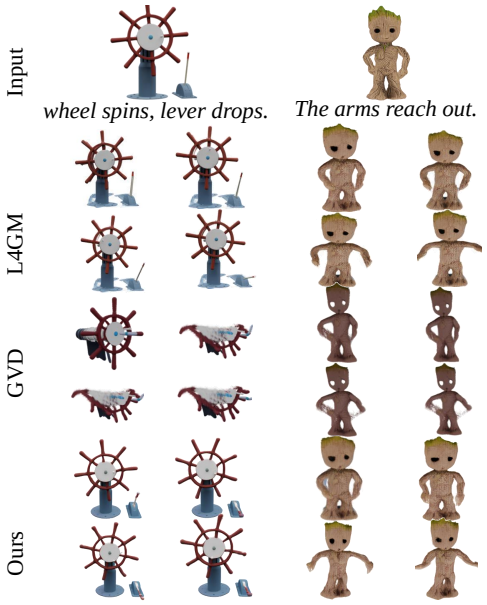


Figure 10: Additional qualitative comparisons on 4DGS generation.

C.5 Motion Quality and Diversity

Beyond the view-aligned VBench scores reported in the main paper, we further evaluate three motion-centric properties that frame-level perceptual metrics do not directly capture. Dynamic Score is computed as the mean optical flow magnitude between consecutive rendered frames, where lower values indicate smoother temporal evolution with fewer flickering or jittering artifacts. Motion Diversity follows the Average Pairwise Distance protocol of [6]: for each reference object we render multiple animations conditioned on different text prompts and report the mean pairwise CLIP-feature distance, where larger values reflect richer prompt-conditioned motion variability. Temporal Consistency is computed using the VBench evaluation pipeline [54] based on pretrained vision-feature similarity between consecutive rendered frames. As summarized in Tab. 8, TESLA attains the lowest Dynamic Score and the highest Motion Diversity and Temporal Consistency, confirming that the temporally structured latent simultaneously suppresses unstable motion artifacts and admits diverse, semantically aligned animations. Qualitative evidence of this controllability is shown in Fig. 7, where the same reference object is animated under multiple text prompts to elicit distinct motion patterns.

Table 8: Quantitative motion evaluation against animation baselines.

Method	Dynamic Score ↓	Motion Diversity ↑	Temporal Consistency ↑
Animate3D [14]	4.37	0.48	0.980
AnimateAnyMesh [45]	3.72	0.73	0.989
Ours	**2.35**	**0.74**	**0.991**

C.6 Comparison with Per-Frame TRELLIS

TESLA builds on the two-stage structure of TRELLIS [46], which raises the question of whether native 4D generation is needed at all. To answer it, we evaluate TRELLIS independently on each frame using the same inputs and stack the resulting per-frame outputs. As shown in Tab. 9, per-frame TRELLIS performs worse on every temporal metric. This gap is structural. Independent per-frame generation establishes no cross-frame correspondence, and no Gaussian primitive persists over time, which leads to flickering across timesteps. A stack of independently generated 3DGS can also be rendered only at the discrete timesteps for which it was decoded, whereas TESLA renders at arbitrary continuous timesteps. TESLA avoids both problems because it jointly models the complete 4D structure of the sequence through temporal attention and the $xyzt$ position encoding, rather than generating sparse structures frame by frame.

Table 9: Comparison with per-frame TRELLIS. Continuous t indicates whether the output can be rendered at arbitrary continuous timesteps.

Method	Dynamic Score ↓	Temp.Cons. ↑	Sub.Cons. ↑	Continuous t
Per-frame TRELLIS [46]	4.210	0.979	0.938	✗
Ours	**2.350**	**0.991**	**0.941**	✓

C.7 More Results

To further demonstrate the versatility and quality of TESLA, we provide additional generation results across diverse object categories and motion patterns in the following Fig 12 and Fig 14.

D Concurrent Work

Helix4D [48], which appeared after our submission, also extends the TRELLIS family to 4D. It adapts TRELLIS.2 from image-to-3D generation to video-conditioned 4D mesh generation, sharing information across frames through sliding-window cross-frame attention anchored on the first frame and injecting time through a 4D positional encoding that repurposes low-frequency spatial RoPE bands. The two methods address different settings. Helix4D is conditioned on an input video that specifies the motion, whereas TESLA generates the motion from a text prompt given only a reference image, and it outputs native 4D Gaussians rather than dynamic meshes.

E Limitations and Broader Impact

TESLA is currently evaluated on object-centric dynamic assets and finite-length animations, so performance on long multi-object interactions, complex scene-level dynamics, and out-of-distribution motion categories remains less explored.

Within this scope, we observe three main limitations. First, quality degrades as motion becomes larger or deformation more complex, since each 4D Gaussian has a limited temporal extent and cannot represent motion that spans much of the sequence. Second, detail is bounded by the sparse voxel lattice, so thin structures near the voxel resolution, such as animal limbs, may intermittently disappear, and fine textures such as faces can appear blurry, especially because high-fidelity faces are rare in our training data. Third, each 4D Gaussian evolves independently without a rigidity or physics prior. This flexibility enables topological changes, but rigid objects may not move rigidly and the motion may not be physically accurate, partly because most training sequences are artist-created. Higher lattice resolution, pretrained 2D diffusion priors, and physics-aware constraints are promising directions for future work.

As a generative 4D content system, TESLA may lower the cost of creating dynamic assets for AR/VR, games, and digital media, while also requiring responsible release practices to reduce misuse in deceptive or unauthorized synthetic content generation.



"This person is looking down."



"The dinosaur ran forward with its steps."



"The flowers slowly bloom."



"The vampire flower made a bite downward."



'The man preparing to punch took a step forward."



"The propeller is rotating."

Figure 11: More visualizations of generated 4DGS results from front views (Part 1).



"The blade extends into the handle."



"The propeller is rotating."



"The tree leaned slightly."



"The fan starts to spin."



"The robot moves forward."



"The head is slightly tilted."

Figure 12: More visualizations of generated 4DGS results from front views (Part 2).

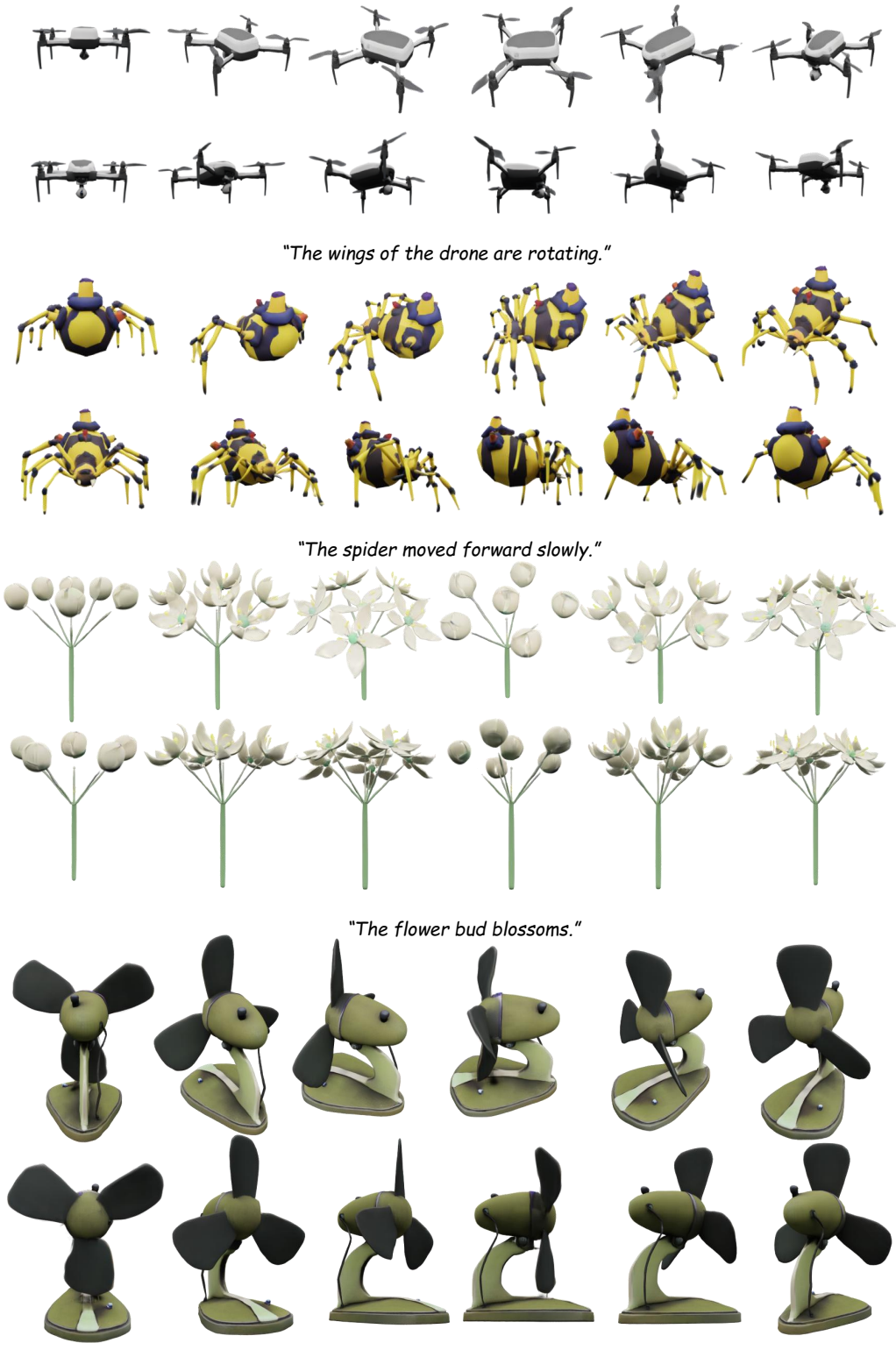
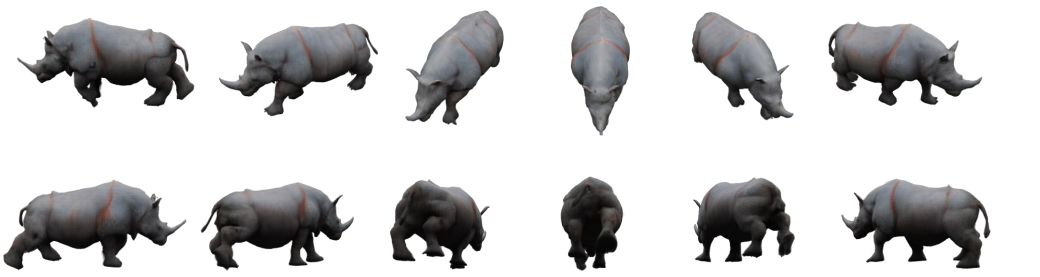
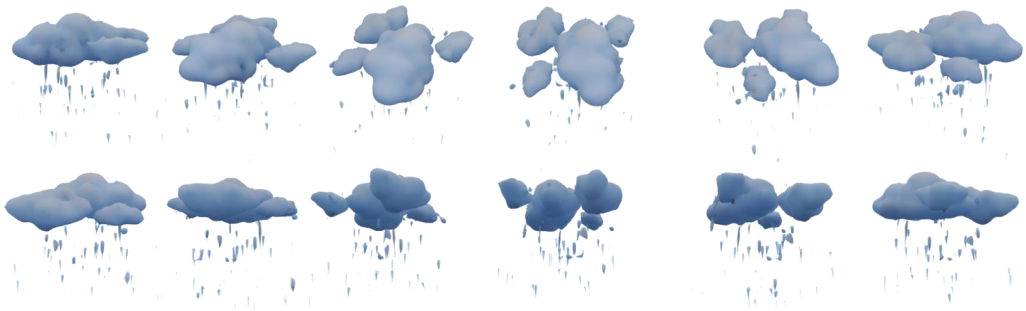


Figure 13: More visualizations of generated 4DGS results from free viewpoints (Part 1).



"The rhinoceros walked forward."



"The clouds keep raining."



"The spider is crawling."



"A flock of birds flies forward."

Figure 14: More visualizations of generated 4DGS results from free viewpoints (Part 2).

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state TESLA's feedforward native 4DGS generation framework, temporally structured latents, native decoder, curated dataset, and empirical improvements, which are described in Secs. 3–4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Sec. 5 discusses limitations on object-centric evaluation, finite-length generation, and generalization to complex scene-level dynamics.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: This paper does not include theoretical results. All formulas describe model architecture and loss functions rather than theorems requiring proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Sec. 4 and the supplementary material provide the datasets, training stages, hardware, loss weights, grid resolution, sequence length, metrics, and model architecture specifications used for the main experiments.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Demonstration code is provided as supplementary material. The complete source code, data, and pretrained checkpoints will be publicly released upon acceptance.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Sec. 4 and the supplementary material specify the optimizer, learning rate, weight decay, warmup and cosine schedule, loss weights, training stages, datasets, sequence length, view sampling, baselines, and metrics.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: We report the average performance as the experimental results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Sec. 4 reports the use of NVIDIA A800 GPUs for training, including the number of GPUs and duration for the decoder, temporal SS flow, and slat flow stages; it also reports inference timing on a single NVIDIA 4090 GPU.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics and confirm that this research conforms with it in every respect.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

Justification: Secs. 1 and 5 discuss potential applications in AR/VR, games, and digital media, as well as the need for responsible release to reduce misuse in deceptive or unauthorized synthetic content generation.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The proposed method focuses on 4D generation for research purposes and does not pose significant risks beyond those of existing video generation models.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The paper cites the datasets, pretrained models, and baselines used in the work, including Deformthings4D, Objaverse, Objaverse-XL, Qwen2.5-VL, Wan2.1, TRELLIS.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.

- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The paper introduces a curated dataset of 45,000 animated sequences, and Sec. 1 plus the supplementary material document its sources, filtering, rendering protocol, captioning process, and scale.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [No]

    Justification: Sec. 4 reports a user study with 20 participants and describes the rating dimensions, but the current draft does not include full participant instructions, screenshots, or compensation details.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [No]

Justification: The current draft reports a user study but does not yet state whether IRB approval or an equivalent review was obtained, nor does it describe participant risk disclosures.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: Sec. 4 and the supplementary material describe the use of Qwen2.5-VL to generate detailed captions for animated training sequences, which affects the text-motion conditioning data used by TESLA.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.