DreamMesh4D: Video-to-4D Generation with Sparse-Controlled Gaussian-Mesh Hybrid Representation

Zhiqi Li[*1,2] **Yiming Chen**[*1,2] **Peidong Liu**[†2]

[1] Zhejiang University [2] Westlake University
{lizhiqi49, chenyiming, liupeidong}@westlake.edu.cn

Abstract

Recent advancements in 2D/3D generative techniques have facilitated the generation of dynamic 3D objects from monocular videos. Previous methods mainly rely on the implicit neural radiance fields (NeRF) or explicit Gaussian Splatting as the underlying representation, and struggle to achieve satisfactory spatial-temporal consistency and surface appearance. Drawing inspiration from modern 3D animation pipelines, we introduce DreamMesh4D, a novel framework combining mesh representation with geometric skinning technique to generate high-quality 4D object from a monocular video. Instead of utilizing classical texture map for appearance, we bind Gaussian splats to triangle face of mesh for differentiable optimization of both the texture and mesh vertices. In particular, DreamMesh4D begins with a coarse mesh obtained through an image-to-3D generation procedure. Sparse points are then uniformly sampled across the mesh surface, and are used to build a deformation graph to drive the motion of the 3D object for the sake of computational efficiency and providing additional constraint. For each step, transformations of sparse control points are predicted using a deformation network, and the mesh vertices as well as the surface Gaussians are deformed via a novel geometric skinning algorithm. The skinning algorithm is a hybrid approach combining LBS (linear blending skinning) and DQS (dual-quaternion skinning), mitigating drawbacks associated with both approaches. The static surface Gaussians and mesh vertices as well as the dynamic deformation network are learned via reference view photometric loss, score distillation loss as well as other regularization losses in a two-stage manner. Extensive experiments demonstrate superior performance of our method in terms of both rendering quality and spatial-temporal consistency. Furthermore, our method is compatible with modern graphic pipelines, showcasing its potential in the 3D gaming and film industry. The source code is available at our website: https://lizhiqi49.github.io/DreamMesh4D.

1 Introduction

The emergence and development of Generative Artificial Intelligence (GenAI) have significantly revolutionized 3D generation techniques in recent years [20]. The technology has effectively allowed the creation of static objects, including their shape, texture, and even an entire scene from a simple text prompt or a single image. Recently, the wave of advancement has been propelled to the filed of dynamic (4D) content generation [45], which offers immense potential in fields including, but not limited to, AR/VR, filming, gaming and animation. However, it's still quite challenging to efficiently

[*]Equal Contribution; [†] Corresponding author.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

Figure 1: Given monocular videos, our method is able to generate high-fidelity dynamic meshes. We also produce a composited scene demo (top bar and left side of the figure) with the generated dynamic meshes, showcasing our method's compatibility with modern 3D engines.

generate high-quality 4D content due to its increased spatial-temporal complexity and higher demand on algorithm design.

The promising strides in 3D generation are largely attributed to the pre-trained large 2D diffusion models [39, 41, 31]. In particular, score distillation sampling (SDS) [33] enables the 3D generation [30, 55, 18, 42, 9] from scratch by distilling 3D knowledge from a pre-trained 2D diffusion model [39]. Following works on text-to-3D [24, 57, 44, 15] and image-to-3D [26, 35, 50, 48] have further improved the performance of 3D generation tasks both in quality and stability. Inspired by the successes of SDS-based 3D generation, recent works [69, 1, 38, 12, 65, 66] explore generating 4D assets by distilling prior knowledge from pre-trained video diffusion models [54, 3] or novel-view synthesis models [26, 43]. Both text-to-4D [45, 1, 25] and image-to-4D methods [69] mainly rely on pre-trained video diffusion models, which are not yet capable of generating high-quality video, thus usually struggle to generate high-quality 4D content. On the contrary, video-to-4D methods [38, 12, 66, 65] directly generate 4D assets from off-the-shelf monocular videos, making the results more appealing and with better spatial-temporal consistency. Existing video-to-4D methods either rely on the implicit dynamic NeRFs [12] or explicit dynamic Gaussian splatting [38, 65, 66] as the underlying representation. Nevertheless, both of them do not have tight constrains for surface, leading to redundant optimization space and impeding the learning of deformation.

Inspired by modern graphic pipelines for 3D animation, we propose **DreamMesh4D**, which exploits 3D triangular mesh representation and sparse-controlled geometric skinning methods [47, 16] for video-to-4D generation. To better supervise the generation with 2D signals, instead of using classic mesh with UV texture maps, we choose a hybrid representation, SuGaR [9], which marries 3D Gaussians to mesh surface for more elaborate appearance modeling. Flat Gaussians are bound to mesh faces based on barycentric coordinates hence the rendering process of 2D images is differentiable with respect to both the position of mesh vertices and the attributes of Gaussians. For high-quality object modeling and efficient motion driving of the object, our method is designed in a static-to-dynamic optimization manner. In particular, during the static stage, an initial coarse mesh is generated utilizing existing image-to-3D generation methods [26, 68, 61]. Then we refine its both geometry and texture by jointly optimizing the mesh vertices as well as the attributes of bound surface Gaussians under the hybrid representation via both the reference image photometric loss and the SDS loss. For dynamic learning, we uniformly sample sparse control nodes from the surface of the refined mesh, to build a deformation graph. Then at each timestamp, transformation associated with each control node is predicted by a deformation network. The deformation of all mesh vertices and surface Gaussians are obtained from those predicted transformations via a novel geometric skinning algorithm, which benefits from both the LBS (linear blending skinning) and DQS (dual-quaternion skinning) methods. The deformation network is optimized under the supervision of photometric loss from reference video frames, novel-view SDS loss and geometric regularization terms.

Extensive experiments are conducted and demonstrate that our method can generate high-fidelity dynamic textured mesh from monocular video, and significantly outperforms previous works both quantitatively and qualitatively, establishing new benchmark in the field of video-to-4D generation. As shown in Fig. 1, our generated assets can be directly simulated in modern 3D engines, showcasing its potential in the 3D gaming and film industries.

2 Related Work

3D Generation Since the introduction of score distillation sampling (SDS) by DreamFusion [33], subsequent works [24, 5, 57, 35, 48, 49, 23] have significantly improved the performance of optimization-based 3D generation algorithms. Many works adopt a multi-stage optimization strategy [24, 5, 57, 48] to enhance generated appearance. Another line of research [26, 44, 27, 19] focuses on training multi-view diffusion models to inject multi-view supervision into SDS loss for addressing the Janus problem. DreamGaussian [49] and GaussianDreamer [64] pioneer the usage of 3D Gaussian[18] as the underlying representation and achieve 3D content generation in minutes. Although these methods demonstrate the potential of 3D Gaussian in 3D content generation, obtaining an object with high-quality geometry is still quite challenging. In this work, we explore to employ a Gaussian-mesh hybrid representation [9] in our 4D generation tasks for better modeling of both object surface geometry and texture.

4D Representations Dynamic 3D representations form the foundation of 4D reconstruction and generation tasks. Most current methods extend static NeRF [30] to dynamic scenarios. These approaches, such as deformable [32, 34, 51, 59] and time-varying [4, 7, 8, 6] NeRF-based methods, are prevalent. There are also some works trying to model dynamic with 4D neural implicit surface [56, 14]. However, these implicit representations suffer from long optimization time and low reconstruction quality due to its computationally expensive volume rendering and implicit representation. Recent interest in 4D tasks has focused on 3D Gaussian representations due to their fast rendering speed and explicit nature. Some works [63, 58, 22] train networks to predict Gaussian kernel deformations, while others model kernel motion via polynomial representation or per-frame optimization [28, 21]. Besides, both SC-GS [11] and HiFi4G [13] employ sparse control points for Gaussian kernel deformation, with SC-GS using LBS and HiFi4G using DQS to drive Gaussians motion, thus ensuring spatial-temporal consistency. In this work, we propose to deform the object through a novel geometric skinning method, handling the artifacts associated with both LBS and DQS.

4D Generation Although great progress has been achieved in 3D generation tasks, 4D generation is still challenging due to its requirement on additional temporal supervision. Since current pre-trained video diffusion models [3, 54] still struggle to generate high-quality video contents, it is challenging to distill useful motion knowledge via SDS optimization. Hence, the performance of existing text-to-4D [45, 1, 25] and image-to-4D methods [69] usually struggle with low appearance quality. Another line of works focus on video-to-4D generation [12, 38, 65]. These methods leverage current multiview diffusion models [26, 43, 27] to calculate the SDS loss [38, 12, 60] or generate per-frame multi-view images [66, 62] as supervision signal. Among them, a concurrent work of our method, SC4D [60], optimizes a set of sparse-controlled dynamic 3D Gaussians by per-frame SDS loss from Zero123 [26] with a coarse-to-fine strategy. However, the issue of unsatisfying surface quality caused by 3D Gaussian-based representation still exists. In contrast, our method is grounding on a Gaussian-mesh hybrid representation [9], enhancing the reconstruction quality both in texture and geometry.

3 Method

In this section, we first introduce the relevant preliminaries in section 3.1. Then we illustrate the details of DreamMesh4D which is divided into static stage and dynamic stage in section 3.2 and 3.3 respectively. The overview of our method is presented in Fig. 2.

3.1 Preliminaries

Geometric Skinning Algorithms Given a mesh with N_v vertices, $\mathcal{M} = \{\mathcal{V}, \mathcal{F}\}$ where $\mathcal{V} = \{\mathbf{v}_i | \mathbf{v}_i \in \mathbb{R}^3\}, i \in \{1, 2, ..., N_v\}$ is the set of vertices and $\mathcal{F}$ represents the triangle faces. In geometric skinning algorithms, there are also some sparse control nodes (bones/skeletons) $\mathcal{P} =$

$\{\mathbf{p}_i|\mathbf{p}_i \in \mathbb{R}^3\}, i \in \{1, 2, ..., N_p\}$, where N_p is the number of control nodes. For a mesh vertex $\mathbf{v} \in \mathcal{V}$, its deformation is calculated by blending a number of neighboring control nodes $\mathcal{N}(\mathbf{v})$ through skinning algorithms. The local deformation for a control node $\mathbf{p} \in \mathcal{N}(\mathbf{v})$ can be decomposed into a deformation matrix $F_\mathbf{p} \in \mathbb{R}^{3\times3}$ and a translation vector $t_\mathbf{p} \in \mathbb{R}^3$, and the deformation matrix can be further decomposed into a rotation matrix $R_\mathbf{p} \in \mathbb{R}^{3\times3}$ and a shear matrix $S_\mathbf{p} \in \mathbb{R}^{3\times3}$ using polar decomposition. The strength of the influence of node $\mathbf{p}$ to vertex $\mathbf{v}$ can be represented as $w_\mathbf{p}$ and $\sum_{\mathbf{p}\in\mathcal{N}(\mathbf{v})} w_\mathbf{p} = 1$. Linear blending skinning (LBS) [47] computes the deformation of vertex $\mathbf{v}$ by linearly blending the influence of nodes in $\mathcal{N}(\mathbf{v})$:

$$\tilde{\mathbf{v}}^{lbs} = \sum_{\mathbf{p}\in\mathcal{N}(\mathbf{v})} w_\mathbf{p}(F_\mathbf{p}\mathbf{v} + t_\mathbf{p}). \tag{1}$$

LBS is widely used due to its simple formulation and natural animations. However, it suffers from the well-known "volume loss" or "candy wrapper" artifacts under complex deformation. As an enhancement, dual-quaternion skinning (DQS) [16] represents the transformation of node $\mathbf{p}$ with a unit dual-quaternion $\zeta_\mathbf{p} = DQ(R_\mathbf{p}, t_\mathbf{p})$. Then the deformation of $\mathbf{v}$ can be calculated with DQS by:

$$\tilde{\mathbf{v}}^{dqs} = \bar{\zeta}\mathbf{v}\bar{\zeta}^*, \text{where } \bar{\zeta} = \frac{\sum_{\mathbf{p}\in\mathcal{N}(\mathbf{v})} w_\mathbf{p}\zeta_\mathbf{p}}{||\sum_{\mathbf{p}\in\mathcal{N}(\mathbf{v})} w_\mathbf{p}\zeta_\mathbf{p}||}, \tag{2}$$

where $\bar{\zeta}^*$ represents the conjugate of $\bar{\zeta}$. DQS can eliminate the artifacts associated with LBS, but cannot handle non-rigid deformation since the strain effect is not considered. It also suffers from an undesirable joint-bulging artifact while blending, which requires artistic manual work to fix [40].

3D Gaussians and SuGaR Gaussian Splatting [18] represents the scene as a collection of 3D Gaussians, where each Gaussian g is characterized by its center $\mu_g \in \mathbb{R}^3$ and covariance $\Sigma_g \in \mathbb{R}^{3\times3}$. The covariance Σ_g is parameterized by a scaling factor $s_g \in \mathbb{R}^3$ and a rotation matrix represented via a unit quaternion $q_g \in \mathbb{R}^4$. Additionally, each Gaussian maintains opacity $\alpha_g \in \mathbb{R}$ and color features $c_g \in \mathbb{R}^C$ for rendering via splatting. Typically, color features are represented using spherical harmonics to model view-dependent effects. During rendering, the 3D Gaussians are projected onto the 2D image plane as 2D Gaussians, and color values are computed through alpha composition of these 2D Gaussians in front-to-back depth order. While the vanilla Gaussian Splatting representation may not perform well in geometry modeling, SuGaR [9] introduces several regularization terms to enforce flatness and alignment of the 3D Gaussians with the object surface. This facilitates extraction of a mesh from the Gaussians through Poisson reconstruction [17]. Furthermore, SuGaR offers a hybrid representation by binding Gaussians to mesh faces. A SuGaR mesh can be represented as $\mathcal{S} = \{\mathcal{V}, \mathcal{F}, \mathcal{G}\}$ where $\mathcal{G}$ denotes all surface Gaussians. For a triangle face $f \in \mathcal{F}$, the Gaussians $\mathcal{G}(f)$ bound on f are defined with barycentric coordinates. This hybrid representation allows joint optimization of texture and geometry through backpropagation.

3.2 Static Stage

The purpose of the static stage is to generate a refined Gaussian splats bounded triangular mesh. This procedure starts with a coarse mesh generated from a reference image I^* that is sampled from all video frames $\mathcal{I}$. Although there exists several fast mesh generation methods [68, 61], we refer to Zero123-based SDS optimization for its stability. In particular, we conduct simple SDS training on a set of randomly initialized 3D Gaussians for a fixed number of steps with regular densification and pruning. After that, we stop densification and pruning, and include SuGaR's regularization terms [9] to make Gaussians aligned to object surface. Finally all Gaussians with opacity lower than a threshold $\bar{\sigma} = 0.5$ are pruned, after which Poisson reconstruction [17] is performed to extract a coarse mesh.

On the generated coarse mesh, we attach $x = 6$ new flat Gaussians to every triangle face. For each training step, we render RGB image $\hat{I}^*$ and mask $\hat{M}^*$ under reference view to calculate RGB loss $\mathcal{L}^s_{ref} = ||\hat{I}^* - I^*||_2^2$ and mask loss $\mathcal{L}^s_{mask} = ||\hat{M}^* - M^*||_2^2$. The SDS loss $\mathcal{L}^s_{SDS}$ based on Zero123 [26] is also computed under randomly sampled views. The total loss for static SuGaR optimization is:

$$\mathcal{L}_{static} = \lambda^s_{SDS}\mathcal{L}^s_{SDS} + \lambda^s_{ref}\mathcal{L}^s_{ref} + \lambda^s_{mask}\mathcal{L}^s_{mask}, \tag{3}$$

where λ^s_{SDS}, λ^s_{ref} and λ^s_{mask} are the weights for different loss terms in static stage.

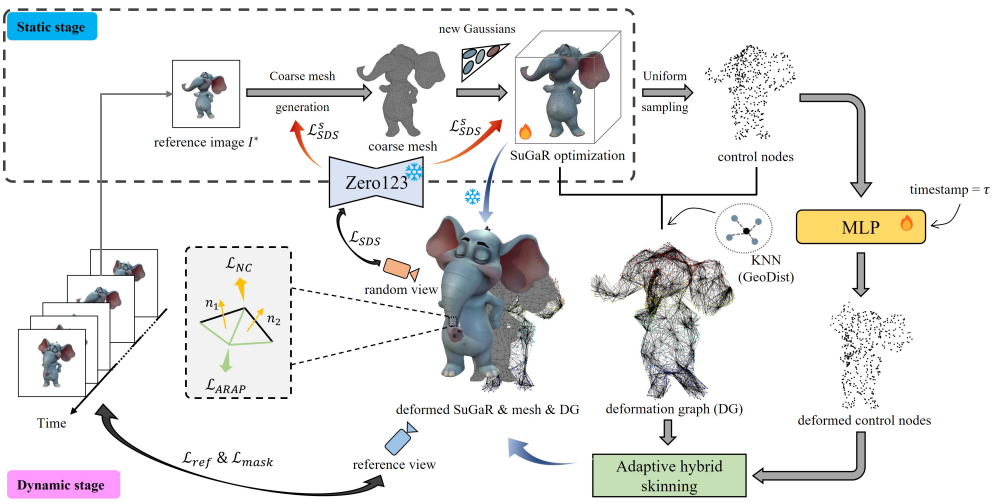

Figure 2: **Overview of DreamMesh4D.** In static stage shown in top left part, a reference image is picked from the input video from with we generate a Gaussian-mesh hybrid representation through a image-to-3D pipeline. As for dynamic stage, we build a deformation graph between mesh vertices and sparse control nodes, and then the mesh and surface Gaussians are deformed by fusing the deformation of control nodes predicted by a MLP through a novel adaptive hybrid skinning algorithm.

3.3 Dynamic Stage

In this section, we are going to delve into the deformation procedure for the Gaussian-mesh hybrid representation. First, we will discuss the construction of the deformation graph on the surface of the refined mesh. Then, we will explain the deformation flow, which progresses from the sparse control nodes to the mesh vertices, and ultimately to the surface Gaussians. We will break down each step to give a clear picture of the entire process.

3.3.1 Deformation Graph Construction

We begin by uniformly sampling N_{node} sparse points on the surface of the refined mesh to serve as control nodes. To establish connections between the mesh vertices and the sparse control nodes, instead of using simple Euclidean distance to locate the nearest nodes (KNN), we pick $N_{neighbor}$ nodes with the shortest geodesic distance (as indicated by GeoDist in Fig. 2) based on the mesh's topology. This ensures that inappropriate connections between disparate mesh regions are avoided. Then, for a vertex $\mathbf{v}$, the influence $w_{\mathbf{p}}$ of each neighboring control node $\mathbf{p} \in \mathcal{N}(\mathbf{v})$ is calculated following [47]:

$$w_{\mathbf{p}} = \frac{\hat{w}_{\mathbf{p}}}{\sum_{\mathbf{p}_k \in \mathcal{N}(\mathbf{v})} \hat{w}_{\mathbf{p}_k}}, \text{where } \hat{w}_{\mathbf{p}_k} = \left(1 - \frac{||\mathbf{v} - \mathbf{p}_k||}{d_{\max}}\right)^2, \tag{4}$$

where $d_{\max}$ is the distance to the $(N_{neighbor} + 1)$-nearest node.

3.3.2 Deformation with Adaptive Hybrid Skinning

Given that the object's texture was refined in the previous static stage, we fix the Gaussians' appearance properties (color and opacity) and focus the dynamic learning phase solely on spatial properties (positions, rotations, scalings). Initially, the local deformations of the control nodes are predicted by a deformation network Ψ. The updated spatial properties post-deformation are subsequently computed by integrating the local deformations of the control nodes through geometric skinning. In particular, given a control node $\mathbf{p} \in \mathcal{P}$ and timestamp τ, the deformation network predict its local deformation at τ following the equation below. Note that we omit the subscript "τ" is omitted here and in rest paragraphs for simplicity.

$$(R_{\mathbf{p}}, S_{\mathbf{p}}, t_{\mathbf{p}}, \eta_{\mathbf{p}}) = \Psi(\mathbf{p}), \tag{5}$$

where $R_{\mathbf{p}}, S_{\mathbf{p}} \in \mathbb{R}^{3\times3}$ and $t_{\mathbf{p}} \in \mathbb{R}^3$ are the rotation, shear matrix and translation respectively. Furthermore, to mitigate artifacts associated with LBS and DQS, we propose an adaptive fusion of their effects to calculate the deformation of mesh vertices. Here, $\eta_{\mathbf{p}} \in \mathbb{R}$ denotes the local rigid strength, indicating the extent to which the region around $\mathbf{p}$ is influenced by DQS at time τ.

Deformation of Control Nodes The predicted shear matrix $S_{\mathbf{p}}$ for node $\mathbf{p}$ is intended only for LBS, whose strength is weaken by the predicted factor $\eta_{\mathbf{p}}$, and the final shear matrix at this location is computed as:

$$\bar{S}_{\mathbf{p}} = (1-\eta_{\mathbf{p}})S_{\mathbf{p}} + \eta_{\mathbf{p}}\mathbf{I}, \tag{6}$$

where $\mathbf{I} \in \mathbb{R}^{3\times3}$ represents an identity matrix indicating no strain effect. Afterwards, the new position of node $\mathbf{p}$ at timestamp τ is:

$$\tilde{\mathbf{p}} = F_{\mathbf{p}}\mathbf{p} + t_{\mathbf{p}} = R_{\mathbf{p}}\bar{S}_{\mathbf{p}}\mathbf{p} + t_{\mathbf{p}}. \tag{7}$$

Deformation of Mesh Vertices For the deformation of a specific vertex $\mathbf{v}$ using hybrid skinning, the new vertex position calculated with LBS, $\tilde{\mathbf{v}}^{lbs}$, and that with DQS, $\tilde{\mathbf{v}}^{dqs}$, can be obtained following Eq.1 and Eq.2 respectively. The local rigid strength at $\mathbf{v}$ can be computed by linearly blending that of neighboring nodes:

$$\eta_{\mathbf{v}} = \sum_{\mathbf{p}\in\mathcal{N}(\mathbf{v})} w_{\mathbf{p}} \cdot \eta_{\mathbf{p}}, \tag{8}$$

then the fused position of $\mathbf{v}$ after deformation is the interpolation between $\tilde{\mathbf{v}}^{lbs}$ and $\tilde{\mathbf{v}}^{dqs}$:

$$\tilde{\mathbf{v}} = (1-\eta_{\mathbf{v}})\tilde{\mathbf{v}}^{lbs} + \eta_{\mathbf{v}}\tilde{\mathbf{v}}^{dqs}. \tag{9}$$

The local rotation and strain at $\mathbf{v}$ are the corresponding linear blending from neighboring control nodes:

$$R_{\mathbf{v}} = \exp\Big(\sum_{\mathbf{p}\in\mathcal{N}(\mathbf{v})} w_{\mathbf{p}} \cdot \log R_{\mathbf{p}}\Big), \tag{10}$$

$$S_{\mathbf{v}} = \sum_{\mathbf{p}\in\mathcal{N}(\mathbf{v})} w_{\mathbf{p}} \cdot \bar{S}_{\mathbf{p}}. \tag{11}$$

Deformation of Surface Gaussians Now we have obtained deformations on the level of mesh vertices, afterwards the deformation on each Gaussian kernel will be calculated. Given a Gaussian kernel $g \in \mathcal{G}(f)$ on a triangle face $f = (\mathbf{v}_a, \mathbf{v}_b, \mathbf{v}_c) \in \mathcal{F}$, its new center at timestamp τ is computed as:

$$\tilde{\mu}_g = \pi_a\tilde{\mathbf{v}}_a + \pi_b\tilde{\mathbf{v}}_b + \pi_c\tilde{\mathbf{v}}_c, \tag{12}$$

where (π_a, π_b, π_c) is the Gaussian's barycentric coordinate relative to the three triangle vertices. The new rotation is calculated by applying the local rotation Δq_g fused from related vertices to its original rotation q_g:

$$\Delta q_g = \exp(\pi_a \cdot \log R_{\mathbf{v}_a} + \pi_b \cdot \log R_{\mathbf{v}_b} + \pi_c \cdot \log R_{\mathbf{v}_c}), \tag{13}$$

$$\tilde{q}_g = \Delta q_g \cdot q_g. \tag{14}$$

We further apply the local shear matrix ΔS_g fused from all three vertices to the original Gaussian scaling s_g to obtain the new scaling:

$$\Delta S_g = \pi_a S_{\mathbf{v_a}} + \pi_b S_{\mathbf{v_b}} + \pi_c S_{\mathbf{v_c}}, \tag{15}$$

$$\tilde{s}_g = \Delta S_g s_g. \tag{16}$$

Training Losses After obtaining the deformed hybrid mesh at timestamp τ, we render its RGB $\hat{I}^*_\tau$ and alpha $\hat{M}^*_\tau$ under reference view. We then compute the reconstruction loss $\mathcal{L}_{ref} = ||\hat{I}^*_\tau - I^*_\tau||_2^2$ and mask loss $\mathcal{L}_{mask} = ||\hat{M}^*_\tau - M^*_\tau||_2^2$, where I^*_τ and M^*_τ are the ground truth image and mask of input video at timestamp τ. For supervision under other views, we calculate SDS loss $\mathcal{L}_{SDS}$ based on Zero123 [26] under randomly sampled views. Furthermore, our mesh-based representation naturally facilitates the introduction of local rigid constraints by leveraging the topology of the mesh. Specifically, the as-rigid-as-possible (ARAP) loss [46] is computed as:

$$\mathcal{L}_{ARAP} = \sum_{\mathbf{v}\in\mathcal{V}} \sum_{\mathbf{v}_n\in\mathcal{C}(\mathbf{v})} \omega_n(\mathbf{v}) \big|\big|(\tilde{\mathbf{v}} - \tilde{\mathbf{v}}_n) - R_{\mathbf{v}}(\mathbf{v} - \mathbf{v}_n)\big|\big|_2^2, \tag{17}$$

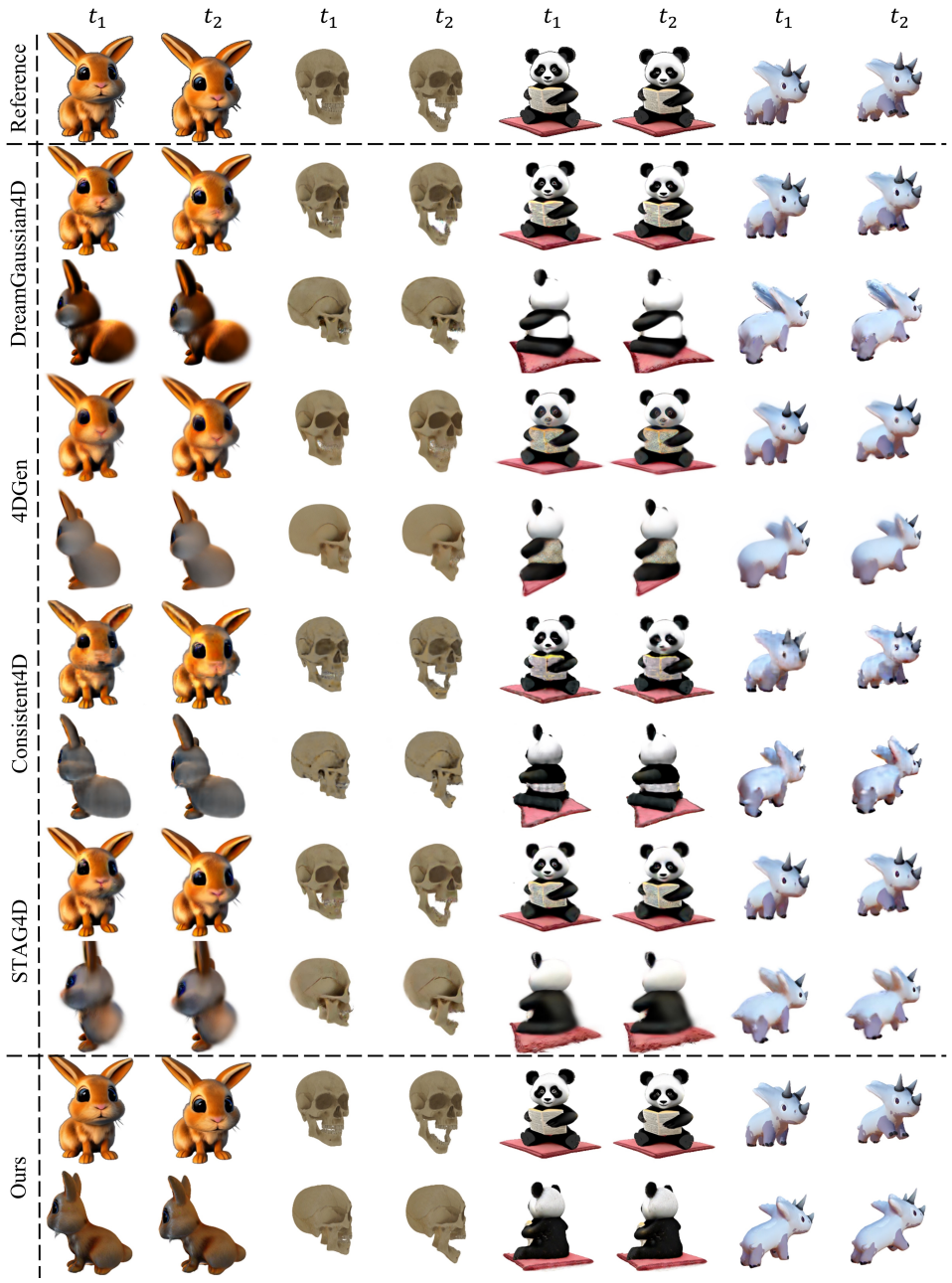

Figure 3: **Qualitative comparison with baselines.** We compare our method with 4 previous video-to-4D methods. The first row provides two ground truth frames for each case. For each compared method, we render each case under reference view and another novel view at the two timestamps. The result demonstrates that our method is able to generate sharper 4D content with rich details, especially for the novel views. Please zoom in for more details.

where $\mathcal{C}(\mathbf{v})$ represents the one-ring neighbors of vertex $\mathbf{v}$, and $\omega_n(\mathbf{v})$ is the cotangent weight [29] between $\mathbf{v}$ and its n-th connected vertex $\mathbf{v}_n$. Furthermore, we also introduce the normal consistency loss $\mathcal{L}_{NC}$ provided by PyTorch3D [37] on the deformed mesh to globally constrain the mesh surface. Hence, the overall objective for our dynamic stage is a weighted combination of the above loss terms:

$$\mathcal{L}_{dynamic} = \lambda_{SDS}\mathcal{L}_{SDS} + \lambda_{ref}\mathcal{L}_{ref} + \lambda_{mask}\mathcal{L}_{mask} + \lambda_{ARAP}\mathcal{L}_{ARAP} + \lambda_{NC}\mathcal{L}_{NC}, \quad (18)$$

where λ_{SDS}, λ_{ref}, λ_{mask}, λ_{ARAP} and λ_{NC} are strengths of different loss terms in dynamic optimization stage.

	PSNR(ref) ↑	SSIM(ref) ↑	LPIPS ↓	FVD ↓	FID-VID ↓	CLIP ↑
Consistent4D	26.58	0.935	0.133	929.39	31.84	0.917
DreamGaussian4D	31.06	0.947	0.143	994.11	32.73	0.913
4DGen (16 frames)	27.02	0.937	0.137	913.10	63.32	0.909
STAG4D	27.99	0.941	0.136	1048.10	38.77	0.905
Ours	**37.04**	**0.980**	**0.126**	**474.96**	**29.14**	**0.938**

Table 1: **Quantitative comparison with baselines.** Our method achieves best score on all metrics.

4 Experiments

4.1 Experiment Setup

Dataset: Our quantitative results are evaluated on the test dataset provided by Consistent4D [12], which contains seven multi-view videos. Each video has one input view for scene generation and four testing views for evaluation. For qualitative evaluation, we curate a set of challenging videos from previous works [12] and those generated by the video diffusion model SVD [3]. **Evaluation metrics**: The per-frame LPIPS [67] score and the CLIP-score [36] are computed between the testing and rendered videos, with the final scores averaged over the four testing views. These two scores serve as image-level metric to assess the structural and semantic similarity between the rendered images and the ground truth. Furthermore, we compute the FID-VID [2] and FVD [52] as video-level metrics to evaluate the video temporal coherence. Note that we report PSNR and SSIM values only for the reference view, as pixel- and patch-wise similarities are too sensitive to reconstruction differences, making them unsuitable for evaluating novel views in our generation task. However, we find they are suitable for evaluating the method's ability of modeling sharp features in the reference view. **Baselines**: We compare our method with previous video-to-4D generation methods: Consistent4D [12], DreamGaussian4D [38], 4DGen [65] and STAG4D [66]. All the experiments of above baselines are conducted using the code from their official GitHub repository.

4.2 Comparison

Qualitative Comparison Fig. 3 shows qualitative results of our method compared to other baseline works. The results reveal that our method generates 4D objects with higher fidelity and more details under reference view. And our method also outperforms other works with better spatial-temporal consistency, demonstrating the effectiveness of our method. Please zoom in for more details.

Quantitative Comparison Table 1 demonstrates superior performance of our method against other baseline works quantitatively. Specifically, our approach notably exceeds the existing state-of-the-art in all measured metrics. Our method excels in both PSNR and SSIM, indicating a high level of reconstruction accuracy. Furthermore, the FVD score is particularly impressive, being only half that of competing methods. We also achieve the lowest FID-VID score, suggesting a significant enhancement in video quality produced by our 4D generation technique. Finally, our method achieves the lowest LPIPS and highest CLIP scores, ensuring both high image realism and semantic consistency. Overall, the numerical data clearly demonstrate the superior capabilities of our method in translating video to 4D content.

4.3 Ablation Study

In this section, we conduct ablation study to analyse the impact of various components on the performance of our method. The components under consideration include: (a) the choice between Euclidean and geodesic distance (EucDist and GeoDist) when constructing deformation graph; (b) our proposed adaptive hybrid skinning algorithm; (c) the ARAP and normal consistency terms (geometric regularization terms); (d) the choice between vanilla 3D Gaussians [18] and Gaussian-mesh representation for our base 3D representation.

EucDist vs. GeoDist Fig. 4(a) provides a comprehensive qualitative analysis of the choice between EucDist and GeoDist. When using EucDist, the vertices on the elephant are incorrectly connected

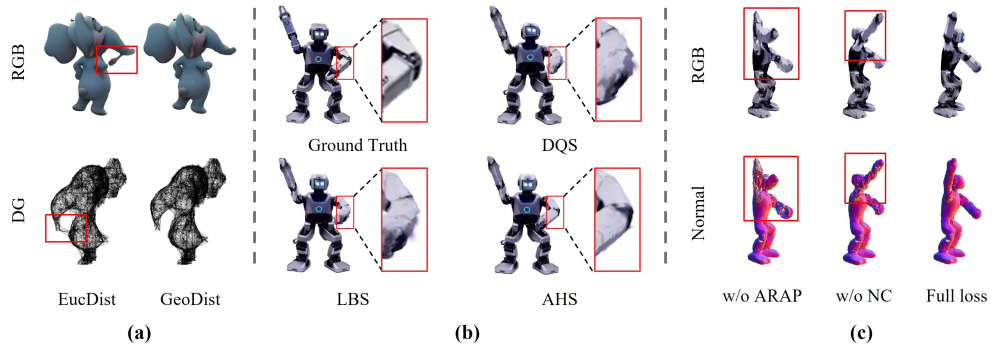

Figure 4: **Qualitative evaluation of ablation studies on:** (a) choice between GeoDist and EucDist for deformation graph (DG) construction; (b) our proposed adaptive hybrid skinning (AHS) against LBS and DQS; (c) effects of ARAP and normal consistency (NC) loss.

	PSNR(ref) ↑	SSIM(ref) ↑	LPIPS ↓	FVD ↓	FID-VID ↓	CLIP ↑
Skinning=DQS	36.28	0.978	0.126	479.83	29.78	**0.940**
Skinning=LBS	36.68	0.980	0.155	540.86	32.19	0.928
w/o arap	37.04	0.979	0.142	751.56	42.08	0.907
w/o normal consistency	36.86	0.980	0.147	519.49	30.90	0.932
Full method	**37.04**	**0.980**	**0.126**	**474.96**	**29.14**	0.938

Table 2: **Quantitative evaluation of ablation study on different components.** During experiments, we keep all other setup unchanged compared to the full method except the tested components. The quantitative scores show that our full method achieves best performance on almost all metrics.

to its body, resulting in significant deformation artifacts. Conversely, GeoDist correctly links the vertices to neighboring nodes, enabling smooth object motion.

Adaptive Hybrid Skinning The upper two rows of Table 2 presents quantitative results when replacing our adaptive hybrid skinning with DQS and LBS respectively. Almost all metrics have a decrease compared to using our adaptive hybrid skinning, showcasing its robustness. We also provide a qualitative comparison on a dancing robot case in Fig. 4(b). When LBS or DQS is used, there are artifacts on the robot's deformed elbow along with uneven surface. In contrast, the artifacts are eliminated and the surface becomes smooth when our adaptive hybrid skinning is used.

Geometric Regularization Terms The third and fourth rows of Table 2 present the scores of metrics when either the ARAP or normal consistency term, respectively, is omitted from Equation 18. Decreases are observed across all metrics compared to full loss terms and it drops significantly when disabling ARAP loss. This circumstance matches the qualitative analysis shown in Fig. 4(c). Without ARAP term, it appears serious distortions on the object geometry. When the normal consistency term is disabled, the object surface becomes less smooth, and consequently, the texture is impaired.

3D Gaussians vs. Gaussian-mesh Hybrid In Fig. 5, we provide the qualitative comparison between 3D Gaussians and the Gaussian-mesh hybrid representation for our base 3D representation. It shows that when utilizing 3D Gaussians, the texture of generated objects is blurry on those parts unseen in reference image. As a contrary, Gaussian-mesh hybrid representation presents clean and high-quality texture under every view, which benefits from the sharp surface provided by mesh.

4.4 Limitations

Despite the superior results achieved, our work still exist several limitations. In particular, our method relies on a pre-trained multi-view diffusion model (Zero123) for novel view supervision through SDS, leading to long optimization time and limited performance on which case Zero123 cannot handle well. Moreover, our method is currently only designed to generate 4D contents at object level from input

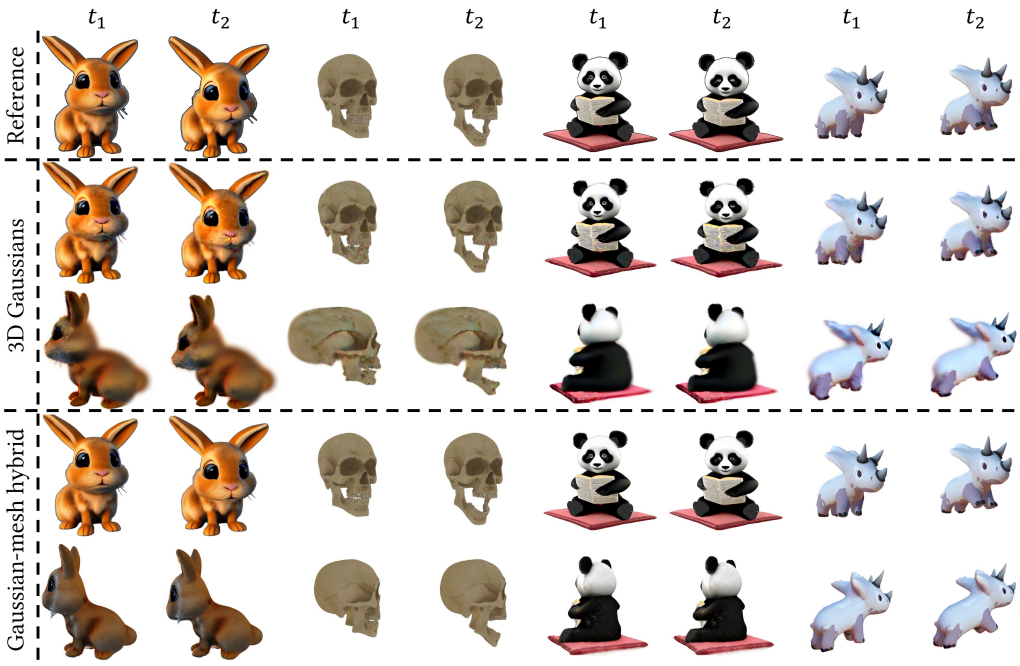

Figure 5: **Qualitative comparison on 3D representation between 3D Gaussians and Gaussian-mesh hybrid representation.** When utilizing 3D Gaussians as our base 3D representation, the texture is blurry on the parts unseen in reference image. As a comparison, the texture is clean and of high quality under every view when employing the Gaussian-mesh hybrid representation.

videos captured under fixed viewpoint. The extension of our framework to scene-level generation or to videos captured with a moving camera remains an area for future exploration. Finally, due to the scarcity of test data, the performance of our method on more complex task is not evaluated. These identified limitations will be addressed in our future research endeavors.

5 Conclusion

In this work, we introduce DreamMesh4D, an innovative video-to-4D framework that generates dynamic meshes through a static-to-dynamic optimization process. By employing a Gaussian-mesh hybrid representation, we simultaneously refine both the geometry and texture of the object. This approach allows the static object to serve as an excellent starting point for dynamic learning. During the dynamic stage, we construct a deformation graph on the object's surface using geodesic distance. Thereafter, the motion of the entire mesh, as well as the surface Gaussians, are driven by sparse control nodes via a novel geometric skinning algorithm named adaptive hybrid skinning. It benefits from the strengths of both Linear Blending Skinning (LBS) and Dual-Quaternion Skinning (DQS), enabling more robust deformation. Extensive experiments have demonstrated the superior performance of our method in generating high-fidelity 4D objects. It significantly surpasses previous methods in both rendering quality and spatial-temporal consistency, establishing a new benchmark for video-to-4D tasks. While our method benefits a lot from the mesh-based representation, it reveals a promising direction in the field of video-to-4D generation. Furthermore, our method's compatibility with modern geometric pipelines showcases its potential applicability in the 3D gaming and film industries.

Acknowledgement

This work was supported in part by National Science and Technology Major Project (2022ZD0115102), in part by National Natural Science Foundation of China (62202389), in part by a grant from the Westlake University-Muyuan Joint Research Institute, and in part by the Westlake Education Foundation.

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. *arXiv preprint arXiv:2311.17984*, 2023.

[2] Yogesh Balaji, Martin Renqiang Min, Bing Bai, Rama Chellappa, and Hans Peter Graf. Conditional gan with discriminative filter generation for text-to-video synthesis. In *IJCAI*, volume 1, page 2, 2019.

[3] Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, et al. Stable video diffusion: Scaling latent video diffusion models to large datasets. *arXiv preprint arXiv:2311.15127*, 2023.

[4] Ang Cao and Justin Johnson. Hexplane: A fast representation for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 130–141, 2023.

[5] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3d: Disentangling geometry and appearance for high-quality text-to-3d content creation. *arXiv preprint arXiv:2303.13873*, 2023.

[6] Jiemin Fang, Taoran Yi, Xinggang Wang, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Matthias Nießner, and Qi Tian. Fast dynamic radiance fields with time-aware neural voxels. In *SIGGRAPH Asia 2022 Conference Papers*, pages 1–9, 2022.

[7] Sara Fridovich-Keil, Giacomo Meanti, Frederik Rahbæk Warburg, Benjamin Recht, and Angjoo Kanazawa. K-planes: Explicit radiance fields in space, time, and appearance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12479–12488, 2023.

[8] Chen Gao, Ayush Saraf, Johannes Kopf, and Jia-Bin Huang. Dynamic view synthesis from dynamic monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5712–5721, 2021.

[9] Antoine Guédon and Vincent Lepetit. Sugar: Surface-aligned gaussian splatting for efficient 3d mesh reconstruction and high-quality mesh rendering. *arXiv preprint arXiv:2311.12775*, 2023.

[10] Yuan-Chen Guo, Ying-Tian Liu, Ruizhi Shao, Christian Laforte, Vikram Voleti, Guan Luo, Chia-Hao Chen, Zi-Xin Zou, Chen Wang, Yan-Pei Cao, and Song-Hai Zhang. threestudio: A unified framework for 3d content generation. https://github.com/threestudio-project/threestudio, 2023.

[11] Yi-Hua Huang, Yang-Tian Sun, Ziyi Yang, Xiaoyang Lyu, Yan-Pei Cao, and Xiaojuan Qi. Sc-gs: Sparse-controlled gaussian splatting for editable dynamic scenes. *arXiv preprint arXiv:2312.14937*, 2023.

[12] Yanqin Jiang, Li Zhang, Jin Gao, Weimin Hu, and Yao Yao. Consistent4d: Consistent 360 {\deg} dynamic object generation from monocular video. *arXiv preprint arXiv:2311.02848*, 2023.

[13] Yuheng Jiang, Zhehao Shen, Penghao Wang, Zhuo Su, Yu Hong, Yingliang Zhang, Jingyi Yu, and Lan Xu. Hifi4g: High-fidelity human performance rendering via compact gaussian splatting. *arXiv preprint arXiv:2312.03461*, 2023.

[14] Erik Johnson, Marc Habermann, Soshi Shimada, Vladislav Golyanik, and Christian Theobalt. Unbiased 4d: Monocular 4d reconstruction with a neural deformation model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6598–6607, 2023.

[15] Oren Katzir, Or Patashnik, Daniel Cohen-Or, and Dani Lischinski. Noise-free score distillation. *arXiv preprint arXiv:2310.17590*, 2023.

[16] Ladislav Kavan, Steven Collins, Jiří Žára, and Carol O'Sullivan. Geometric skinning with approximate dual quaternion blending. *ACM Transactions on Graphics (TOG)*, 27(4):1–23, 2008.

[17] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, page 0, 2006.

[18] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics (ToG)*, 42(4):1–14, 2023.

[19] Weiyu Li, Rui Chen, Xuelin Chen, and Ping Tan. Sweetdreamer: Aligning geometric priors in 2d diffusion for consistent text-to-3d. *arXiv preprint arXiv:2310.02596*, 2023.

[20] Xiaoyu Li, Qi Zhang, Di Kang, Weihao Cheng, Yiming Gao, Jingbo Zhang, Zhihao Liang, Jing Liao, Yan-Pei Cao, and Ying Shan. Advances in 3d generation: A survey. *arXiv preprint arXiv:2401.17807*, 2024.

[21] Zhan Li, Zhang Chen, Zhong Li, and Yi Xu. Spacetime gaussian feature splatting for real-time dynamic view synthesis. *arXiv preprint arXiv:2312.16812*, 2023.

[22] Yiqing Liang, Numair Khan, Zhengqin Li, Thu Nguyen-Phuoc, Douglas Lanman, James Tompkin, and Lei Xiao. Gaufre: Gaussian deformation fields for real-time dynamic novel view synthesis. *arXiv preprint arXiv:2312.11458*, 2023.

[23] Yixun Liang, Xin Yang, Jiantao Lin, Haodong Li, Xiaogang Xu, and Yingcong Chen. Luciddreamer: Towards high-fidelity text-to-3d generation via interval score matching. *arXiv preprint arXiv:2311.11284*, 2023.

[24] Chen-Hsuan Lin, Jun Gao, Luming Tang, Towaki Takikawa, Xiaohui Zeng, Xun Huang, Karsten Kreis, Sanja Fidler, Ming-Yu Liu, and Tsung-Yi Lin. Magic3d: High-resolution text-to-3d content creation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 300–309, 2023.

[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. *arXiv preprint arXiv:2312.13763*, 2023.

[26] Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. Zero-1-to-3: Zero-shot one image to 3d object. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9298–9309, 2023.

[27] Yuan Liu, Cheng Lin, Zijiao Zeng, Xiaoxiao Long, Lingjie Liu, Taku Komura, and Wenping Wang. Syncdreamer: Generating multiview-consistent images from a single-view image. *arXiv preprint arXiv:2309.03453*, 2023.

[28] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. *arXiv preprint arXiv:2308.09713*, 2023.

[29] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and mathematics III*, pages 35–57. Springer, 2003.

[30] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65 (1):99–106, 2021.

[31] OpenAI. Dall-e 3. https://openai.com/dall-e-3.

[32] Keunhong Park, Utkarsh Sinha, Jonathan T Barron, Sofien Bouaziz, Dan B Goldman, Steven M Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5865–5874, 2021.

[33] Ben Poole, Ajay Jain, Jonathan T Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion. In *ICLR*, 2023.

[34] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10318–10327, 2021.

[35] Guocheng Qian, Jinjie Mai, Abdullah Hamdi, Jian Ren, Aliaksandr Siarohin, Bing Li, Hsin-Ying Lee, Ivan Skorokhodov, Peter Wonka, Sergey Tulyakov, et al. Magic123: One image to high-quality 3d object generation using both 2d and 3d diffusion priors. *arXiv preprint arXiv:2306.17843*, 2023.

[36] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[37] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv preprint arXiv:2007.08501*, 2020.

[38] 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.

[39] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[40] Nadine Abu Rumman and Marco Fratarcangeli. State of the art in skinning techniques for articulated deformable characters. In *International Conference on Computer Graphics Theory and Applications*, volume 2, pages 200–212. SciTePress, 2016.

[41] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494, 2022.

[42] Tianchang Shen, Jun Gao, Kangxue Yin, Ming-Yu Liu, and Sanja Fidler. Deep marching tetrahedra: a hybrid representation for high-resolution 3d shape synthesis. *Advances in Neural Information Processing Systems*, 34:6087–6101, 2021.

[43] Ruoxi Shi, Hansheng Chen, Zhuoyang Zhang, Minghua Liu, Chao Xu, Xinyue Wei, Linghao Chen, Chong Zeng, and Hao Su. Zero123++: a single image to consistent multi-view diffusion base model. *arXiv preprint arXiv:2310.15110*, 2023.

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

[45] Uriel Singer, Shelly Sheynin, Adam Polyak, Oron Ashual, Iurii Makarov, Filippos Kokkinos, Naman Goyal, Andrea Vedaldi, Devi Parikh, Justin Johnson, et al. Text-to-4d dynamic scene generation. *arXiv preprint arXiv:2301.11280*, 2023.

[46] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, pages 109–116. Citeseer, 2007.

[47] Robert W Sumner, Johannes Schmid, and Mark Pauly. Embedded deformation for shape manipulation. In *ACM siggraph 2007 papers*, pages 80–es. 2007.

[48] Jingxiang Sun, Bo Zhang, Ruizhi Shao, Lizhen Wang, Wen Liu, Zhenda Xie, and Yebin Liu. Dreamcraft3d: Hierarchical 3d generation with bootstrapped diffusion prior. *arXiv preprint arXiv:2310.16818*, 2023.

[49] Jiaxiang Tang, Jiawei Ren, Hang Zhou, Ziwei Liu, and Gang Zeng. Dreamgaussian: Generative gaussian splatting for efficient 3d content creation. *arXiv preprint arXiv:2309.16653*, 2023.

[50] Junshu Tang, Tengfei Wang, Bo Zhang, Ting Zhang, Ran Yi, Lizhuang Ma, and Dong Chen. Make-it-3d: High-fidelity 3d creation from a single image with diffusion prior. *arXiv preprint arXiv:2303.14184*, 2023.

[51] Edgar Tretschk, Ayush Tewari, Vladislav Golyanik, Michael Zollhöfer, Christoph Lassner, and Christian Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a dynamic scene from monocular video. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 12959–12970, 2021.

[52] Thomas Unterthiner, Sjoerd Van Steenkiste, Karol Kurach, Raphael Marinier, Marcin Michalski, and Sylvain Gelly. Towards accurate generative models of video: A new metric & challenges. *arXiv preprint arXiv:1812.01717*, 2018.

[53] Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. https://github.com/huggingface/diffusers, 2022.

[54] Jiuniu Wang, Hangjie Yuan, Dayou Chen, Yingya Zhang, Xiang Wang, and Shiwei Zhang. Modelscope text-to-video technical report. *arXiv preprint arXiv:2308.06571*, 2023.

[55] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *arXiv preprint arXiv:2106.10689*, 2021.

[56] Yiming Wang, Qin Han, Marc Habermann, Kostas Daniilidis, Christian Theobalt, and Lingjie Liu. Neus2: Fast learning of neural implicit surfaces for multi-view reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3295–3306, 2023.

[57] Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan Li, Hang Su, and Jun Zhu. Prolificdreamer: High-fidelity and diverse text-to-3d generation with variational score distillation. *arXiv preprint arXiv:2305.16213*, 2023.

[58] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Xinggang Wang. 4d gaussian splatting for real-time dynamic scene rendering. *arXiv preprint arXiv:2310.08528*, 2023.

[59] Tianhao Wu, Fangcheng Zhong, Andrea Tagliasacchi, Forrester Cole, and Cengiz Oztireli. D^2nerf: Self-supervised decoupling of dynamic and static objects from a monocular video. *Advances in neural information processing systems*, 35:32653–32666, 2022.

[60] Zijie Wu, Chaohui Yu, Yanqin Jiang, Chenjie Cao, Fan Wang, and Xiang Bai. Sc4d: Sparse-controlled video-to-4d generation and motion transfer. *arXiv preprint arXiv:2404.03736*, 2024.

[61] Jiale Xu, Weihao Cheng, Yiming Gao, Xintao Wang, Shenghua Gao, and Ying Shan. Instantmesh: Efficient 3d mesh generation from a single image with sparse-view large reconstruction models. *arXiv preprint arXiv:2404.07191*, 2024.

[62] Zeyu Yang, Zijie Pan, Chun Gu, and Li Zhang. Diffusion^2: Dynamic 3d content generation via score composition of orthogonal diffusion models. *arXiv preprint arXiv:2404.02148*, 2024.

[63] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. *arXiv preprint arXiv:2309.13101*, 2023.

[64] Taoran Yi, Jiemin Fang, Guanjun Wu, Lingxi Xie, Xiaopeng Zhang, Wenyu Liu, Qi Tian, and Xinggang Wang. Gaussiandreamer: Fast generation from text to 3d gaussian splatting with point cloud priors. *arXiv preprint arXiv:2310.08529*, 2023.

[65] 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.

[66] Yifei Zeng, Yanqin Jiang, Siyu Zhu, Yuanxun Lu, Youtian Lin, Hao Zhu, Weiming Hu, Xun Cao, and Yao Yao. Stag4d: Spatial-temporal anchored generative 4d gaussians. *arXiv preprint arXiv:2403.14939*, 2024.

[67] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018.

[68] Ruowen Zhao, Zhengyi Wang, Yikai Wang, Zihan Zhou, and Jun Zhu. Flexidreamer: Single image-to-3d generation with flexicubes. *arXiv preprint arXiv:2404.00987*, 2024.

[69] 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.

A Appendix / supplemental material

A.1 Additional Qualitative Results

Here we provide more qualitative comparisons of our method against baseline works in Fig. 6. Please see our project page for more temporal qualitative results with video format.

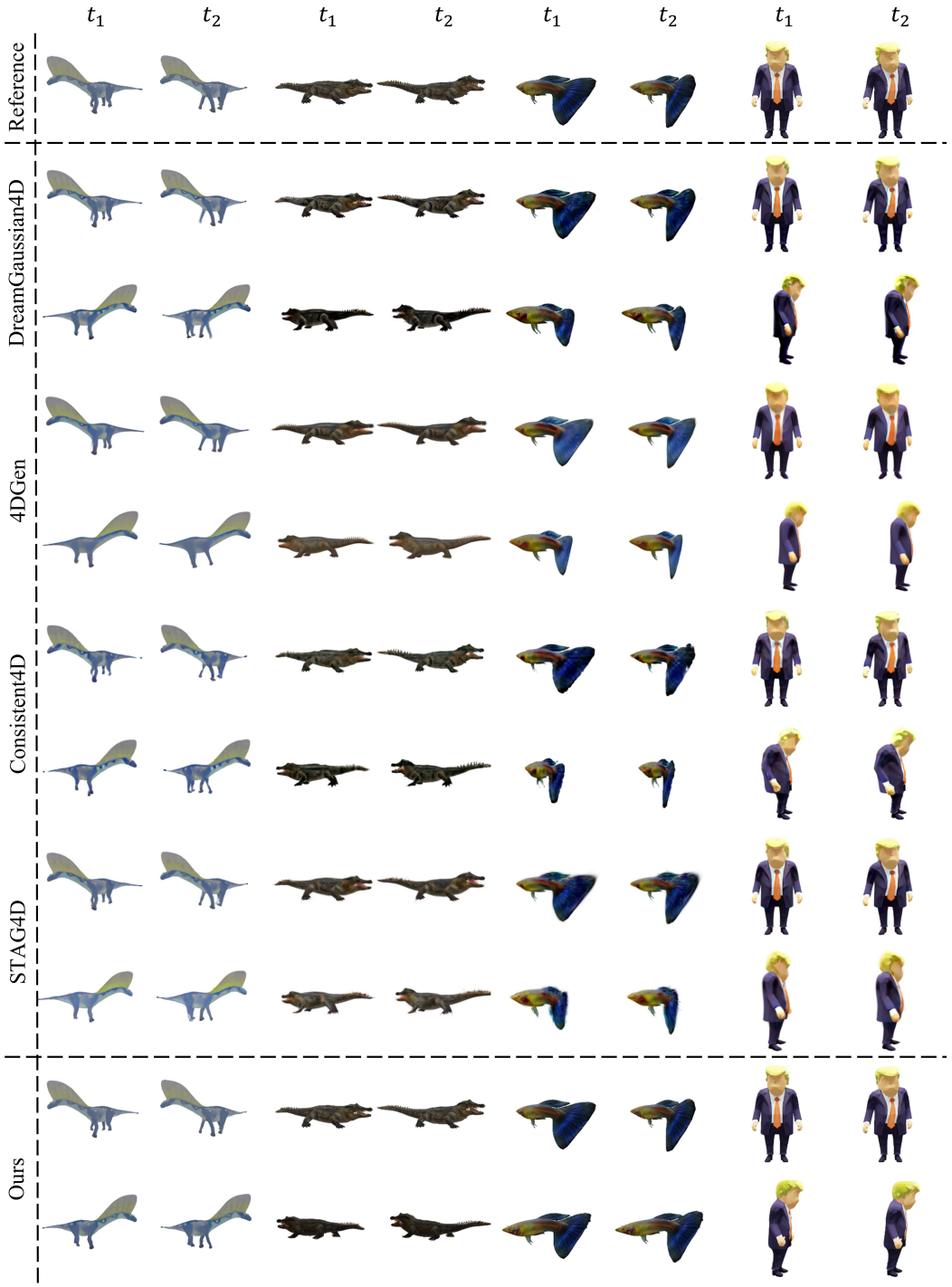

Figure 6: **Additional qualitative comparison with baselines.**

A.2 Implementation Details

Static Stage In the process of both coarse mesh generation and SuGaR refinement, the loss in Equation 3 is utilized for supervision, with the strength of different terms as $\lambda^s_{SDS} = 0.1$, $\lambda^s_{ref} = 1000$ and $\lambda^s_{mask} = 500$. For the generation of coarse mesh, a set of randomly initialized 3D Gaussians are optimized for 3000 steps in total. In the former 1500 steps, we do densification and pruning every 100 steps. After the 1500 steps, densification and pruning are stopped, and we introduce additional opacity binary and density regularization terms as described in [9] into optimization until step 3000. Finally, we prune all Gaussians with opacity less than 0.5 and extract coarse mesh through Poisson reconstruction. Afterwards, we bind $x = 6$ new flat Gaussians to each triangle face of the coarse mesh, and do optimization for 2000 steps.

Dynamic Stage In dynamic stage, we defaultly sample $N_{node} = 1024$ control nodes and assign $N_{neighbor} = 4$ neighboring nodes for each vertex when constructing deformation graph. For each training step, 8 frames are randomly sampled from the input video for supervision. And for each sampled timestamp, we randomly sample 2 views for the calculation of SDS loss. All images are rendered at resolution 512×512 with white background. The camera distance to world center is fixed as 3.8 and the degree of field-of-view (FoV) is fixed as $20°$. As for the strenghts of different loss terms, we defautly set $\lambda_{SDS} = 0.1$, $\lambda_{ref} = 5000$, $\lambda_{mask} = 500$ and $\lambda_{NC} = 10$. The value of λ_{ARAP} is chosen case-specifically in [1, 10] according to the motion amplitude of object. The deformation network is zero-ly initialized and totally optimized for 2000 steps with learning rate as 0.00032. All of our experiments are conducted on a single NVIDIA RTX 4090 GPU.

Licenses Here we provide the URL, citations, and licenses of open-sourced assets used in this work in Table 3.

URL	Citation	License
https://github.com/threestudio-project/threestudio	[10]	Apache-2.0 license
https://github.com/huggingface/diffusers	[53]	Apache-2.0 license
https://github.com/facebookresearch/pytorch3d	[37]	BSD License
https://github.com/cvlab-columbia/zero123	[26]	MIT License
https://github.com/Anttwo/SuGaR	[9]	Gaussian-Splatting License

Table 3: **URL, citations and licenses of the open-sourced assets used in this work.**

A.3 Additional Experiments

	Consistent4D	DreamGaussian4D	4DGen	STAG4D	Ours
Training Time	2.0h	0.6h	3.0h	1.6h	0.8h
Memory	28GB	20GB	15GB	7GB	8GB

Table 4: Comparison of computation cost of different methods.

Computation Cost In Table 4, we report the computation cost of our method and other compared baseline methods, demonstrating the computation efficiency of our method.

#Gaussians per face	PSNR(ref)↑	SSIM(ref)↑	LPIPS↓	FVD↓	FID-VID↓	CLIP↑
1	36.17	0.977	0.134	523.39	27.18	0.939
3	36.55	0.979	0.129	496.46	27.60	**0.943**
4	36.60	0.979	0.128	519.35	26.81	0.940
6	**36.63**	**0.979**	**0.127**	**477.63**	**25.96**	0.940

Table 5: Quantitative evaluation of ablation study on the number of Gaussians per triangle face.

Ablation on Number of Face Gaussians We also conduct comparisons on different number of Gaussians per face. The qualitative and quantitative comparison results are presented in Fig. 7 and Table A.3. The results demonstrate that the more Gaussains utilized per triangle face, the more detailed appearance can be obtained (e.g., the eyes and nose of the dog). Empirically we find that 6-Gaussians-per-face can already deliver satisfying performance by considering the rendering quality and training time. Hence we keep this setup for all experiment cases.

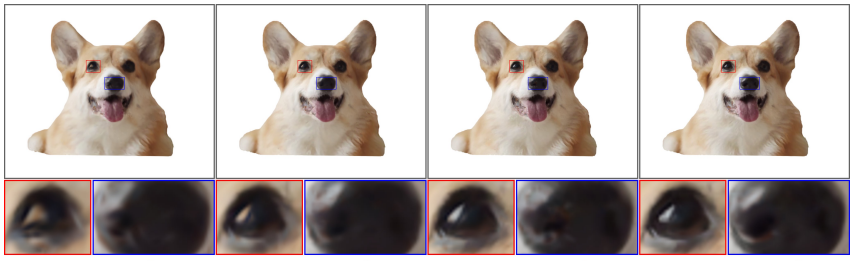

Figure 7: **Qualitative comparison on the number of Gaussians per face.** The appearance quality of details (e.g. the eyes and nose) is getting better when binding more number of Gaussians on triangle face.

Ablation on Number of Control Nodes Here we conduct an additional ablation study on the choice of the number of control nodes, N_{node}, and the number of connected nodes for each vertex, $N_{neighbor}$. We try [256, 512, 1024] for N_{node} and [4, 8, 16] for $N_{neighbor}$, and the quantitative results are presented in Table 6. There are no significant distinct on scores of different metrics under different combination of N_{node} and $N_{neighbor}$, except for PSNR(ref), on which $\{N_{node} = 1024, N_{neighbor} = 4\}$ achieves the highest score. While PSNR(ref) is a key metric revealing reconstruction quality, we pick $\{N_{node} = 1024, N_{neighbor} = 4\}$ as the default setup for our method.

	PSNR(ref)	SSIM(ref)	LPIPS	FVD	FID-VID	CLIP
$N_{node} = 1024, N_{neighbor} = 16$	36.39	0.979	0.128	559.43	32.08	0.932
$N_{node} = 1024, N_{neighbor} = 8$	36.70	0.980	0.128	521.15	30.94	0.936
$N_{node} = 1024, N_{neighbor} = 4$	**37.82**	0.980	0.128	516.46	31.20	0.937
$N_{node} = 512, N_{neighbor} = 16$	36.14	0.979	0.127	542.32	**30.19**	0.937
$N_{node} = 512, N_{neighbor} = 8$	36.28	0.979	0.127	**505.86**	30.34	0.935
$N_{node} = 512, N_{neighbor} = 4$	36.42	0.979	0.127	511.78	30.59	0.936
$N_{node} = 256, N_{neighbor} = 8$	35.92	0.978	0.129	512.25	32.21	0.935
$N_{node} = 256, N_{neighbor} = 4$	35.97	0.979	0.127	522.46	30.55	0.935

Table 6: **Quantitative results of setup on number of control nodes and mesh vertex connectivity.**