Align Your Gaussians:
Text-to-4D with Dynamic 3D Gaussians and Composed Diffusion Models

Huan Ling[1,2,3] * Seung Wook Kim[1,2,3] * Antonio Torralba[4] Sanja Fidler[1,2,3] Karsten Kreis[1]

[1]NVIDIA [2]Vector Institute [3]University of Toronto [4]MIT

Project page: https://research.nvidia.com/labs/toronto-ai/AlignYourGaussians/

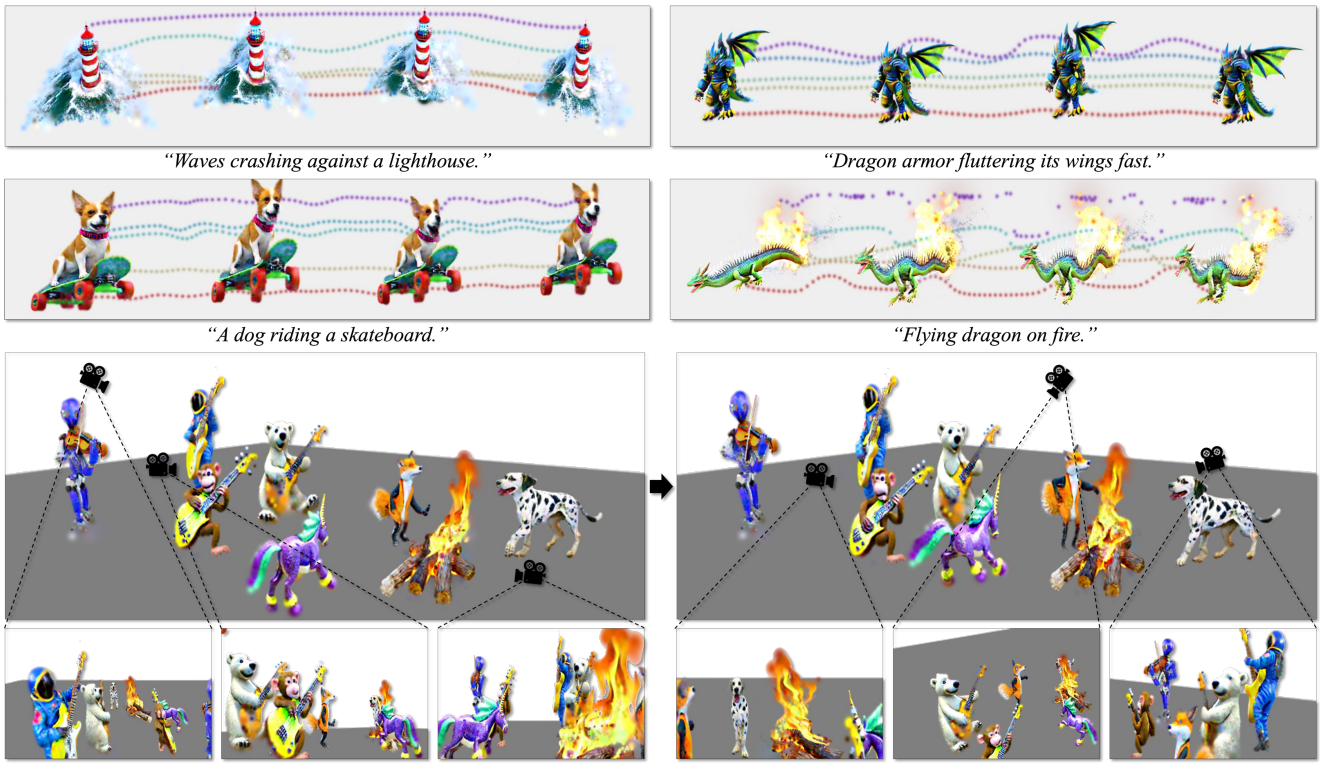

Figure 1. **Text-to-4D synthesis with *Align Your Gaussians (AYG)*.** *Top:* Different dynamic 4D sequences. Dotted lines represent dynamics of deformation field. *Bottom:* Multiple dynamic 4D objects are composed within a large dynamic scene; two time frames shown.

Abstract

*Text-guided diffusion models have revolutionized image and video generation and have also been successfully used for optimization-based 3D object synthesis. Here, we instead focus on the underexplored text-to-4D setting and synthesize dynamic, animated 3D objects using score distillation methods with an additional temporal dimension. Compared to previous work, we pursue a novel compositional generation-based approach, and combine text-to-image, text-to-video, and 3D-aware multiview diffusion models to provide feedback during 4D object optimization, thereby simultaneously enforcing temporal consistency, high-quality visual appearance and realistic geometry. Our method, called **Align Your Gaussians (AYG)**, leverages dynamic 3D Gaussian Splatting with deformation fields as 4D representation. Crucial to AYG is a novel method to regularize the distribution of the moving 3D Gaussians and thereby stabilize the optimization and induce motion. We also propose a motion amplification mechanism as well as a new autoregressive synthesis scheme to generate and combine multiple 4D sequences for longer generation. These techniques allow us to synthesize vivid dynamic scenes, outperform previous work qualitatively and quantitatively and achieve state-of-the-art text-to-4D performance. Due to the Gaussian 4D representation, different 4D animations can be seamlessly combined, as we demonstrate. AYG opens up promising avenues for animation, simulation and digital content creation as well as synthetic data generation.*

1. Introduction

Generative modeling of dynamic 3D scenes has the potential to revolutionize how we create games, movies, simu-

*Equal contribution.

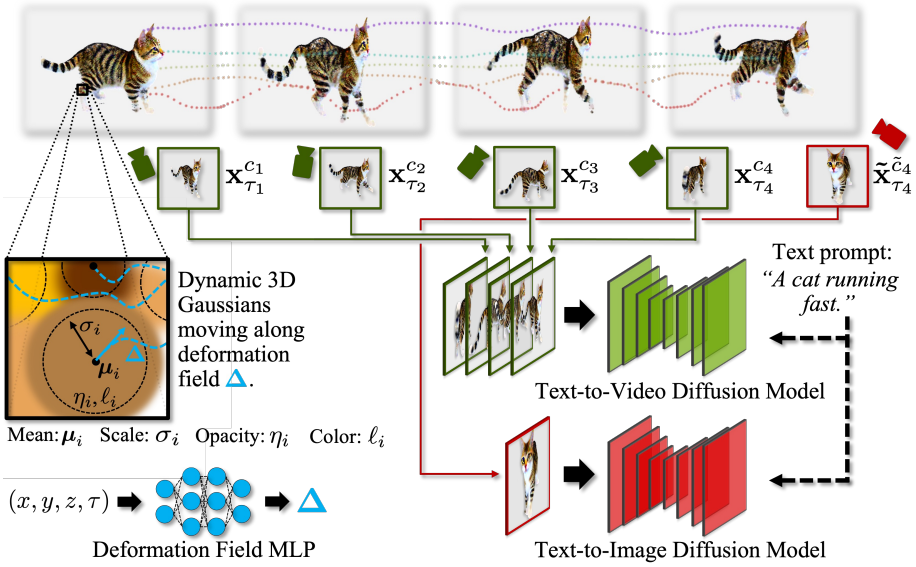

Figure 2. **Text-to-4D synthesis with AYG.** We generate dynamic 4D scenes via score distillation. We initialize the 4D sequence from a static 3D scene (generated first, Fig. 3), which is represented by 3D Gaussians with means $\boldsymbol{\mu}_i$, scales σ_i, opacities η_i and colors ℓ_i. Consecutive rendered frames $\mathbf{x}_{\tau_j}^{c_j}$ from the 4D sequence at times τ_j and camera positions c_j are diffused and fed to a text-to-video diffusion model [7] (**green arrows**), which provides a distillation gradient that is backpropagated through the rendering process into a deformation field $\Delta(x, y, z, \tau)$ (dotted lines) that captures scene motion. Simultaneously, random frames $\tilde{\mathbf{x}}_{\tau_j}^{\tilde{c}_j}$ are diffused and given to a text-to-image diffusion model [70] (**red arrows**) whose gradients ensure that high visual quality is maintained frame-wise.

lations, animations and entire virtual worlds. Many works have shown how a wide variety of 3D objects can be synthesized via score distillation techniques [10, 11, 31, 41, 52, 62, 79, 85, 88, 92, 109], but they typically only synthesize static 3D scenes, although we live in a moving, dynamic world. While image diffusion models have been successfully extended to video generation [1, 7, 22, 28, 78, 90, 91, 107], there is little research on similarly extending 3D synthesis to 4D generation with an additional temporal dimension.

We propose *Align Your Gaussians (AYG)*, a novel method for 4D content creation. In contrast to previous work [79], we leverage dynamic 3D Gaussians [36] as backbone 4D representation, where a deformation field [59, 63] captures scene dynamics and transforms the collection of 3D Gaussians to represent object motion. AYG takes a compositional generation-based perspective and leverages the combined gradients of latent text-to-image [70], text-to-video [7] and 3D-aware text-to-multiview-image [76] diffusion models in a score distillation-based synthesis framework. A 3D-aware multiview diffusion model and a regular text-to-image model are used to generate an initial high-quality 3D shape. Afterwards, we compose the gradients of a text-to-video and a text-to-image model; the gradients of the text-to-video model optimize the deformation field to capture temporal dynamics, while the text-to-image model ensures that high visual quality is maintained for all time frames (Fig. 2). To this end, we trained a dedicated text-to-video model; it is conditioned on the frame rate and can create useful gradients both for short and long time intervals, which allows us to generate long and smooth 4D sequences.

We developed several techniques to ensure stable optimization and learn vivid dynamic 4D scenes in AYG: We employ a novel regularization method that uses a modified version of the Jensen-Shannon divergence to regularize the locations of the 3D Gaussians such that the mean and variance of the set of 3D Gaussians is preserved as they move. Furthermore, we use a motion amplification method that carefully scales the gradients from the text-to-video model and enhances motion. To extend the length of the 4D sequences or combine different dynamic scenes with changing text guidance, we introduce an autoregressive generation scheme which interpolates the deformation fields of consecutive sequences. We also propose a new view-guidance method to generate consistent 3D scenes for initialization of the 4D stage, and we leverage the concurrent classifier score distillation method [102].

We find that AYG can generate diverse, vivid, detailed and 3D-consistent dynamic scenes (Fig. 1), achieving state-of-the-art text-to-4D performance. We also show long, autoregressively extended 4D scenes, including ones with varying text guidance, which has not been demonstrated before. A crucial advantage of AYG's 4D Gaussian backbone representation is that different 4D animations can trivially be combined and composed together, which we also show.

We envision broad applications in digital content creation, where AYG takes a step beyond the literature on text-to-3D and captures our world's rich dynamics. Moreover, AYG can generate 4D scenes with exact tracking labels for free, a promising feature for synthetic data generation.

Contributions. *(i)* We propose AYG, a system for text-to-4D content creation leveraging dynamic 3D Gaussians with deformation fields as 4D representation. *(ii)* We show how to tackle the text-to-4D task through score distillation within a new compositional generation framework, combining 2D, 3D, and video diffusion models. *(iii)* To scale AYG, we introduce a novel regularization method and a new motion amplification technique. *(iv)* Experimentally, we achieve state-of-the-art text-to-4D performance and generate high-quality, diverse, and dynamic 4D scenes. *(v)* For the first time, we also show how our 4D sequences can be extended in time with a new autoregressive generation scheme and even creatively composed in large scenes.

2. Background

3D Gaussian Splatting [36] represents 3D scenes by N 3D Gaussians with positions $\boldsymbol{\mu}_i$, covariances Σ_i, opacities η_i and colors ℓ_i (Fig. 2). Rendering corresponds to projection of the 3D Gaussians onto the 2D camera's image plane, producing 2D Gaussians with projected means $\hat{\boldsymbol{\mu}}_i$ and covariances $\hat{\Sigma}_i$. The color $\mathcal{C}(\mathbf{p})$ of image pixel $\mathbf{p}$ can be calculated through point-based volume rendering [111] as

$$\mathcal{C}(\mathbf{p}) = \sum_{i=1}^{N} \ell_i \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j), \tag{1}$$

$$\alpha_i = \eta_i \exp\left[-\frac{1}{2} (\mathbf{p} - \hat{\boldsymbol{\mu}}_i)^\top \hat{\Sigma}_i^{-1} (\mathbf{p} - \hat{\boldsymbol{\mu}}_i)\right], \tag{2}$$

where j iterates over the Gaussians along the ray through the scene from pixel $\mathbf{p}$ until Gaussian i. To accelerate rendering, the image plane can be divided into tiles, which are processed in parallel. Initially proposed for 3D scene reconstruction, 3D Gaussian Splatting uses gradient-based thresholding to densify areas that need more Gaussians to capture fine details, and unnecessary Gaussians with low opacity are pruned every few thousand optimization steps.

Diffusion Models and Score Distillation Sampling. Diffusion-based generative models (DMs) [18, 27, 57, 80, 81] use a forward diffusion process that gradually perturbs data, such as images or entire videos, towards entirely random noise, while a neural network is learnt to denoise and reconstruct the data. DMs have also been widely used for score distillation-based generation of 3D objects [62]. In that case, a 3D object, represented for instance by a neural radiance field (NeRF) [54] or 3D Gaussians [36], like here, with parameters $\boldsymbol{\theta}$ is rendered from different camera views and the renderings $\mathbf{x}$ are diffused and given to a text-to-image DM. In the score distillation sampling (SDS) framework, the DM's denoiser is then used to construct a gradient that is backpropagated through the differentiable rendering process g into the 3D scene representation and updates the scene representation to make the scene rendering look more realistic, like images modeled by the DM. Rendering and using DM feedback from many different camera perspectives then encourages the scene representation to form a geometrically consistent 3D scene. The SDS gradient [62] is

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_{\text{SDS}}(\mathbf{x} = g(\boldsymbol{\theta})) = \mathbb{E}_{t,\boldsymbol{\epsilon}} \left[w(t) \left(\hat{\boldsymbol{\epsilon}}_\phi(\mathbf{z}_t, v, t) - \boldsymbol{\epsilon} \right) \frac{\partial \mathbf{x}}{\partial \boldsymbol{\theta}} \right],$$

where $\mathbf{x}$ denotes the 2D rendering, t is the time up to which the diffusion is run to perturb $\mathbf{x}$, $w(t)$ is a weighting function, and $\mathbf{z}_t$ is the perturbed rendering. Further, $\hat{\boldsymbol{\epsilon}}_\phi(\mathbf{z}_t, v, t)$ is the DM's denoiser neural network that predicts the diffusion noise $\boldsymbol{\epsilon}$. It is conditioned on $\mathbf{z}_t$, the diffusion time t and a text prompt v for guidance. Classifier-free guidance (CFG) [26] typically amplifies the text conditioning.

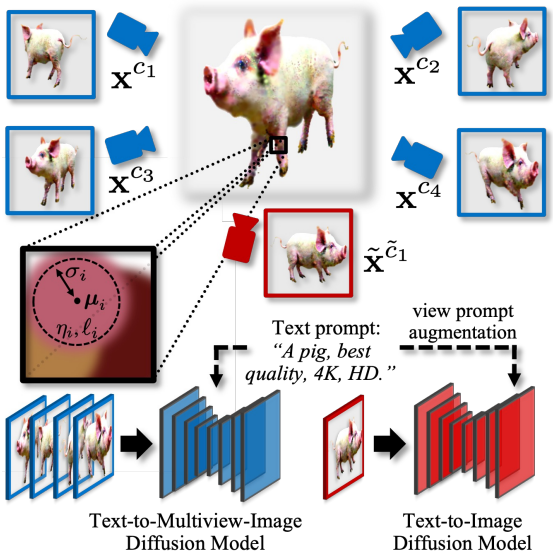

Figure 3. In **AYG's initial 3D stage** we synthesize a static 3D scene leveraging a text-guided multiview diffusion model [76] and a regular text-to-image model [70]. The text-to-image model receives viewing angle-dependent text prompts and leverages view guidance (Sec. 3.4). See Fig. 2 for 4D stage and descriptions.

2.1. Related Work

See Supp. Material for an extended discussion. Here, we only briefly mention the most relevant related literature.

As discussed, AYG builds on text-driven image [5, 14, 21, 61, 67, 70, 72, 98], video [1, 7, 22, 25, 28, 38, 78, 90, 91, 94, 107] and 3D-aware DMs [42, 44, 45, 56, 64, 75, 76, 104], uses score distillation sampling [10, 11, 17, 31, 41, 48, 52, 62, 85, 88, 92, 96, 109] and leverages 3D Gaussian Splatting [36] as well as deformation fields [8, 59, 60, 63, 84] for its 4D representation. The concurrent works DreamGaussian [83], GSGEN [12] and GaussianDreamer [101] use 3D Gaussian Splatting to synthesize static 3D scenes, but do not consider dynamics. Dynamic 3D Gaussian Splatting has been used for 4D reconstruction [50, 93, 110], but not for 4D generation. The idea to compose the gradients of multiple DMs has been used before for controllable image generation [19, 43], but has received little attention in 3D or 4D synthesis.

Most related to AYG is *Make-A-Video3D (MAV3D)* [79], to the best of our knowledge the only previous work that generates dynamic 4D scenes with score distillation. MAV3D uses NeRFs with HexPlane [9] features as 4D representation, in contrast to AYG's dynamic 3D Gaussians, and it does not disentangle its 4D representation into a static 3D representation and a deformation field modeling dynamics. MAV3D's representation prevents it from composing multiple 4D objects into large dynamic scenes, which our 3D Gaussian plus deformation field representation easily enables, as we show. Moreover, MAV3D's sequences are limited in time, while we show a novel autoregressive generation scheme to extend our 4D sequences.

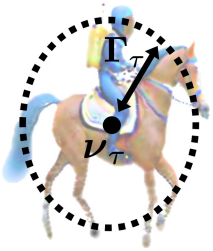

Figure 4. **AYG's JSD-based regularization** of the evolving 4D Gaussians (see Sec. 3.4) calculates the 3D mean $\boldsymbol{\nu}_\tau$ and diagonal covariance matrix $\boldsymbol{\Gamma}_\tau$ of the set of dynamic 3D Gaussians at different times τ of the 4D sequence and regularizes them to not vary too much.

AYG outperforms MAV3D qualitatively and quantitatively and synthesizes significantly higher-quality 4D scenes. Our novel compositional generation-based approach contributes to this, which MAV3D does not pursue. Finally, instead of regular SDS, used by MAV3D, in practice AYG employs classifier score distillation [102] (see Sec. 3.4).

3. Align Your Gaussians

In Sec. 3.1, we present AYG's 4D representation, and in Sec. 3.2, we introduce its compositional generation framework with multiple DMs. In Sec. 3.3, we lay out AYG's score distillation framework in practice, and in Sec. 3.4, we discuss several novel methods and extensions to scale AYG.

3.1. AYG's 4D Representation

AYG's 4D representation combines 3D Gaussian Splatting [36] with deformation fields [59, 63] to capture the 3D scene and its temporal dynamics in a disentangled manner. Specifically, each 4D scene consists of a set of N 3D Gaussians as in Sec. 2. Following Kerbl et al. [36], we also use two degrees of spherical harmonics to model view-dependent effects, this is, directional color, and thereby improve the 3D Gaussians' expressivity. Moreover, we restrict the 3D Gaussians' covariance matrices to be isotropic with scales σ_i. We made this choice as our 3D Gaussians move as a function of time and learning expressive dynamics is easier for spherical Gaussians. We denote the collection of learnable parameters of our 3D Gaussians as $\boldsymbol{\theta}$. The scene dynamics are modeled by a deformation field $\boldsymbol{\Delta}_{\boldsymbol{\Phi}}(x, y, z, \tau) = (\Delta x, \Delta y, \Delta z)$, defined through a multi-layer perceptron (MLP) with parameters $\boldsymbol{\Phi}$. Specifically, for any 3D location (x, y, z) and time τ, the deformation field predicts a displacement $(\Delta x, \Delta y, \Delta z)$. The 3D Gaussians smoothly follow these displacements to represent a moving and deforming 4D scene (Fig. 2). Note that in practice we preserve the initial 3D Gaussians for the first frame, *i.e.* $\boldsymbol{\Delta}_{\boldsymbol{\Phi}}(x, y, z, 0) = (0, 0, 0)$, by setting $\boldsymbol{\Delta}_{\boldsymbol{\Phi}}(x, y, z, \tau) = (\xi(\tau)\Delta x, \xi(\tau)\Delta y, \xi(\tau)\Delta z)$ where $\xi(\tau) = \tau^{0.35}$ such that $\xi(0) = 0$ and $\xi(1) = 1$. Following Luiten et al. [50], we regularize the deformation field so that nearby Gaussians deform similarly ("rigidity regularization", see Supp. Mat.).

Apart from the intuitive decomposition into a backbone 3D representation and a deformation field to model dynamics, a crucial advantage of AYG's dynamic 3D Gaussian-based representation is that different dynamic scenes, each with its own set of Gaussians and deformation field, can be easily combined, thereby enabling promising 3D dynamic content creation applications (see Fig. 1). This is due to the explicit nature of this representation, in contrast to typical NeRF-based representations. Moreover, learning 4D scenes with score distillation requires many scene renderings. This also makes 3D Gaussians ideal due to their rendering efficiency [36]. Note that early on we also explored MAV3D's HexPlane- and NeRF-based 4D representation [79], but we were not able to achieve satisfactory results.

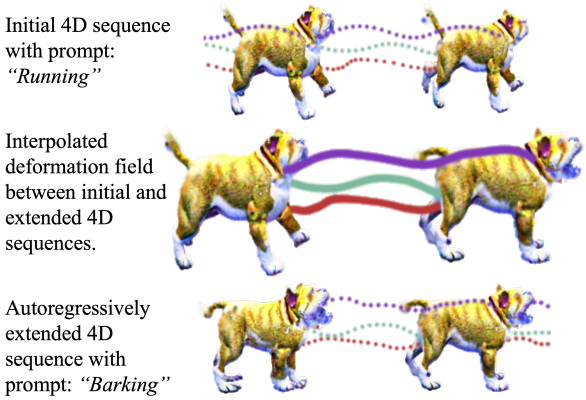

Figure 5. **AYG's autoregressive extension scheme** interpolates the deformation fields of an initial and an extended 4D sequence within an overlap interval between the two sequences (Sec. 3.4).

3.2. Text-to-4D as Compositional Generation

We would like AYG's synthesized dynamic 4D scenes to be of high visual quality, be 3D-consistent and geometrically correct, and also feature expressive and realistic temporal dynamics. This suggests to compose different text-driven DMs during the distillation-based generation to capture these different aspects. *(i)* We use the text-to-image model Stable Diffusion (SD) [70], which has been trained on a broad set of imagery and provides a strong general image prior. *(ii)* We also utilize the 3D-aware text-conditioned multi-view DM MVDream [76], which generates multi-view images of 3D objects, was fine-tuned from SD on the object-centric 3D dataset Objaverse [15, 16] and provides a strong 3D prior. It defines a distribution over four multiview-consistent images corresponding to object renderings from four different camera perspectives $c_1, ..., c_4$. Moreover, we train a text-to-video DM, following VideoLDM [7], but with a larger text-video dataset (HDVG-130M [90] and Webvid-10M [4]) and additional conditioning on the videos' frame rate (see Supp. Material for details). This video DM provides temporal feedback when rendering 2D frame sequences from our dynamic 4D scenes. All used DMs are *latent* DMs [70, 86], which means that in practice we first encode renderings of our 4D scenes into the models' latent spaces, calculate score distillation gradients there, and backpropagate them through the models' encoders. All DMs leverage the SD 2.1 backbone and share the same encoder. To keep the notation simple, we do

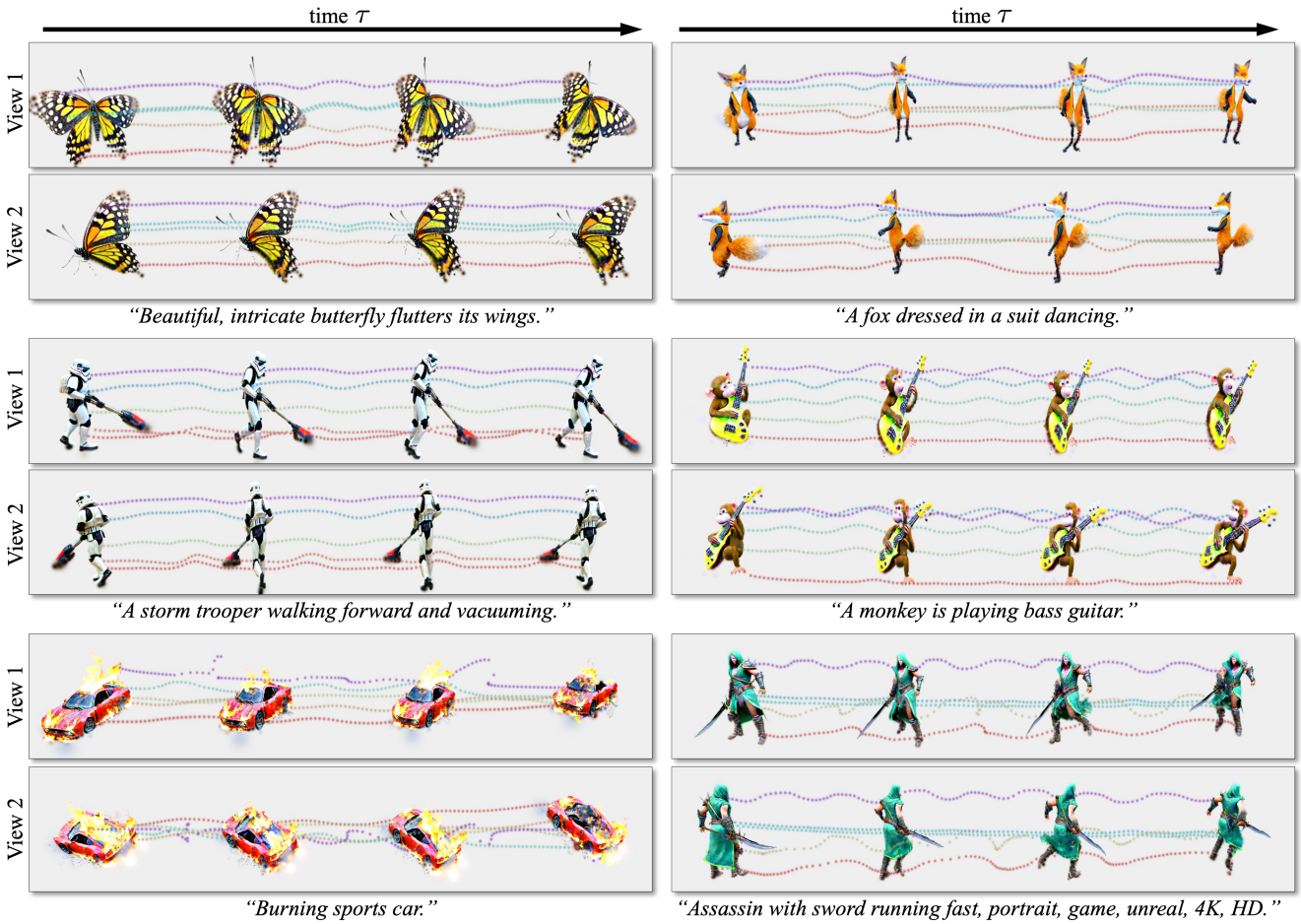

Figure 6. **Text-to-4D synthesis with AYG.** Various samples shown in two views each. Dotted lines denote deformation field dynamics.

not explicitly incorporate the encoding into our mathematical description below and the visualizations (Figs. 2 and 3).

We disentangle optimization into first synthesizing a static 3D Gaussian-based object $\boldsymbol{\theta}$, and then learning the deformation field $\boldsymbol{\Phi}$ to add scene dynamics.

Stage 1: 3D Synthesis (Fig. 3). We first use MVDream's multiview image prior to generate a static 3D scene via score distillation (Supp. Mat. for details). Since MVDream on its own would generate objects in random orientations, we enforce a canonical pose by combining MVDream's gradients with those of regular SD, while augmenting the text-conditioning for SD with directional texts "front view", "side view", "back view" and "overhead view" [62]. Formally, we can derive a score distillation gradient (see Sec. 3.3) by minimizing the reverse Kulback-Leibler divergence (KLD) from the rendering distribution to the product of the composed MVDream and SD model distributions

$$\mathrm{KL}\left(q_{\boldsymbol{\theta}}\left(\{\mathbf{z}^{c_i}\}_4, \{\tilde{\mathbf{z}}^{\tilde{c}_j}\}_K\right) \middle|\middle| p_{\mathrm{3D}}^{\alpha}\left(\{\mathbf{z}^{c_i}\}_4\right) \prod_{j=1}^{K} p_{\mathrm{im}}^{\beta}\left(\tilde{\mathbf{z}}^{\tilde{c}_j}\right)\right),$$

similar to Poole et al. [62] (App. A.4). Here, $p_{\mathrm{3D}}(\{\mathbf{z}^{c_i}\}_4)$ represents the MVDream-defined multiview image distribution over four diffused renderings from camera views c_i, denoted as the set $\{\mathbf{z}^{c_i}\}_4$ (we omit the diffusion time t subscript for brevity). Moreover, $p_{\mathrm{im}}(\tilde{\mathbf{z}}^{\tilde{c}_j})$ is the SD-based general image prior and $\{\tilde{\mathbf{z}}^{\tilde{c}_j}\}_K$ is another set of K diffused scene renderings. In principle, the renderings for SD and MVDream can be from different camera angles c_i and $\tilde{c}_j$, but in practice we choose $K{=}4$ and use the same renderings. Furthermore, α and β are adjustable temperatures of the distributions p_{3D} and p_{im}, and $q_{\boldsymbol{\theta}}$ denotes the distribution over diffused renderings defined by the underlying 3D scene representation $\boldsymbol{\theta}$, which is optimized through the differentiable rendering. We also use the Gaussian densification method discussed in Sec. 2 (see Supp. Material).

Stage 2: Adding Dynamics for 4D Synthesis (Fig. 2). While in stage 1, we only optimize the 3D Gaussians, in stage 2, the main 4D stage, we optimize (only) the deformation field $\boldsymbol{\Phi}$ to capture motion and extend the static 3D scene to a dynamic 4D scene with temporal dimension τ. To this end, we compose the text-to-image and text-to-video DMs and formally minimize a reverse KLD of the form

$$\mathrm{KL}\left(q_{\boldsymbol{\Phi}}\left(\{\mathbf{z}_{\tau_i}^{c_i}\}_F, \{\tilde{\mathbf{z}}_{\tilde{\tau}_j}^{\tilde{c}_j}\}_M\right) \middle|\middle| p_{\mathrm{vid}}^{\gamma}\left(\{\mathbf{z}_{\tau_i}^{c_i}\}_F\right) \prod_{j=1}^{M} p_{\mathrm{im}}^{\kappa}\left(\tilde{\mathbf{z}}_{\tilde{\tau}_j}^{\tilde{c}_j}\right)\right),$$

where $p_{\mathrm{vid}}(\{\mathbf{z}_{\tau_i}^{c_i}\}_F)$ is the video DM-defined distribution

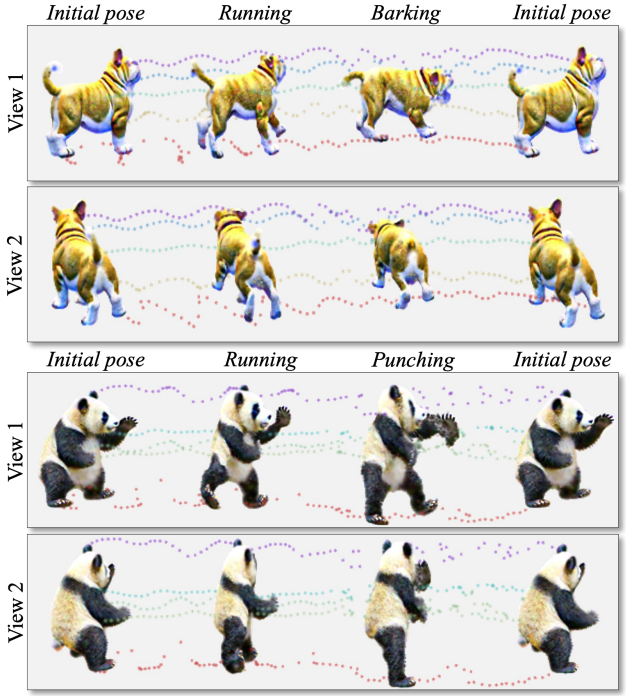

Figure 7. **Autoregressively extended text-to-4D synthesis.** AYG is able to autoregressively extend dynamic 4D sequences, combine sequences with different text-guidance, and create looping animations, returning to the initial pose (also see Supp. Video).

over F 4D scene renderings $\{\mathbf{z}_{\tau_i}^{c_i}\}_F$ taken at times τ_i and camera angles c_i (F=16 for our model). Similar to before, M additional renderings are given to the SD-based general image prior, and γ and κ are temperatures. The renderings $\{\tilde{\mathbf{z}}_{\tilde{\tau}_j}^{\tilde{c}_j}\}_M$ fed to regular SD can be taken at different times $\tilde{\tau}_j$ and cameras $\tilde{c}_j$ than the video model frames, but in practice M=4 and we use three random renderings as well as the 8th middle frame among the ones given to the video model. $q_{\mathbf{\Phi}}$ defines the distribution over renderings by the 4D scene with the learnable deformation field parameters $\mathbf{\Phi}$. We could render videos from the 4D scene with a fixed camera, but in practice dynamic cameras, *i.e.* varying c_i, help to learn more vivid 4D scenes, similar to Singer et al. [79].

Moreover, following Singer et al. [79], our video DM is conditioned on the frame rate (fps) and we choose the times $0 \leq \tau_i \leq 1$ accordingly by sampling fps $\in \{4, 8, 12\}$ and the starting time. We render videos from the 4D scene and condition the video DM with the sampled fps. This helps generating not only sufficiently long but also temporally smooth 4D animations, as different fps correspond to long-term and short-term dynamics. Therefore, when rendering short but high fps videos they only span part of the entire length of the 4D sequence. Also see Supp. Material.

Optimizing the deformation field while supervising both with a video and image DM is crucial. The video DM generates temporal dynamics, but text-to-video DMs are not as robust as general text-to-image DMs. Including the image DM during this stage ensures stable optimization and that high visual frame quality is maintained (ablations in Sec. 4).

A crucial advantage of the disentangled two stage design is that AYG's main 4D synthesis method—the main innovation of this work—could in the future in principle also be applied to 3D objects originating from other generation systems or even to synthetic assets created by digital artists.

3.3. AYG's Score Distillation in Practice

Above, we have laid out AYG's general synthesis framework. The full stage 2 score distillation gradient including CFG can be expressed as (stage 1 proceeds analogously)

$$\nabla_{\mathbf{\Phi}} \mathcal{L}_{\text{SDS}}^{\text{AYG}} = \mathbb{E}_{t,\boldsymbol{\epsilon}^{\text{vid}},\boldsymbol{\epsilon}^{\text{im}}} \Bigg[w(t) \Bigg\{ \gamma \Bigg(\omega_{\text{vid}} \big[\hat{\boldsymbol{\epsilon}}^{\text{vid}}(\mathbf{Z}, v, t) - \hat{\boldsymbol{\epsilon}}^{\text{vid}}(\mathbf{Z}, t) \big] + \underbrace{\hat{\boldsymbol{\epsilon}}^{\text{vid}}(\mathbf{Z}, v, t) - \boldsymbol{\epsilon}^{\text{vid}}}_{\boldsymbol{\delta}_{\text{gen}}^{\text{vid}}} \Bigg) + \kappa \Bigg(\omega_{\text{im}} \big[\hat{\boldsymbol{\epsilon}}^{\text{im}}(\tilde{\mathbf{Z}}, v, t) - \hat{\boldsymbol{\epsilon}}^{\text{im}}(\tilde{\mathbf{Z}}, t) \big] + \underbrace{\hat{\boldsymbol{\epsilon}}^{\text{im}}(\tilde{\mathbf{Z}}, v, t) - \boldsymbol{\epsilon}^{\text{im}}}_{\boldsymbol{\delta}_{\text{gen}}^{\text{im}}} \Bigg) \Bigg\} \frac{\partial \{\mathbf{x}\}}{\partial \mathbf{\Phi}} \Bigg], \tag{3}$$

where $\mathbf{Z} := \{\mathbf{z}_{\tau_i}^{c_i}\}_F$, $\tilde{\mathbf{Z}} := \{\tilde{\mathbf{z}}_{\tilde{\tau}_j}^{\tilde{c}_j}\}_M$, $\omega_{\text{vid/im}}$ are the CFG scales for the video and image DMs, $\hat{\boldsymbol{\epsilon}}^{\text{vid}}(\mathbf{Z}, v, t)$ and $\hat{\boldsymbol{\epsilon}}^{\text{im}}(\tilde{\mathbf{Z}}, v, t)$ are the corresponding denoiser networks and $\boldsymbol{\epsilon}^{\text{vid}}$ and $\boldsymbol{\epsilon}^{\text{im}}$ are the diffusion noises (an analogous SDS gradient can be written for stage 1). Moreover, $\{\mathbf{x}\}$ denotes the set of all renderings from the 4D scene through which the SDS gradient is backpropagated, and which are diffused to produce $\mathbf{Z}$ and $\tilde{\mathbf{Z}}$. Recently, ProlificDreamer [92] proposed a scheme where the control variates $\boldsymbol{\epsilon}^{\text{vid/im}}$ above are replaced by DMs that model the rendering distribution, are initialized from the DMs guiding the synthesis ($\hat{\boldsymbol{\epsilon}}^{\text{vid}}(\mathbf{Z}, v, t)$ and $\hat{\boldsymbol{\epsilon}}^{\text{im}}(\tilde{\mathbf{Z}}, v, t)$ here), and are then slowly fine-tuned on the diffused renderings ($\mathbf{Z}$ or $\tilde{\mathbf{Z}}$ here). This means that at the beginning of optimization the terms $\boldsymbol{\delta}_{\text{gen}}^{\text{vid/im}}$ in Eq. (3) would be zero. Inspired by this observation and aiming to avoid ProlificDreamer's cumbersome fine-tuning, we instead propose to simply set $\boldsymbol{\delta}_{\text{gen}}^{\text{vid/im}} = 0$ entirely and optimize with

$$\nabla_{\mathbf{\Phi}} \mathcal{L}_{\text{CSD}}^{\text{AYG}} = \mathbb{E}_{t,\boldsymbol{\epsilon}^{\text{vid}},\boldsymbol{\epsilon}^{\text{im}}} \Bigg[w(t) \Bigg\{ \omega_{\text{vid}} \underbrace{\big[\hat{\boldsymbol{\epsilon}}^{\text{vid}}(\mathbf{Z}, v, t) - \hat{\boldsymbol{\epsilon}}^{\text{vid}}(\mathbf{Z}, t) \big]}_{\boldsymbol{\delta}_{\text{cls}}^{\text{vid}}} + \omega_{\text{im}} \underbrace{\big[\hat{\boldsymbol{\epsilon}}^{\text{im}}(\tilde{\mathbf{Z}}, v, t) - \hat{\boldsymbol{\epsilon}}^{\text{im}}(\tilde{\mathbf{Z}}, t) \big]}_{\boldsymbol{\delta}_{\text{cls}}^{\text{im}}} \Bigg\} \frac{\partial \{\mathbf{x}\}}{\partial \mathbf{\Phi}} \Bigg], \tag{4}$$

where we absorbed γ and κ into $\omega_{\text{vid/im}}$. Interestingly, this exactly corresponds to the concurrently proposed classifier score distillation (CSD) [102], which points out that the above two terms $\boldsymbol{\delta}_{\text{cls}}^{\text{vid/im}}$ in Eq. (4) correspond to implicit classifiers predicting v from the video or images, respectively. CSD then uses only $\boldsymbol{\delta}_{\text{cls}}^{\text{vid/im}}$ for score distillation, resulting in improved performance over SDS. We discovered that scheme independently, while aiming to inherit ProlificDreamer's strong performance. Supp. Material for details.

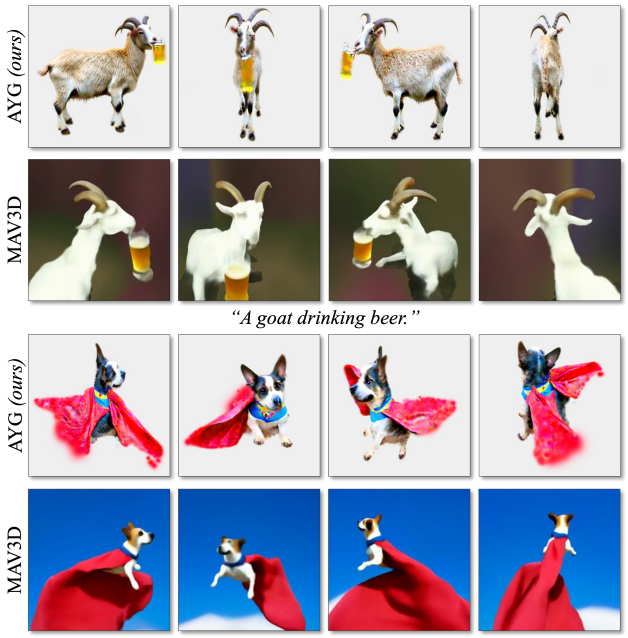

Figure 8. **AYG (*ours*) vs. MAV3D [79].** We show four 4D frames for different times and camera angles (also see Supp. Video).

3.4. Scaling Align Your Gaussians

To scale AYG and achieve state-of-the-art text-to-4D performance, we introduce several further novel techniques.

Distribution Regularization of 4D Gaussians. We developed a method to stabilize optimization and ensure realistic learnt motion. We calculate the means $\boldsymbol{\nu}_\tau$ and diagonal covariances $\boldsymbol{\Gamma}_\tau$ of the entire set of 3D Gaussians (using their means $\boldsymbol{\mu}_i$) at times τ along the 4D sequence (Fig. 4). Defining a Normal distribution $\mathcal{N}(\boldsymbol{\nu}_\tau, \boldsymbol{\Gamma}_\tau)$ with these means and covariances, we regularize with a modified version of the Jensen-Shannon divergence JSD $(\mathcal{N}(\boldsymbol{\nu}_0, \boldsymbol{\Gamma}_0)||\mathcal{N}(\boldsymbol{\nu}_\tau, \boldsymbol{\Gamma}_\tau))$ between the 3D Gaussians at the initial and later frames τ (see Supp. Material). This ensures that the mean and the diagonal covariance of the distribution of the Gaussians stay approximately constant and encourages AYG to generate meaningful and complex dynamics instead of simple global translations and object size changes.

Extended Autoregressive Generation. By default, AYG produces relatively short 4D sequences, which is due to the guiding text-to-video model, which itself only generates short video clips (see Blattmann et al. [7]). To overcome this limitation, we developed a method to autoregressively extend the 4D sequences. We use the middle 4D frame from a first sequence as the initial frame of a second sequence, optimizing a second deformation field, optionally using a different text prompt. As the second sequence is initialized from the middle frame of the first sequence, there is an overlap interval with length 0.5 of the total length of each sequence. When optimizing for the second deformation field, we smoothly interpolate between the first and second deformation fields for the overlap region (Fig. 5). Specifically, we define $\boldsymbol{\Delta}^{\text{interpol}}_{\boldsymbol{\Phi}_{12}} = (1-\chi(\tau))\boldsymbol{\Delta}_{\boldsymbol{\Phi}_1} + \chi(\tau)\boldsymbol{\Delta}_{\boldsymbol{\Phi}_2}$ where χ is a linear function with $\chi(\tau_{0.5}) = 0$ and $\chi(\tau_{1.0}) = 1$, $\tau_{0.5}$ and $\tau_{1.0}$ represent the middle and last time frames of the first sequence, $\boldsymbol{\Delta}^{\text{interpol}}_{\boldsymbol{\Phi}_{12}}$ is the interpolated deformation field, and $\boldsymbol{\Delta}_{\boldsymbol{\Phi}_1}$(kept fixed) and $\boldsymbol{\Delta}_{\boldsymbol{\Phi}_2}$ are the deformation fields of the first and second sequence, respectively. We additionally minimize $\mathcal{L}_{\text{Interpol-Reg.}} = ||\boldsymbol{\Delta}_{\boldsymbol{\Phi}_1} - \boldsymbol{\Delta}^{\text{interpol}}_{\boldsymbol{\Phi}_{12}}||^2_2$ within the overlap region to regularize the optimization process of $\boldsymbol{\Delta}_{\boldsymbol{\Phi}_2}$. For the non-overlap regions, we just use the corresponding $\boldsymbol{\Delta}_{\boldsymbol{\Phi}}$. With this careful interpolation technique the deformation field smoothly transitions from the first sequence's into the second sequence's. Without it, we obtained abrupt, unrealistic transitions.

Table 1. **Comparison to MAV3D [79]** by user study on synthesized 4D scenes with 28 text prompts. Numbers are percentages.

Method preference	AYG *(ours)* preferred	MAV3D [79] preferred	Equal preference
Overall Quality	**53.6**	38.8	7.6
3D Appearance	**47.4**	37.2	15.4
3D Text Alignment	**45.9**	38.8	15.3
Motion Amount	**45.9**	38.8	15.3
Motion Text Alignment	**47.4**	33.7	18.9
Motion Realism	**44.4**	43.9	11.7

Motion Amplification. When a set of 4D scene renderings is given to the text-to-video model, it produces a (classifier) score distillation gradient for each frame i. We expect most motion when the gradient for each frame points into a different direction. With that in mind, we propose a motion amplification technique. We post-process the video model's individual frame scores $\boldsymbol{\delta}^{\text{vid}}_{\text{cls}\,i}$ ($i \in \{1, ..., F\}$) as $\boldsymbol{\delta}^{\text{vid}}_{\text{cls}\,i} \rightarrow \langle \boldsymbol{\delta}^{\text{vid}}_{\text{cls}\,i} \rangle + \omega_{\text{ma}} \left(\boldsymbol{\delta}^{\text{vid}}_{\text{cls}\,i} - \langle \boldsymbol{\delta}^{\text{vid}}_{\text{cls}\,i} \rangle \right)$, where $\langle \boldsymbol{\delta}^{\text{vid}}_{\text{cls}\,i} \rangle$ is the average score over the F video frames and ω_{ma} is the motion amplifier scale. This scheme is inspired by CFG and reproduces regular video model scores for ω_{ma}=1. For larger ω_{ma}, the difference between the individual frames' scores and the average is amplified, thereby encouraging larger frame differences and more motion.

View Guidance. In AYG's 3D stage, for the text-to-image model we use a new *view guidance*. We construct an additional implicit classifier term $\omega_{\text{vg}} \left[\hat{\boldsymbol{\epsilon}}^{\text{im}}(\mathbf{z}, v^{\text{aug}}, t) - \hat{\boldsymbol{\epsilon}}^{\text{im}}(\mathbf{z}, v, t) \right]$, where v^{aug} denotes the original text prompt v augmented with directional texts such as "front view" (see Sec. 3.2) and ω_{vg} is the guidance scale. View guidance amplifies the effect of directional text prompt augmentation.

Negative Prompting. We also use negative prompt guidance during both the 3D and 4D stages. During the 4D stage, we use *"low motion, static statue, not moving, no motion"* to encourage AYG to generate more dynamic and vivid 4D scenes. Supp. Material for 3D stage and details.

4. Experiments

Text-to-4D. In Fig. 6, we show text-to-4D sequences generated by AYG (hyperparameters and details in Supp. Ma-

Align Your Gaussians (full model)	Overall Quality	3D Appearance	3D Text Alignment	Motion Amount	Motion Text Alignment	Motion Realism
v.s. w/o rigidity regularization	**45.8**/13.3	**43.3**/19.2	**38.3**/15.0	**40.8**/15.0	**42.5**/18.3	**30.8**/26.7
v.s. w/o motion amplifier	**43.3**/23.3	**37.5**/28.3	**30.8**/26.7	**45.8**/10.8	**37.5**/26.7	**33.3**/31.7
v.s. w/o initial 3D stage	**67.5**/15.0	**57.5**/21.7	**64.2**/15.0	**60.8**/21.7	**60.8**/20.8	**59.2**/24.2
v.s. w/o JSD-based regularization	**40.0**/25.0	**40.0**/27.5	**36.7**/27.5	**41.7**/24.2	**39.2**/29.2	**45.0**/24.2
v.s. w/o image DM score in 4D stage	**42.5**/22.5	**39.2**/27.5	**36.7**/25.8	**33.3**/25.9	**37.5**/30.0	27.5/**40.0**
v.s. SDS instead of CSD	**44.2**/35.8	**40.0**/27.5	**35.8**/35.0	**35.0**/27.5	**35.0**/34.2	32.5/**35.8**
v.s. 3D stage w/o MVDream	**66.7**/21.7	**48.3**/34.2	**38.3**/34.2	**41.7**/22.5	**40.0**/27.5	**40.8**/27.5
v.s. 4D stage with MVDream	**50.8**/27.5	**38.3**/34.2	**41.6**/29.2	**39.2**/35.0	**44.2**/30.0	**39.2**/31.7
v.s. video model with only fps 4	**46.7**/15.8	27.5/**36.7**	**30.0**/23.3	**36.7**/30.0	**31.7**/26.7	**32.5**/28.3
v.s. video model with only fps 12	**48.3**/29.2	**30.8**/29.2	**29.2**/28.3	**35.0**/28.3	**35.0**/30.0	**39.2**/26.7
v.s. w/o dynamic cameras	**32.5**/25.0	**32.5**/31.7	**35.0**/33.3	**35.0**/32.5	**35.8**/33.3	**32.5**/25.0
v.s. w/o negative prompting	**44.2**/28.3	**38.3**/32.5	**31.7**/29.2	29.2/**31.6**	**33.3**/30.0	**37.5**/28.3

Table 2. **Ablation study** by user study on synthesized 4D scenes with 30 text prompts. For each pair of numbers, the left number is the percentage that the full AYG model is preferred and the right number indicates preference percentage for ablated model as described in left column. The numbers do not add up to 100 and the difference is due to users voting "no preference" (details in Supp. Material).

terial). AYG can generate realistic, expressive, detailed and vivid dynamic 4D scenes (4D scenes can be rendered at varying speeds and frame rates). Importantly, our method demonstrates zero-shot generalization capabilities to creative text prompts corresponding to scenes that are unlikely to be found in the diffusion models' training images and videos. More results in Supp. Material and on project page.

To compare AYG to MAV3D [79], we performed a comprehensive user study where we took the 28 rendered videos from MAV3D's project page[2] and compared them to corresponding generations from AYG with the same text prompts (Table 1). We asked the users to rate overall quality, 3D appearance and text alignment, as well as motion amount, motion text alignment and motion realism (user study details in Supp. Material). AYG outperforms MAV3D on all metrics, achieving state-of-the-art text-to-4D performance (we also evaluated R-Precision [32, 58] on a larger prompt set used by MAV3D [78, 79], performing on par, see Supp. Mat.; however, R-Precision is a meaningless metric to evaluate *dynamic* scenes). Qualitative comparisons are shown in Fig. 8 (more in Supp. Mat.). We see that AYG produces more detailed 4D outputs. Note that MAV3D uses an extra background model, while AYG does not. Adding a similar background model would be easy but is left to future work.

Ablation Studies. Next, we performed an ablation study on AYG's different components. We used a set of 30 text prompts and generated 4D scenes for versions of AYG with missing or modified components, see Table 2. Using the same categories as before, we asked users to rate preference of our full method vs. the ablated AYG variants. Some components have different effects with respect to 3D appearance and motion, but we generally see that all components matter significantly in terms of overall quality, *i.e.*, for all ablations our full method is strongly preferred over the ablated AYG versions. This justifies AYG's design. A thorough discussion is presented in the Supp. Material, but we highlight some relevant observations. We see that our novel JSD-based regularization makes a major difference, and we also observe that the motion amplifier indeed has a strong effect for "Motion Amount". Moreover, our compositional approach is crucial. Running the 4D stage without image DM feedback produces much worse 3D and overall quality. Also the decomposition into two stages is important—carrying out 4D synthesis without initial 3D stage performs poorly.

Temporally Extended 4D Synthesis and Large Scene Composition. In Fig. 7, we show autoregressively extended text-to-4D results with changing text prompts (also see Supp. Video). AYG can realistically connect different 4D sequences and generate expressive animations with changing dynamics and behavior. We can also create sequences that loop endlessly by enforcing that the last frame of a later sequence matches the first frame of an earlier one and suppressing the deformation field there (similar to how we enforce zero deformation at τ=0 in Sec. 3.1). Finally, due to the explicit nature of the dynamic 3D Gaussians, AYG's 4D representation, multiple animated 4D objects can be easily composed into larger scenes, each shape with its own deformation field defining its dynamics. We show this in Fig. 1, where each dynamic object in the large scene is generated, except for the ground plane. These capabilities, not shown by previous work [79], are particularly promising for practical content creation applications.

5. Conclusions

We presented *Align Your Gaussians* for expressive text-to-4D synthesis. AYG builds on dynamic 3D Gaussian Splatting with deformation fields as well as score distillation with multiple composed diffusion models. Novel regularization and guidance techniques allow us to achieve state-of-the-art dynamic scene generation and we also show temporally extended 4D synthesis as well as the composition of multiple dynamic objects within a larger scene. AYG has many potential applications for creative content creation and it could also be used in the context of synthetic data generation. For example, AYG would enable synthesis of videos and 4D sequences with exact tracking labels, useful for training discriminative models. AYG currently cannot easily produce topological changes of the dynamic objects. Overcoming this limitation would be an exciting avenue for future work. Other directions include scaling AYG beyond object-centric generation and personalized 4D synthesis. The initial 3D object could be generated from a personalized diffusion model (*e.g.* DreamBooth3D [66, 71]) or with image-to-3D methods [29, 42, 44, 45, 64] and then animated with AYG.

[2] https://make-a-video3d.github.io/

References

[1] Jie An, Songyang Zhang, Harry Yang, Sonal Gupta, Jia-Bin Huang, Jiebo Luo, and Xi Yin. Latent-Shift: Latent Diffusion with Temporal Shift for Efficient Text-to-Video Generation. *arXiv preprint arXiv:2304.08477*, 2023. 2, 3, 15

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer Normalization. *arXiv preprint arXiv:1607.06450*, 2016. 17

[3] 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. 16

[4] Max Bain, Arsha Nagrani, Gül Varol, and Andrew Zisserman. Frozen in Time: A Joint Video and Image Encoder for End-to-End Retrieval. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 4, 24

[5] Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Qinsheng Zhang, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, Bryan Catanzaro, Tero Karras, and Ming-Yu Liu. eDiff-I: Text-to-Image Diffusion Models with Ensemble of Expert Denoisers. *arXiv preprint arXiv:2211.01324*, 2022. 3, 15

[6] Miguel Ángel Bautista, Pengsheng Guo, Samira Abnar, Walter Talbott, Alexander T Toshev, Zhuoyuan Chen, Laurent Dinh, Shuangfei Zhai, Hanlin Goh, Daniel Ulbricht, Afshin Dehghan, and Joshua M. Susskind. GAUDI: A Neural Architect for Immersive 3D Scene Generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 15

[7] Andreas Blattmann, Robin Rombach, Huan Ling, Tim Dockhorn, Seung Wook Kim, Sanja Fidler, and Karsten Kreis. Align your Latents: High-Resolution Video Synthesis with Latent Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2, 3, 4, 7, 15, 19, 24, 25

[8] Hongrui Cai, Wanquan Feng, Xuetao Feng, Yan Wang, and Juyong Zhang. Neural Surface Reconstruction of Dynamic Scenes with Monocular RGB-D Camera. In *Thirty-sixth Conference on Neural Information Processing Systems (NeurIPS)*, 2022. 3, 16

[9] 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 (CVPR)*, 2023. 3, 15

[10] Rui Chen, Yongwei Chen, Ningxin Jiao, and Kui Jia. Fantasia3D: Disentangling Geometry and Appearance for High-quality Text-to-3D Content Creation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 2, 3, 15

[11] Yiwen Chen, Chi Zhang, Xiaofeng Yang, Zhongang Cai, Gang Yu, Lei Yang, and Guosheng Lin. IT3D: Improved Text-to-3D Generation with Explicit View Synthesis. *arXiv preprint arXiv:2308.11473*, 2023. 2, 3, 15

[12] Zilong Chen, Feng Wang, and Huaping Liu. Text-to-3D using Gaussian Splatting. *arXiv preprint arXiv:2309.16585*, 2023. 3, 16

[13] Jaeyoung Chung, Suyoung Lee, Hyeongjin Nam, Jaerin Lee, and Kyoung Mu Lee. LucidDreamer: Domain-free Generation of 3D Gaussian Splatting Scenes. *arXiv preprint arXiv:2311.13384*, 2023. 16

[14] Xiaoliang Dai, Ji Hou, Chih-Yao Ma, Sam Tsai, Jialiang Wang, Rui Wang, Peizhao Zhang, Simon Vandenhende, Xiaofang Wang, Abhimanyu Dubey, Matthew Yu, Abhishek Kadian, Filip Radenovic, Dhruv Mahajan, Kunpeng Li, Yue Zhao, Vladan Petrovic, Mitesh Kumar Singh, Simran Motwani, Yi Wen, Yiwen Song, Roshan Sumbaly, Vignesh Ramanathan, Zijian He, Peter Vajda, and Devi Parikh. Emu: Enhancing Image Generation Models Using Photogenic Needles in a Haystack. *arXiv preprint arXiv:2309.15807*, 2023. 3, 15

[15] 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. *arXiv preprint arXiv:2307.05663*, 2023. 4

[16] 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. 4

[17] C. Deng, C. Jiang, C. R. Qi, X. Yan, Y. Zhou, L. Guibas, and D. Anguelov. NeRDi: Single-View NeRF Synthesis with Language-Guided Diffusion as General Image Priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 3, 15

[18] Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion Models Beat GANs on Image Synthesis. In *Advances in Neural Information Processing Systems*, 2021. 3, 15

[19] Yilun Du, Conor Durkan, Robin Strudel, Joshua B. Tenenbaum, Sander Dieleman, Rob Fergus, Jascha Sohl-Dickstein, Arnaud Doucet, and Will Grathwohl. Reduce, Reuse, Recycle: Compositional Generation with Energy-Based Diffusion Models and MCMC. In *Proceedings of the 40th International Conference on Machine Learning*, 2023. 3, 16, 20

[20] Lincong Feng, Muyu Wang, Maoyu Wang, Kuo Xu, and Xiaoli Liu. MetaDreamer: Efficient Text-to-3D Creation With Disentangling Geometry and Texture. *arXiv preprint arXiv:2311.10123*, 2023. 15

[21] Zhida Feng, Zhenyu Zhang, Xintong Yu, Yewei Fang, Lanxin Li, Xuyi Chen, Yuxiang Lu, Jiaxiang Liu, Weichong Yin, Shikun Feng, Yu Sun, Li Chen, Hao Tian, Hua Wu, and Haifeng Wang. ERNIE-ViLG 2.0: Improving Text-to-Image Diffusion Model With Knowledge-Enhanced Mixture-of-Denoising-Experts. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 3, 15

[22] Songwei Ge, Seungjun Nah, Guilin Liu, Tyler Poon, Andrew Tao, Bryan Catanzaro, David Jacobs, Jia-Bin Huang, Ming-Yu Liu, and Yogesh Balaji. Preserve Your Own Correlation: A Noise Prior for Video Diffusion Models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 2, 3, 15

[23] Rohit Girdhar, Mannat Singh, Andrew Brown, Quentin Duval, Samaneh Azadi, Sai Saketh Rambhatla, Akbar Shah, Xi Yin, Devi Parikh, and Ishan Misra. Emu Video: Factorizing Text-to-Video Generation by Explicit Image Conditioning. *arXiv preprint arXiv:2311.10709*, 2023. 15

[24] Jiatao Gu, Shuangfei Zhai, Yizhe Zhang, Josh Susskind, and Navdeep Jaitly. Matryoshka Diffusion Models. *arXiv preprint arXiv:2310.15111*, 2023. 15

[25] Yuwei Guo, Ceyuan Yang, Anyi Rao, Yaohui Wang, Yu Qiao, Dahua Lin, and Bo Dai. AnimateDiff: Animate Your Personalized Text-to-Image Diffusion Models without Specific Tuning. *arXiv preprint arXiv:2307.04725*, 2023. 3, 15

[26] Jonathan Ho and Tim Salimans. Classifier-Free Diffusion Guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. 3, 23

[27] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising Diffusion Probabilistic Models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. 3, 15, 21, 22

[28] Jonathan Ho, William Chan, Chitwan Saharia, Jay Whang, Ruiqi Gao, Alexey Gritsenko, Diederik P. Kingma, Ben Poole, Mohammad Norouzi, David J. Fleet, and Tim Salimans. Imagen Video: High Definition Video Generation with Diffusion Models. *arXiv preprint arXiv:2210.02303*, 2022. 2, 3, 15

[29] Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. LRM: Large Reconstruction Model for Single Image to 3D. *arXiv preprint arXiv:2311.04400*, 2023. 8

[30] Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. Simple Diffusion: End-to-End Diffusion for High Resolution Images. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023. 15

[31] Yukun Huang, Jianan Wang, Yukai Shi, Xianbiao Qi, Zheng-Jun Zha, and Lei Zhang. DreamTime: An Improved Optimization Strategy for Text-to-3D Content Creation. *arXiv preprint arXiv:2306.12422*, 2023. 2, 3, 15

[32] A. Jain, B. Mildenhall, J. T. Barron, P. Abbeel, and B. Poole. Zero-Shot Text-Guided Object Generation with Dream Fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 8, 28

[33] Yanqin Jiang, Li Zhang, Jin Gao, Weimin Hu, and Yao Yao. Consistent4D: Consistent 360° Dynamic Object Generation from Monocular Video. *arXiv preprint arxiv:2311.02848*, 2023. 16

[34] Nikolai Kalischek, Torben Peters, Jan D. Wegner, and Konrad Schindler. Tetrahedral Diffusion Models for 3D Shape Generation. *arXiv preprint arXiv:2211.13220*, 2022. 15

[35] Oren Katzir, Or Patashnik, Daniel Cohen-Or, and Dani Lischinski. Noise-Free Score Distillation. *arXiv preprint arXiv:2310.17590*, 2023. 15

[36] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for Real-Time Radiance Field Rendering. *ACM Transactions on Graphics*, 42(4), 2023. 2, 3, 4, 16, 17

[37] Isaac Kerlow. *The Art of 3D Computer Animation and Effects*. Wiley Publishing, 4th edition, 2009. 16

[38] Levon Khachatryan, Andranik Movsisyan, Vahram Tadevosyan, Roberto Henschel, Zhangyang Wang, Shant Navasardyan, and Humphrey Shi. Text2Video-Zero: Text-to-Image Diffusion Models are Zero-Shot Video Generators. *arXiv preprint arXiv:2303.13439*, 2023. 3, 15, 25

[39] Seung Wook Kim, Bradley Brown, Kangxue Yin, Karsten Kreis, Katja Schwarz, Daiqing Li, Robin Rombach, Antonio Torralba, and Sanja Fidler. NeuralField-LDM: Scene Generation with Hierarchical Latent Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 15

[40] 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. 15, 16

[41] 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 (CVPR)*, 2023. 2, 3, 15

[42] Minghua Liu, Ruoxi Shi, Linghao Chen, Zhuoyang Zhang, Chao Xu, Xinyue Wei, Hansheng Chen, Chong Zeng, Jiayuan Gu, and Hao Su. One-2-3-45++: Fast Single Image to 3D Objects with Consistent Multi-View Generation and 3D Diffusion. *arXiv preprint arXiv:2311.07885*, 2023. 3, 8, 15

[43] Nan Liu, Shuang Li, Yilun Du, Antonio Torralba, and Joshua B. Tenenbaum. Compositional Visual Generation with Composable Diffusion Models. In *Computer Vision – ECCV 2022*, 2022. 3, 16, 20

[44] 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 (ICCV)*, 2023. 3, 8, 15

[45] 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. 3, 8, 15

[46] Zhen Liu, Yao Feng, Michael J. Black, Derek Nowrouzezahrai, Liam Paull, and Weiyang Liu. MeshDiffusion: Score-based Generative 3D Mesh Modeling. In *International Conference on Learning Representations (ICLR)*, 2023. 15

[47] Xiaoxiao Long, Yuan-Chen Guo, Cheng Lin, Yuan Liu, Zhiyang Dou, Lingjie Liu, Yuexin Ma, Song-Hai Zhang, Marc Habermann, Christian Theobalt, and Wenping Wang. Wonder3D: Single Image to 3D using Cross-Domain Diffusion. *arXiv preprint arXiv:2310.15008*, 2023. 15

[48] Jonathan Lorraine, Kevin Xie, Xiaohui Zeng, Chen-Hsuan Lin, Towaki Takikawa, Nicholas Sharp, Tsung-Yi Lin, Ming-Yu Liu, Sanja Fidler, and James Lucas. ATT3D: Amortized Text-to-3D Object Synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 3, 15

[49] Bruce D Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI'81: 7th international joint conference on Artificial intelligence*, 1981. 24

[50] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3D Gaussians: Tracking by Persistent Dynamic View Synthesis. *arXiv preprint arXiv:2308.09713*, 2023. 3, 4, 16, 17

[51] Shitong Luo and Wei Hu. Diffusion Probabilistic Models for 3D Point Cloud Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 15

[52] G. Metzer, E. Richardson, O. Patashnik, R. Giryes, and D. Cohen-Or. Latent-NeRF for Shape-Guided Generation of 3D Shapes and Textures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2, 3, 15

[53] Marko Mihajlovic, Sergey Prokudin, Marc Pollefeys, and Siyu Tang. ResFields: Residual Neural Fields for Spatiotemporal Signals. *arXiv preprint arXiv:2309.03160*, 2023. 16

[54] 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 *ECCV*, 2020. 3, 15

[55] Vinod Nair and Geoffrey E Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010. 17

[56] 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. 3, 15

[57] Alexander Quinn Nichol and Prafulla Dhariwal. Improved Denoising Diffusion Probabilistic Models. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021. 3, 15

[58] Dong Huk Park, Samaneh Azadi, Xihui Liu, Trevor Darrell, and Anna Rohrbach. Benchmark for Compositional Text-to-Image Synthesis. In *NeurIPS Datasets and Benchmarks*, 2021. 8, 28

[59] 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 (ICCV)*, 2021. 2, 3, 4, 16

[60] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. HyperNeRF: A Higher-Dimensional Representation for Topologically Varying Neural Radiance Fields. *ACM Trans. Graph.*, 40(6), 2021. 3, 16

[61] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. SDXL: Improving Latent Diffusion Models for High-Resolution Image Synthesis. *arXiv preprint arXiv:2307.01952*, 2023. 3, 15

[62] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. DreamFusion: Text-to-3D using 2D Diffusion. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023. 2, 3, 5, 15, 16, 21, 22

[63] 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 (CVPR)*, 2020. 2, 3, 4, 16

[64] Guocheng Qian, Jinjie Mai, Abdullah Hamdi, Jian Ren, Aliaksandr Siarohin, Bing Li, Hsin-Ying Lee, Ivan Skorokhodov, Peter Wonka, Sergey Tulyakov, and Bernard Ghanem. Magic123: One Image to High-Quality 3D Object Generation Using Both 2D and 3D Diffusion Priors. *arXiv preprint arXiv:2306.17843*, 2023. 3, 8, 15

[65] 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 *Proceedings of the 38th International Conference on Machine Learning (ICML)*, 2021. 28

[66] Amit Raj, Srinivas Kaza, Ben Poole, Michael Niemeyer, Ben Mildenhall, Nataniel Ruiz, Shiran Zada, Kfir Aberman, Michael Rubenstein, Jonathan Barron, Yuanzhen Li, and Varun Jampani. DreamBooth3D: Subject-Driven Text-to-3D Generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 8

[67] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical Text-Conditional Image Generation with CLIP Latents. *arXiv preprint arXiv:2204.06125*, 2022. 3, 15

[68] Davis Rempe, Zhengyi Luo, Xue Bin Peng, Ye Yuan, Kris Kitani, Karsten Kreis, Sanja Fidler, and Or Litany. Trace and Pace: Controllable Pedestrian Animation via Guided Trajectory Diffusion. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 16

[69] Geoffrey Roeder, Yuhuai Wu, and David Duvenaud. Sticking the Landing: Simple, Lower-Variance Gradient Estimators for Variational Inference. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. 22

[70] 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 (CVPR)*, 2022. 2, 3, 4, 15, 19, 20, 24, 29

[71] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 8

[72] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed

Ghasemipour, Raphael Gontijo-Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 3, 15

[73] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. LAION-5B: An open large-scale dataset for training next generation image-text models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 24

[74] Katja Schwarz, Seung Wook Kim, Jun Gao, Sanja Fidler, Andreas Geiger, and Karsten Kreis. WildFusion: Learning 3D-Aware Latent Diffusion Models in View Space. *arXiv preprint arXiv:2311.13570*, 2023. 15

[75] 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. 3, 15

[76] Yichun Shi, Peng Wang, Jianglong Ye, Mai Long, Kejie Li, and Xiao Yang. MVDream: Multi-view Diffusion for 3D Generation. *arXiv preprint arXiv:2308.16512*, 2023. 2, 3, 4, 15, 19, 20, 24, 25, 29

[77] J. Shue, E. Chan, R. Po, Z. Ankner, J. Wu, and G. Wetzstein. 3D Neural Field Generation Using Triplane Diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 15

[78] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, Devi Parikh, Sonal Gupta, and Yaniv Taigman. Make-A-Video: Text-to-Video Generation without Text-Video Data. In *The Eleventh International Conference on Learning Representations (ICLR)*, 2023. 2, 3, 8, 15, 16, 24, 27, 28

[79] Uriel Singer, Shelly Sheynin, Adam Polyak, Oron Ashual, Iurii Makarov, Filippos Kokkinos, Naman Goyal, Andrea Vedaldi, Devi Parikh, Justin Johnson, and Yaniv Taigman. Text-to-4D Dynamic Scene Generation. In *Proceedings of the 40th International Conference on Machine Learning*, 2023. 2, 3, 4, 6, 7, 8, 15, 17, 20, 24, 26, 27, 30, 31, 36

[80] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep Unsupervised Learning using Nonequilibrium Thermodynamics. In *International Conference on Machine Learning (ICML)*, 2015. 3, 15

[81] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-Based Generative Modeling through Stochastic Differential Equations. In *International Conference on Learning Representations (ICLR)*, 2021. 3, 15, 21, 22

[82] 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. 15

[83] 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. 3, 16, 25

[84] 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 (ICCV)*, 2021. 3, 16

[85] Christina Tsalicoglou, Fabian Manhardt, Alessio Tonioni, Michael Niemeyer, and Federico Tombari. TextMesh: Generation of Realistic 3D Meshes From Text Prompts. In *International conference on 3D vision (3DV)*, 2024. 2, 3, 15

[86] Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based Generative Modeling in Latent Space. In *Neural Information Processing Systems (NeurIPS)*, 2021. 4, 15, 22, 24

[87] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *Advances in neural information processing systems*, 30, 2017. 17

[88] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A. Yeh, and Greg Shakhnarovich. Score Jacobian Chaining: Lifting Pretrained 2D Diffusion Models for 3D Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 2, 3, 15

[89] Tengfei Wang, Bo Zhang, Ting Zhang, Shuyang Gu, Jianmin Bao, Tadas Baltrusaitis, Jingjing Shen, Dong Chen, Fang Wen, Qifeng Chen, and Baining Guo. RODIN: A Generative Model for Sculpting 3D Digital Avatars Using Diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 15

[90] Wenjing Wang, Huan Yang, Zixi Tuo, Huiguo He, Junchen Zhu, Jianlong Fu, and Jiaying Liu. VideoFactory: Swap Attention in Spatiotemporal Diffusions for Text-to-Video Generation. *arXiv preprint arXiv:2305.10874*, 2023. 2, 3, 4, 15, 24

[91] Yaohui Wang, Xinyuan Chen, Xin Ma, Shangchen Zhou, Ziqi Huang, Yi Wang, Ceyuan Yang, Yinan He, Jiashuo Yu, Peiqing Yang, Yuwei Guo, Tianxing Wu, Chenyang Si, Yuming Jiang, Cunjian Chen, Chen Change Loy, Bo Dai, Dahua Lin, Yu Qiao, and Ziwei Liu. LAVIE: High-Quality Video Generation with Cascaded Latent Diffusion Models. *arXiv preprint arXiv:2309.15103*, 2023. 2, 3, 15

[92] 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. In *Thirty-seventh Conference on Neural Information Processing Systems (NeurIPS)*, 2023. 2, 3, 6, 15, 23

[93] 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. 3, 16

[94] Jay Zhangjie Wu, Yixiao Ge, Xintao Wang, Stan Weixian Lei, Yuchao Gu, Yufei Shi, Wynne Hsu, Ying Shan, Xiaohu

Qie, and Mike Zheng Shou. Tune-a-video: One-shot tuning of image diffusion models for text-to-video generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 3, 15, 25

[95] Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. PhysGaussian: Physics-Integrated 3D Gaussians for Generative Dynamics. *arXiv preprint arXiv:2311.12198*, 2023. 16

[96] Dejia Xu, Yifan Jiang, Peihao Wang, Zhiwen Fan, Yi Wang, and Zhangyang Wang. NeuralLift-360: Lifting An In-the-wild 2D Photo to A 3D Object with 360° Views. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 3, 15

[97] Yinghao Xu, Hao Tan, Fujun Luan, Sai Bi, Peng Wang, Jiahao Li, Zifan Shi, Kalyan Sunkavalli, Gordon Wetzstein, Zexiang Xu, and Kai Zhang. DMV3D: Denoising Multi-View Diffusion using 3D Large Reconstruction Model. *arXiv preprint arXiv:2311.09217*, 2023. 15

[98] Zeyue Xue, Guanglu Song, Qiushan Guo, Boxiao Liu, Zhuofan Zong, Yu Liu, and Ping Luo. RAPHAEL: Text-to-Image Generation via Large Mixture of Diffusion Paths. *arXiv preprint arXiv:2305.18295*, 2023. 3, 15

[99] Gengshan Yang, Minh Vo, Natalia Neverova, Deva Ramanan, Andrea Vedaldi, and Hanbyul Joo. BANMo: Building Animatable 3D Neural Models from Many Casual Videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 16

[100] 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. 16

[101] 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. 3, 16

[102] Xin Yu, Yuan-Chen Guo, Yangguang Li, Ding Liang, Song-Hai Zhang, and Xiaojuan Qi. Text-to-3D with Classifier Score Distillation. *arXiv preprint arXiv:2310.19415*, 2023. 2, 4, 6, 16, 23, 24

[103] Ye Yuan, Jiaming Song, Umar Iqbal, Arash Vahdat, and Jan Kautz. PhysDiff: Physics-Guided Human Motion Diffusion Model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 16

[104] Xiaohui Zeng, Arash Vahdat, Francis Williams, Zan Gojcic, Or Litany, Sanja Fidler, and Karsten Kreis. LION: Latent Point Diffusion Models for 3D Shape Generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. 3, 15

[105] 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. 16

[106] Yufeng Zheng, Xueting Li, Koki Nagano, Sifei Liu, Karsten Kreis, Otmar Hilliges, and Shalini De Mello. A Unified Approach for Text- and Image-guided 4D Scene Generation. *arXiv preprint arXiv:2311.16854*, 2023. 16

[107] Daquan Zhou, Weimin Wang, Hanshu Yan, Weiwei Lv, Yizhe Zhu, and Jiashi Feng. MagicVideo: Efficient Video Generation With Latent Diffusion Models. *arXiv preprint arXiv:2211.11018*, 2023. 2, 3, 15

[108] Linqi Zhou, Yilun Du, and Jiajun Wu. 3D Shape Generation and Completion Through Point-Voxel Diffusion. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 15

[109] Junzhe Zhu and Peiye Zhuang. HiFA: High-fidelity Text-to-3D with Advanced Diffusion Guidance. *arXiv preprint arXiv:2305.18766*, 2023. 2, 3, 15

[110] Wojciech Zielonka, Timur Bagautdinov, Shunsuke Saito, Michael Zollhöfer, Justus Thies, and Javier Romero. Drivable 3D Gaussian Avatars. *arXiv preprint arxiv:2311.08581*, 2023. 3, 16

[111] M. Zwicker, H. Pfister, J. van Baar, and M. Gross. EWA volume splatting. In *Proceedings Visualization, 2001. VIS '01.*, 2001. 3

Contents

A. Supplementary Videos

For fully rendered videos, we primarily refer the reader to our project page, https://research.nvidia.com/labs/toronto-ai/AlignYourGaussians/, which shows all our results in best quality.

Moreover, we include 3 videos in the following google drive folder: https://drive.google.com/drive/folders/1I7e6aj-7BBrIVdePyHEQDGUxLEEls3-e:

- `ayg_text_to_4d.mp4`: This video contains many text-to-4D generation results, comparisons to MAV3D [79] and autoregressively extended and looping 4D sequences. Moreover, we show how we compose different 4D sequences into a large scene as in Fig. 1.
- `ayg_ablation_study.mp4`: This video contains different generated text-to-4D sequences for our ablation study, comparing our main Align Your Gaussians model with the different modified versions in the ablations. Note that these ablation results only use 4,000 optimization steps in the second dynamic 4D optimization stage for efficiency (see Appendix F.2).
- `ayg_new_video_model.mp4`: This video shows generated 2D video samples for different fps conditionings from our newly trained latent video diffusion model for this project.

Note that all videos shown on the project page leverage an additional fine-tuning stage with additional optimization steps compared to the results shown in the paper and in `ayg_text_to_4d.mp4`. See Appendix C.9 for a discussion on additional fine-tuning and optimization.

B. Related Work—Extended Version

Diffusion Models. Align Your Gaussians leverages multiple different diffusion models (DMs) [27, 80, 81]. DMs have revolutionized deep generative modeling in the visual domain, most prominently for image synthesis [18, 30, 57, 70, 74]. They leverage a forward diffusion process that gradually perturbs input data towards entirely random noise, while a denoiser neural network is learnt to reconstruct the input. Specifically, AYG builds on large-scale text-to-image [5, 14, 21, 24, 61, 67, 70, 72, 98], text-to-video [1, 7, 22, 23, 25, 28, 38, 78, 90, 91, 94, 107] and text-to-multiview-image DMs [42, 44, 45, 47, 64, 75, 76, 97]. A popular framework for efficient yet expressive generative modeling with DMs is the *latent* DM approach, where data is mapped into a compressed latent space with an autoencoder and the DM is trained in a more efficient manner in this latent space [70, 86]. The most prominent model of this type is Stable Diffusion [70], and AYG uses exclusively latent DMs which are based on Stable Diffusion. Specifically, apart from Stable Diffusion 2.1, we retrain a VideoLDM [7] for fps-conditioned text-to-video synthesis, which starts from a Stable Diffusion image generator as base model. Moreover, AYG uses MVDream [76], a text-guided multiview latent DM similarly fine-tuned from Stable Diffusion.

Text-to-3D Generation with Score Distillation. Text-conditioned image DMs are often trained on large-scale datasets consisting of hundreds of millions or billions of text-image pairs. However, such huge text-annotated datasets are not available for 3D data. While there exists a rich literature on 3D DMs trained on small explicit 3D or multiview image datasets [6, 34, 39, 46, 51, 56, 77, 89, 104, 108], the lack of large text-annotated 3D training datasets is a challenge for 3D generative modeling. Aiming to overcome these limitations, score distillation methods, introduced in the seminal work by Poole et al. [62], use large-scale text-guided 2D diffusion models to distill 3D objects in a per-instance optimization process. A 3D scene, parametrized by learnable parameters, is rendered from different camera perspectives and the renderings are given to a 2D diffusion model, which can provide gradients to make the renderings look more realistic. These gradients can be backpropagated through the rendering process and used to update the 3D scene representation. This has now become a flourishing research direction for 3D generative modeling [10, 11, 17, 20, 31, 35, 40, 41, 48, 52, 82, 85, 88, 92, 96, 109]. AYG also builds on the score distillation framework.

Text-to-4D Generation. The vast majority of papers on score distillation tackles static 3D object synthesis. To the best of our knowledge, there is only one previous paper that leverages score distillation for text-guided generation of dynamic 4D scenes, *Make-A-Video3D (MAV3D)* [79]. Hence, this is the most related work to AYG. However, MAV3D uses neural radiance fields [54] with HexPlane [9] features as 4D representation, in contrast to AYG's dynamic 3D Gaussians, and it does not disentangle its 4D representation into a static 3D representation and a deformation field modeling dynamics. MAV3D's representation prevents it from composing multiple 4D objects into large dynamic scenes, which our 3D Gaussian plus deformation field representation easily enables, as we show. Moreover, MAV3D's sequences are limited in time, while we show a novel autoregressive generation scheme to extend our 4D sequences. AYG outperforms MAV3D qualitatively and quantitatively and synthesizes significantly higher-quality 4D scenes. Our novel compositional generation-based approach contributes to this, which MAV3D does not pursue. More specifically, in contrast to MAV3D, our AYG shows how to simultaneously leverage image and video diffusion models for improved synthesis in the 4D generation stage, and moreover

leverages a 3D-aware multiview image diffusion model for improved 3D generation in the initial stage. Finally, instead of regular score distillation sampling [62], used by MAV3D, in practice AYG employs classifier score distillation [102]. Note that MAV3D did not release any code or model checkpoints and its 4D score distillation leverages the large-scale Make-A-Video [78] text-to-video diffusion model, which also is not available publicly.

3D Gaussian Splatting and Deformation Fields. AYG leverages a 3D Gaussian Splatting-based 4D representation with deformation fields to model dynamics. 3D Gaussian Splatting [36] has been introduced as an efficient 3D scene representation and, concurrently with our work, has also been employed for text-to-3D generation by DreamGaussian [83], GSGEN [12], GaussianDreamer [101], LucidDreamer [40] and another work also called LucidDreamer [13]. However, these works synthesize only static 3D scenes, but do not consider dynamics. Deformation fields have been widely used for dynamic 3D scene reconstruction [8, 33, 53, 59, 60, 63, 84]. Concurrently with our work, also several papers on dynamic 3D Gaussian Splatting emerged [50, 93, 95, 100, 110], similarly tackling the dynamic 3D scene reconstruction task. However, none of these works address generative modeling.

Compositional Generation. A crucial part of AYG is that it combines multiple diffusion models when performing score distillation. Specifically, in the 4D stage we combine a text-to-image with a text-to-video diffusion model in the form of a product distribution between the two and we then leverage the gradients of this product distribution for score distillation. This can be viewed as a form of compositional generation. Usually, in compositional generation, different diffusion models, or the same diffusion model with different text-conditionings, are combined such that multiple conditions are fulfilled during generation. Formally, this can be achieved by forming the product distribution of different models, see, for instance, Liu et al. [43] and Du et al. [19]. This has been used for controllable image generation. Analogously, we perform a form of compositional generation by composing an image and a video diffusion model to generate dynamic 4D assets whose renderings obey both the text-to-image and text-to-video model distributions simultaneously. Instead of aiming for controllability our goal is to simultaneously enforce smooth temporal dynamics (video model) and high individual frame quality (image model).

Animation and Motion Synthesis. AYG is also broadly related to the rich literature on character animation and motion synthesis with deep generative models; see, for example, the recent works by Rempe et al. [68] and Yuan et al. [103]. However, this line of work usually only considers the generation of joint configurations of human skeletons or similar low-dimensional simplified representations. An interesting direction for future work would be to combine these types of models with a method like our AYG. For instance, one could first synthesize a rough motion trajectory or motion path, and AYG could then generate a detailed character animation following this trajectory. There is also work on reconstructing animatable 3D assets directly from videos [99]. It would be interesting to extend AYG towards extracting assets from the synthesized 4D scenes which are simulation-ready and fully animatable using standard graphics software. Finally, AYG is also related to the broader literature on computer animation using classical methods without deep learning; see, for instance, Kerlow [37].

Further Concurrent Work. Concurrently with us, Dream-in-4D [106] also developed a new text-to-4D synthesis scheme. In contrast to AYG, the work leverages a NeRF-based 4D representation and focuses on image-guided and personalized generation, instead of using dynamic 3D Gaussians and targeting temporally extended generation as well as the possibility to easily compose different 4D assets in large scenes (one of the goals of AYG). Hence, their direction is complementary to ours, and it would be interesting to also extend AYG to personalized and image-guided 4D generation. The image-to-4D task is also tackled by Animate124 [105], which similarly leverages a dynamic NeRF-based representation. Moreover, 4D-fy [3] also addresses text-to-4D synthesis, combining supervision signals from image, video and 3D-aware diffusion models. However, like Dream-in-4D and Animate124, the work leverages a NeRF-based representation with a multi-resolution feature grid in contrast to AYG's dynamic 3D Gaussians. Moreover, 4D-fy does not explicitly disentangle shape and dynamics into a 3D component and a deformation field. Note that, in contrast to AYG, none of the mentioned works demonstrated temporally extended generation, the possibility to change the text prompt during synthesis or the ability to easily compose multiple 4D assets in large scenes.

C. Details of Align Your Gaussians' 4D Representation and Optimization

C.1. 3D Representation

We initialize each scene with 1,000 3D Gaussians spread across a sphere with radius 0.3 located on the origin $(0, 0, 0)$. Each Gaussian is initialized with a random RGB color and spherical harmonics coefficients of 0. As we use isotropic covariances, we only make use of a single scaling parameter per Gaussian, such that Gaussians are spherical. We follow Kerbl et al. [36] to initialize the scaling and opacity parameters. Similar to Kerbl et al. [36], we add and delete Gaussians during the initial 3D optimization steps to densify regions requiring more detail and prune Gaussians that are not used for rendering. Gaussians

with an average magnitude of position gradients above 0.002 are densified, and Gaussians with opacity less than 0.005 are pruned every 1,000 steps starting from the 500-th step. Additionally, we limit the number of Gaussians such that we only perform the densification step if the total number of Gaussians is less than 50,000, and we also reset the opacities to have the maximum value of 0.01 (after sigmoid) every 3,000 steps.

C.2. Deformation Field

After the 3D Gaussians are optimized into a 3D scene in the first stage, the scene dynamics are modeled by a deformation field $\mathbf{\Delta_\Phi}(x, y, z, \tau) = (\Delta x, \Delta y, \Delta z)$, defined through a multi-layer perceptron (MLP) with parameters $\mathbf{\Phi}$. We encode (x, y, z, τ) with the sinusoidal positional encoding [87]. We use 4 frequencies (each with sine and cosine components), resulting in a 32-dimensional input. The MLP with parameters $\mathbf{\Phi}$ consists of linear layers with ReLU [55] activation functions. We include Layer Normalization [2] before the ReLU every second layer, as we found it helped stabilize optimization. The last layer of $\mathbf{\Phi}$ contains a linear layer that is initialized with zero weights, followed by a soft clamping layer ($f(x) = \tanh(x/0.5) * 0.5$) so that it produces the 3-dimensional output $(\Delta x, \Delta y, \Delta z)$ clamped between $(-0.5, 0.5)$. Furthermore, we preserve the initial 3D Gaussians for the first frame, *i.e.* $\mathbf{\Delta_\Phi}(x, y, z, 0) = (0, 0, 0)$, by multipying the output with $\xi(\tau)$, and the final output is given as $(\xi(\tau)\Delta x, \xi(\tau)\Delta y, \xi(\tau)\Delta z)$ where $\xi(\tau) = \tau^{0.35}$ such that $\xi(0) = 0$ and $\xi(1) = 1$. Given the deformation offsets $(\Delta x, \Delta y, \Delta z)$, we can visualize the dynamic 4D scene defined by Gaussians at a given time τ by running $\mathbf{\Phi}$ for all Gaussians and rendering them using the renderer from Kerbl et al. [36].

C.3. Frames-Per-Second (fps) Sampling

At each 4D optimization step, we sample a fps $\in \{4, 8, 12\}$ from the distribution $p(\text{fps} = 4) = 0.81, p(\text{fps} = 8) = 0.14, p(\text{fps} = 12) = 0.05$. This scheme samples lower fps more (faster video speed) to encourage our optimization process to converge to a 4D scene with larger motion. Our time interval ranges from 0 to 1, and we set 0.75 as the length of the time interval covering a 16 frame 4 fps video (*i.e.* $\tau \in [0, 1]$ and 0.75 corresponds to a 4 seconds video). After an fps value is sampled, we sample the starting time $\tau_s \sim U(0, 1 - 3.0/\text{fps})$, and calculate the frame times for the 16 frames from τ_s and fps. This strategy is inspired by Singer et al. [79].

C.4. Rigidity Regularization

Following Luiten et al. [50], we regularize the deformation field such that nearby Gaussians deform similarly. From the 3D optimized scene after the first stage, we pre-calculate 40 nearest neighbor Gaussians for each Gaussian. Then, at each 4D optimization step, we reduce the following loss:

$$\mathcal{L}_{\text{Rigidity-Reg.}} = \frac{1}{40} \sum_{i=1}^{40} ||\mathbf{\Delta_\Phi} - \mathbf{\Delta}_{\mathbf{\Phi}_{\text{NN}_i}}||_2^2 \tag{5}$$

where the term inside the summation denotes the L2 distance between the deformation values of Gaussians and their i-th nearest neighbor.

C.5. JSD-based Regularization of the Evolving Distribution of the Dynamic 3D Gaussians

Here, we explicitly write out the regularization term introduced in Sec. 3.4, which is based on the Jensen-Shannon divergence and used to regularize the evolving distribution of the dynamic 3D Gaussians. It is used during the optimization of the deformation field when generating the temporal dynamics of the 4D sequences. The idea is to regularize the mean and the diagonal covariance of the distribution of the 3D Gaussians to stay approximately constant during the optimization of the temporal dynamics. The mean and the diagonal covariance capture an object's position (mean of the distribution of the Gaussians) and its size (variances of the distribution of the Gaussians in each xyz-direction, *i.e.*, diagonal entries of the covariance matrix). Hence, keeping the mean and the diagonal covariance approximately constant ensures that the temporal dynamics of the 4D assets induced by the video model do not simply move the objects around or change them in size, but instead produce more complex and meaningful deformations and motions, while maintaining object size and position. The technique stabilizes the optimization of the deformation field and helps ensuring that the learnt motion is realistic. Note that we also do not necessarily want the mean and covariances to be exactly the same across time, because some motion may naturally require them to vary (*e.g.* a running horse would potentially be more realistic with an oscillating center of mass), hence we use a "soft" regularization strategy instead of, for instance, a hard re-centering and re-scaling during the optimization.

Since we only want to regularize the mean and the variances in each xyz-direction, we can model the distribution of the 3D Gaussians at time τ by a Gaussian distribution $\mathcal{N}(\boldsymbol{\nu}_\tau, \mathbf{\Gamma}_\tau)$. We calculate the means $\boldsymbol{\nu}_\tau$ and diagonal covariances

$\mathbf{\Gamma}_\tau$ of the entire set of 3D Gaussians (using their means $\boldsymbol{\mu}_i$) at times τ along the 4D sequence (see Fig. 4). As explained above, we would like to ensure that this distribution stays approximately the same during the optimization of the deformation field. To this end, we choose the Jensen–Shannon Divergence (JSD) as similarity metric measuring the distance between the distributions at different times τ. The JSD between $\mathcal{N}(\boldsymbol{\nu}_0, \mathbf{\Gamma}_0)$ at time 0 and $\mathcal{N}(\boldsymbol{\nu}_\tau, \mathbf{\Gamma}_\tau)$ at time τ is

$$\begin{aligned}\text{JSD}\left(\mathcal{N}(\boldsymbol{\nu}_0, \mathbf{\Gamma}_0)||\mathcal{N}(\boldsymbol{\nu}_\tau, \mathbf{\Gamma}_\tau)\right) = &\frac{1}{2}\text{KL}\left(\mathcal{N}(\boldsymbol{\nu}_0, \mathbf{\Gamma}_0)\middle|\middle|\frac{1}{2}\left(\mathcal{N}(\boldsymbol{\nu}_0, \mathbf{\Gamma}_0) + \mathcal{N}(\boldsymbol{\nu}_\tau, \mathbf{\Gamma}_\tau)\right)\right) \\ &+ \frac{1}{2}\text{KL}\left(\mathcal{N}(\boldsymbol{\nu}_\tau, \mathbf{\Gamma}_\tau)\middle|\middle|\frac{1}{2}\left(\mathcal{N}(\boldsymbol{\nu}_0, \mathbf{\Gamma}_0) + \mathcal{N}(\boldsymbol{\nu}_\tau, \mathbf{\Gamma}_\tau)\right)\right).\end{aligned} \tag{6}$$

Unfortunately, the mixture distribution on the right hand side of the KL terms is generally not Gaussian. Therefore, it is generally not possible to derive a closed-form expression for the JSD between two Gaussians. However, we can make a simplifying modification here. Instead of calculating the mixture distribution we average the means and covariances from the two distributions $\mathcal{N}(\boldsymbol{\nu}_0, \mathbf{\Gamma}_0)$ and $\mathcal{N}(\boldsymbol{\nu}_\tau, \mathbf{\Gamma}_\tau)$ and construct a corresponding Gaussian distribution $\mathcal{N}\left(\frac{1}{2}\left(\boldsymbol{\nu}_0 + \boldsymbol{\nu}_\tau\right), \frac{1}{2}\left(\mathbf{\Gamma}_0 + \mathbf{\Gamma}_\tau\right)\right)$, which we use instead of the mixture distribution. Hence, we have

$$\begin{aligned}\text{JSD}\left(\mathcal{N}(\boldsymbol{\nu}_0, \mathbf{\Gamma}_0)||\mathcal{N}(\boldsymbol{\nu}_\tau, \mathbf{\Gamma}_\tau)\right) \rightarrow \mathcal{L}_{\text{JSD-Reg.}} := &\frac{1}{2}\text{KL}\left(\mathcal{N}(\boldsymbol{\nu}_0, \mathbf{\Gamma}_0)\middle|\middle|\mathcal{N}\left(\frac{1}{2}\left(\boldsymbol{\nu}_0 + \boldsymbol{\nu}_\tau\right), \frac{1}{2}\left(\mathbf{\Gamma}_0 + \mathbf{\Gamma}_\tau\right)\right)\right) \\ &+ \frac{1}{2}\text{KL}\left(\mathcal{N}(\boldsymbol{\nu}_\tau, \mathbf{\Gamma}_\tau)\middle|\middle|\mathcal{N}\left(\frac{1}{2}\left(\boldsymbol{\nu}_0 + \boldsymbol{\nu}_\tau\right), \frac{1}{2}\left(\mathbf{\Gamma}_0 + \mathbf{\Gamma}_\tau\right)\right)\right),\end{aligned} \tag{7}$$

which serves as our novel JSD-based regularization term $\mathcal{L}_{\text{JSD-Reg.}}$ to regularize the evolving distribution of the dynamic 3D Gaussians. Specifically, the deformation field is regularized with $\mathcal{L}_{\text{JSD-Reg.}}$ such that the mean and the variances in xyz-direction of the distribution of the 3D Gaussians at time τ do not deviate significantly from the corresponding mean and variances at time 0 (recall that the initial Gaussians at $\tau = 0$ are not subject to any deformation and hence fixed).

We can write out the above as

$$\begin{aligned}\mathcal{L}_{\text{JSD-Reg.}} := &\frac{1}{2}\text{KL}\left(\mathcal{N}(\boldsymbol{\nu}_0, \mathbf{\Gamma}_0)\middle|\middle|\mathcal{N}\left(\frac{1}{2}\left(\boldsymbol{\nu}_0 + \boldsymbol{\nu}_\tau\right), \frac{1}{2}\left(\mathbf{\Gamma}_0 + \mathbf{\Gamma}_\tau\right)\right)\right) \\ &+ \frac{1}{2}\text{KL}\left(\mathcal{N}(\boldsymbol{\nu}_\tau, \mathbf{\Gamma}_\tau)\middle|\middle|\mathcal{N}\left(\frac{1}{2}\left(\boldsymbol{\nu}_0 + \boldsymbol{\nu}_\tau\right), \frac{1}{2}\left(\mathbf{\Gamma}_0 + \mathbf{\Gamma}_\tau\right)\right)\right) \\ = &\sum_{i\in\{x,y,z\}}\left[-\frac{1}{2}\log\left[2\right] + \frac{1}{2}\log\left[\mathbf{\Gamma}_0^i + \mathbf{\Gamma}_\tau^i\right] - \frac{1}{4}\log\left[\mathbf{\Gamma}_0^i\right] - \frac{1}{4}\log\left[\mathbf{\Gamma}_\tau^i\right] + \frac{1}{4}\frac{(\boldsymbol{\nu}_\tau^i - \boldsymbol{\nu}_0^i)^2}{\mathbf{\Gamma}_0^i + \mathbf{\Gamma}_\tau^i}\right],\end{aligned} \tag{8}$$

for the three dimensions $i \in \{x, y, z\}$ and where $\mathbf{\Gamma}^i$ denotes the i-th diagonal entry in the diagonal covariance matrix $\mathbf{\Gamma}$ (since we use diagonal covariance matrices, we obtain this simple factorized form). The gradients with respect to $\boldsymbol{\nu}_\tau$ and $\mathbf{\Gamma}_\tau$, which are calculated from the learnt distribution of the evolving 3D Gaussians, are

$$\nabla_{\boldsymbol{\nu}_\tau^i}\mathcal{L}_{\text{JSD-Reg.}} = \frac{1}{2}\frac{\boldsymbol{\nu}_\tau^i - \boldsymbol{\nu}_0^i}{\mathbf{\Gamma}_0^i + \mathbf{\Gamma}_\tau^i}, \tag{9}$$

$$\nabla_{\mathbf{\Gamma}_\tau^i}\mathcal{L}_{\text{JSD-Reg.}} = \frac{1}{2}\frac{1}{\mathbf{\Gamma}_0^i + \mathbf{\Gamma}_\tau^i} - \frac{1}{4}\frac{1}{\mathbf{\Gamma}_\tau^i} - \frac{1}{4}\frac{(\boldsymbol{\nu}_\tau^i - \boldsymbol{\nu}_0^i)^2}{(\mathbf{\Gamma}_0^i + \mathbf{\Gamma}_\tau^i)^2}. \tag{10}$$

By analyzing the first and second derivatives it is easy to see that $\mathcal{L}_{\text{JSD-Reg.}}$ has a unique minimum when

$$\boldsymbol{\nu}_\tau^i = \boldsymbol{\nu}_0^i \tag{11}$$

and

$$\mathbf{\Gamma}_\tau^i = \frac{1}{2}\left[(\boldsymbol{\nu}_\tau^i - \boldsymbol{\nu}_0^i)^2 + \sqrt{(\boldsymbol{\nu}_\tau^i - \boldsymbol{\nu}_0^i)^4 + 4{\mathbf{\Gamma}_0^i}^2}\right] \overset{\boldsymbol{\nu}_\tau^i = \boldsymbol{\nu}_0^i}{=} \mathbf{\Gamma}_0^i \tag{12}$$

for all $i \in \{x, y, z\}$. This implies that $\mathcal{L}_{\text{JSD-Reg.}}$ is a meaningful regularization term for the means and variances in xyz-direction of the distribution of the evolving dynamic 3D Gaussians. It ensures that the distribution's mean $\boldsymbol{\nu}_\tau$ and diagonal covariance $\mathbf{\Gamma}_\tau$ remain close to the initial $\boldsymbol{\nu}_0$ and $\mathbf{\Gamma}_0$ during the optimization of the deformation field. That said, note that

$\mathcal{L}_{\text{JSD-Reg.}}$ is not necessarily an accurate approximation of the true JSD. However, we found that this approach worked very well in practice during score distillation.

We opted for the JSD-based approach, because we wanted to use a symmetric distribution similarity metric. Potentially, we could have formulated our regularization also based on a Wasserstein distance, symmetrized KL divergence or regular non-symmetric KL divergence. However, we implemented the JSD-based approach above first and it worked right away and significantly improved AYG's optimization behavior and the 4D results. Therefore, we decided to leave the exploration of differently defined regularizations for the evolving distribution of the dynamic 3D Gaussians to future research.

C.6. Camera Distribution

In the initial 3D stage, we follow MVDream [76] to sample random cameras. At each optimization step, we sample a field-of-view $fov \sim U(15, 60)$, an elevation angle $elv \sim U(10, 45)$ and an azimuth angle $azm \sim U(0, 360)$. The camera distance is calculated as $cam\_d = s/\tan(\frac{fov}{2} * \frac{\pi}{180})$ where $s \sim U(0.8, 1.0)$ is a random scaling factor. The sampled camera looks towards the origin (0,0,0) and the images are rendered at 256×256 resolution.

At each 4D optimization step, we sample $fov \sim U(40, 70)$, $elv \sim U(-10, 45)$, $azm \sim U(0, 360)$, and $cam\_d \sim U(1.5, 3.0)$. We use *dynamic* cameras where the camera location changes across the temporal time domain. For this purpose, we additionally sample offset values $elv\_offset \sim U(-13.5, 30)$ and $azm\_offset \sim U(-45, 45)$ and construct 16 cameras (to be used for rendering, and consequently, to be used as input to the video diffusion model that expects a 16-frame input) by setting $elv_i = elv + elv\_offset * i/15$ and $azm_i = azm + azm\_offset * i/15$ for $i \in \{0, ..., 15\}$, where elv_i and azm_i corresponds to the elevation and azimuth angles of the i-th frame's camera. We use the same field-of-view and camera distance across the 16 frames. We render frames at 160×256 resolution, which conforms to the aspect ratio that the video diffusion model is trained on.

C.7. Diffusion Models

As discussed, AYG leverages a latent text-to-image diffusion model [70], a latent text-to-video diffusion model [7], and a latent text-to-multiview-image diffusion model [76] in its compositional 4D score distillation framework.

For the latent text-to-multiview-image diffusion model we use MVDream [76].[3] For the latent text-to-video diffusion model, we train our own model, largely following VideoLDM [7] but with more training data and additional fps conditioning, see Appendix E.1. For the latent text-to-image diffusion model we use the spatial backbone of our newly trained video diffusion model, removing its temporal layers. This spatial backbone corresponds to a version of Stable Diffusion 2.1 [70][4] that was fine-tuned for generation at resolution 320×512 (see details in Blattmann et al. [7]).

C.8. Rendering Resolution

In the initial 3D synthesis stage, we render the scenes at resolution 256×256. This is the resolution at which MVDream operates, *i.e.*, its encoder takes images of this resolution. We additionally perform bilinear rescaling to 320×512 to calculate our text-to-image model's score distillation gradient. The text-to-image model's encoder accepts images of this resolution after fine-tuning (see previous subsection). In the main 4D synthesis stage, we render at resolution 256×160 and perform bilinear rescaling to 320×512 to calculate both the text-to-image model's and the text-to-video model's score distillation gradients. They share the same encoder, which was fine-tuned to operate at this resolution. For evaluation and visualization, we render at 512×320 resolution and pad to 512×512 if necessary.

C.9. Additional Fine-tuning and Optimization

We explored adding more Gaussians to further improve the quality of the generated 4D scenes. As mentioned in Appendix C.1, we only perform densification steps if the total number of Gaussians is less than 50,000 for the initial 3D synthesis stage, and then optimize the deformation field in our 4D synthesis stage, as we found that this strategy gave us the best motion. Once optimized, we can further fine-tune the dynamic 4D scenes by doing a similar two-stage optimization. In the first additional stage, we continue optimizing the 3D Gaussians (not the deformation field) and do further densification steps even when the number of Gaussians is more than 50,000. On average, we end up with 150,000 Gaussians in this stage. After that, in the second fine-tuning stage, we continue optimizing the deformation field, which has already been optimized well previously, with all the Gaussians including the newly added ones. We use the same hyperparameters as in the initial 3D/4D stages except that we reduce the learning rates by 1/5 and increase the rendering image resolution to 512×512 for

[3] https://huggingface.co/MVDream/MVDream

[4] https://huggingface.co/stabilityai/stable-diffusion-2-1

the additional 3D optimization stage. For the two additional stages, we optimize for 7,000 steps and 3,000 steps on average, respectively. Also see the hyperparameters in Appendix E.2.

All results shown on the project page, https://research.nvidia.com/labs/toronto-ai/AlignYourGaussians/, correspond to 4D assets subject to this additional fine-tuning. All results shown in the paper itself as well as in the supplementary video `ayg_text_to_4d.mp4` only use a single 3D plus 4D optimization run without this additional fine-tuning. Also the quantitative and qualitative comparisons to MAV3D [79] in Sec. 4 and Appendix F.1 only use a single 3D and 4D optimization. This implies that with the additional fine-tuning described here and shown on the project page, we would possibly outperform MAV3D by an even larger margin in the user study. Also the ablation studies were carried out with only a single 3D plus 4D optimization without fine-tuning, and in that case we additionally only optimized for a total of 4,000 steps in the second 4D optimization stage in the interest of efficiency (see Sec. 4 and Appendix F.2).

D. Align Your Gaussian's Synthesis Framework

Here, we discuss AYG's compositional generation framework in more detail and derive AYG's score distillation scheme.

D.1. AYG's Compositional Generation Framework

Our goal during AYG's main 4D distillation is to simultaneously synthesize smooth and realistic temporal dynamics while also maintaining a high individual frame quality. Formally, this means that we would like to fulfill two objectives at once:

1. A set of 2D frames $\{\mathbf{x}_{\tau_i}^{c_i}\}_F$ rendered from the 4D sequence at consecutive time steps τ_i and smoothly changing camera positions c_i (dynamic cameras) should form a realistic dynamic video and should be high probability under the text-to-video model $p_{\text{vid}}(\{\mathbf{z}_{\tau_i}^{c_i}\}_F)$.
2. Individual frames $\tilde{\mathbf{x}}_{\tilde{\tau}_j}^{\tilde{c}_j}$ at possibly different time steps $\tilde{\tau}_j$ and camera positions $\tilde{c}_j$ should, each individually, be high probability under the text-to-image model $p_{\text{im}}(\tilde{\mathbf{x}}_{\tilde{\tau}_j}^{\tilde{c}_j})$.

If we want to ensure that both criteria are fulfilled simultaneously, it means that we are interested in generating 4D sequences whose renderings obey *the product distribution* of p_{vid} and p_{im}, *i.e.*, $p_{\text{vid}}\, p_{\text{im}}$. This exactly corresponds to a form of compositional generation [19, 43]. In compositional generation, one often combines different diffusion models, or the same diffusion model with different text-conditionings, such that multiple conditions are fulfilled during generation, usually with the goal of controllable image generation or image editing. In that case, one leverages the product distribution of the corresponding diffusion model distributions with the different conditionings. Conceptually, this is analogous to what we are doing in AYG, just that we are doing a version of compositional generation by composing an image and a video diffusion model to generate dynamic 4D assets whose renderings obey both the text-to-image and text-to-video model distributions simultaneously. Instead of aiming for controllability our goal is to simultaneously enforce smooth temporal dynamics (video model) and high individual frame quality (image model).

One may ask, why do we even need the additional text-to-image model? Should the video model itself not also enforce high individual frame quality? The number of text-video pairs used to train text-to-video models is typically significantly smaller than the huge text-image datasets utilized when training text-to-image models. This makes text-to-video models, including the one we trained, often slightly less robust and more unstable than text-to-image models and they yield noisier gradients in the score distillation setting. Generally, video diffusion models are not as mature as image diffusion models yet. This motivates us to compose both, the image and the video model, where the video model can synthesize temporal dynamics and the image model ensures that high individual frame quality is maintained despite potentially somewhat noisy gradients of the video model. Looking at it from another perspective, the image model essentially stabilizes the optimization process.

We have explained this for the 4D generation stage here, but something very similar occurs also in AYG's initial 3D generation stage, where a text-conditioned multiview-image diffusion model [76] is composed with a regular large-scale text-to-image model [70] in the form of their product distribution for score distillation. The motivation is almost the same. The multiview image model is able to generate a set of multiview-consistent images, similar to the set of consecutive frames of the video model, but it was trained only on a limited 3D dataset and therefore does encode the biases of this small 3D dataset. Including an additional general text-to-image prior can stabilize optimization and boost performance. Moreover, by augmenting the image model's text conditioning with view prompts, we can easily break the symmetry inherent in the multiview diffusion model, which does not generate objects in a canonical orientation. Note, however, that our main contribution is the compositional generation approach for AYG's 4D stage and this has been our main focus in this paper. AYG's 4D stage could also be combined with other methods to generate the initial static 3D assets.

D.2. AYG's Score Distillation Scheme

Let us now formalize this, while focusing on the 4D stage (derivations for the 3D stage proceed analogously). Following the above motivation, when optmizing the deformation field $\mathbf{\Phi}$ through score distillation we seek to minimize a reverse Kullback-Leibler divergence (KLD) from $q_{\mathbf{\Phi}}$, the distribution over the 4D scene's renderings defined through the deformation field $\mathbf{\Phi}$, to the product distribution $p_{\text{vid}}\, p_{\text{im}}$ of the diffusion models used as teachers, *i.e.*, $\text{KL}\big(q_{\mathbf{\Phi}} \big|\big| p_{\text{vid}} p_{\text{im}}\big)$. Since p_{vid} and p_{im} are defined through diffusion models, in practice we minimize the KL not just for the clean renderings, but for diffused renderings at all diffusion time steps t, *i.e.*,

$$\text{KL}\Bigg(q_{\mathbf{\Phi}}\left(\{\mathbf{z}_{\tau_i}^{c_i}\}_F, \{\tilde{\mathbf{z}}_{\tilde{\tau}_j}^{\tilde{c}_j}\}_M\right) \Bigg|\Bigg| p_{\text{vid}}^{\gamma}\left(\{\mathbf{z}_{\tau_i}^{c_i}\}_F\right) \prod_{j=1}^{M} p_{\text{im}}^{\kappa}\left(\tilde{\mathbf{z}}_{\tilde{\tau}_j}^{\tilde{c}_j}\right)\Bigg), \tag{13}$$

which is exactly the equation from the main paper in Sec. 3.2. Here, we now added some details: In general we can always consider a set of $F + M$ rendered frames, where $\{\mathbf{z}_{\tau_i}^{c_i}\}_F$ are the F consecutive frames given to the video model and $\{\tilde{\mathbf{z}}_{\tilde{\tau}_j}^{\tilde{c}_j}\}_M$ are another M independent frames given to the image model (since these are independent, the image model operates on them independently; hence the product in the equation). Following the notation in the main paper, $\mathbf{z}$ denotes diffused renderings and we omit diffusion time subscripts t. Note that, as discussed in the main paper, in practice the M frames given to the image model are just a subset of the F frames given to the video model. However, the framework is general. Moreover, γ and κ are temperatures which scale the width of the distributions p_{vid} and p_{im}. They can act as hyperparameters controlling the relative strength of the gradients coming from the video and image model, respectively. Note that for now we omit explicitly indicating the text conditioning in the equations in the interest of brevity.

The derivation of our score distillation framework through minimizing the reverse KLD between the rendering distribution and the teacher distribution follows Poole et al. [62] (Appendix A.4). In practice, one typically gives different weights to the KL minimization for different diffusion times t, and the full SDS gradient can be written as

$$\nabla_{\mathbf{\Phi}} \mathcal{L}_{\text{SDS}}^{\text{AYG}}(\{\mathbf{x}\} = g(\mathbf{\Phi})) = \mathbb{E}_{t,\boldsymbol{\epsilon}}\Bigg[w(t)\frac{\sigma_t}{\alpha_t}\nabla_{\mathbf{\Phi}}\text{KL}\Bigg(q_{\mathbf{\Phi}}\left(\{\mathbf{z}_{\tau_i}^{c_i}\}_F, \{\tilde{\mathbf{z}}_{\tilde{\tau}_j}^{\tilde{c}_j}\}_M\right) \Bigg|\Bigg| p_{\text{vid}}^{\gamma}\left(\{\mathbf{z}_{\tau_i}^{c_i}\}_F\right) \prod_{j=1}^{M} p_{\text{im}}^{\kappa}\left(\tilde{\mathbf{z}}_{\tilde{\tau}_j}^{\tilde{c}_j}\right)\Bigg)\Bigg], \tag{14}$$

where $w(t)$ is a weighting function and α_t and σ_t are the diffusion models' noise schedule (note that all our diffusion models use the same noise schedules). Specifically, the diffused renderings $\mathbf{z}$ are given as $\mathbf{z} = \alpha_t \mathbf{x} + \sigma_t \boldsymbol{\epsilon}$, where $\mathbf{x}$ is a clean rendering, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ is noise from a standard normal distribution, and the parameters α_t and σ_t are defined such the signal-to-noise ratio α_t/σ_t is strictly decreasing as a function of the diffusion time t. Moreover, usually $\alpha_t \in [0, 1]$ and $\sigma_t \in [0, 1]$ and in our case we use a "variance-preserving" noise schdule [27, 81] such that $\alpha_t^2 + \sigma_t^2 = 1$. Furthermore, in Eq. (14) above, g denotes the differentiable rendering function of the 4D scene, which produces all clean renderings $\{\mathbf{x}\}_{F+M}$ (omitting camera and 4D sequence time sub- and superscripts to keep notation concise), which are then diffused into $\{\mathbf{z}_{\tau_i}^{c_i}\}_F$ and $\{\tilde{\mathbf{z}}_{\tilde{\tau}_j}^{\tilde{c}_j}\}_M$.

We can now expand the KL divergence, similar to Poole et al. [62]:

$$\nabla_{\mathbf{\Phi}} \mathcal{L}_{\text{SDS}}^{\text{AYG}}(\{\mathbf{x}\}_{F+M} = g(\mathbf{\Phi})) = \mathbb{E}_{t,\boldsymbol{\epsilon}}\Bigg[w(t)\frac{\sigma_t}{\alpha_t}\Bigg(\nabla_{\mathbf{\Phi}} \log q_{\mathbf{\Phi}}\left(\{\mathbf{z}_{\tau_i}^{c_i}\}_F, \{\tilde{\mathbf{z}}_{\tilde{\tau}_j}^{\tilde{c}_j}\}_M\right) - \nabla_{\mathbf{\Phi}} \log\Bigg[p_{\text{vid}}^{\gamma}\left(\{\mathbf{z}_{\tau_i}^{c_i}\}_F\right) \prod_{j=1}^{M} p_{\text{im}}^{\kappa}\left(\tilde{\mathbf{z}}_{\tilde{\tau}_j}^{\tilde{c}_j}\right)\Bigg]\Bigg)\Bigg]. \tag{15}$$

Let us now introduce some abbreviations for brevity, as in the main paper, and define $\mathbf{Z} := \{\mathbf{z}_{\tau_i}^{c_i}\}_F$, $\tilde{\mathbf{Z}} := \{\tilde{\mathbf{z}}_{\tilde{\tau}_j}^{\tilde{c}_j}\}_M$ and $\mathbf{X} := \{\mathbf{x}\}_{F+M}$. Moreover, for a concise notation we write the product over the image distributions as one distribution over all the different diffused renderings with a slight abuse of notation, *i.e.*,

$$\prod_{j=1}^{M} p_{\text{im}}^{\kappa}\left(\tilde{\mathbf{z}}_{\tilde{\tau}_j}^{\tilde{c}_j}\right) = p_{\text{im}}^{\kappa}\left(\{\tilde{\mathbf{z}}_{\tilde{\tau}_j}^{\tilde{c}_j}\}_M\right) = p_{\text{im}}^{\kappa}\left(\tilde{\mathbf{Z}}\right). \tag{16}$$

We can keep in mind that $p_{\text{im}}^{\kappa}\left(\tilde{\mathbf{Z}}\right)$ decomposes into a product, which will lead to different independent gradients. Now we

have

$$\begin{aligned}\nabla_{\boldsymbol{\Phi}} \mathcal{L}_{\text{SDS}}^{\text{AYG}}(\mathbf{X} = g(\boldsymbol{\Phi})) &= \mathbb{E}_{t,\boldsymbol{\epsilon}}\left[w(t)\frac{\sigma_t}{\alpha_t}\left(\nabla_{\boldsymbol{\Phi}} \log q_{\boldsymbol{\Phi}}\left(\mathbf{Z}, \tilde{\mathbf{Z}}\right) - \nabla_{\boldsymbol{\Phi}} \log\left[p_{\text{vid}}^{\gamma}\left(\mathbf{Z}\right) p_{\text{im}}^{\kappa}\left(\tilde{\mathbf{Z}}\right)\right]\right)\right] \\ &= \mathbb{E}_{t,\boldsymbol{\epsilon}}\left[w(t)\frac{\sigma_t}{\alpha_t}\left(\nabla_{\boldsymbol{\Phi}} \log q_{\boldsymbol{\Phi}}\left(\mathbf{Z}\right) - \gamma\, \nabla_{\boldsymbol{\Phi}} \log p_{\text{vid}}\left(\mathbf{Z}\right)\right.\right. \\ &\qquad \left.\left. + \nabla_{\boldsymbol{\Phi}} \log q_{\boldsymbol{\Phi}}\left(\tilde{\mathbf{Z}}\right) - \kappa\, \nabla_{\boldsymbol{\Phi}} \log p_{\text{im}}\left(\tilde{\mathbf{Z}}\right)\right)\right],\end{aligned} \tag{17}$$

where we decomposed the log-terms. We can now analyze the individual terms, analogous to Poole et al. [62]. We have

$$\nabla_{\boldsymbol{\Phi}} \log p_{\text{vid}}\left(\mathbf{Z}\right) = \nabla_{\mathbf{Z}} \log p_{\text{vid}}\left(\mathbf{Z}\right) \frac{\partial \mathbf{Z}}{\partial \boldsymbol{\Phi}} = -\frac{1}{\sigma_t}\hat{\boldsymbol{\epsilon}}^{\text{vid}}(\mathbf{Z}, v, t)\frac{\partial \mathbf{Z}}{\partial \boldsymbol{\Phi}} = -\frac{\alpha_t}{\sigma_t}\hat{\boldsymbol{\epsilon}}^{\text{vid}}(\mathbf{Z}, v, t)\frac{\partial \mathbf{X}}{\partial \boldsymbol{\Phi}}, \tag{18}$$

where we inserted the text-to-video model's denoiser neural network $\hat{\boldsymbol{\epsilon}}^{\text{vid}}(\mathbf{Z}, v, t)$, which represents the diffusion model's score $\nabla_{\mathbf{Z}} \log p_{\text{vid}}\left(\mathbf{Z}\right)$ as [27, 81, 86]

$$\nabla_{\mathbf{Z}} \log p_{\text{vid}}\left(\mathbf{Z}\right) \sim -\frac{1}{\sigma_t}\hat{\boldsymbol{\epsilon}}^{\text{vid}}(\mathbf{Z}, v, t), \tag{19}$$

and we now also explicitly indicate the model's conditioning on the diffusion time t and a text prompt v. The image model term $\nabla_{\boldsymbol{\Phi}} \log p_{\text{im}}\left(\tilde{\mathbf{Z}}\right)$ can be written similarly. Moreover, we have

$$\begin{aligned}\nabla_{\boldsymbol{\Phi}} \log q_{\boldsymbol{\Phi}}\left(\mathbf{Z}\right) &= \left(\frac{\partial \log q_{\boldsymbol{\Phi}}\left(\mathbf{Z}\right)}{\partial \mathbf{X}} + \frac{\partial \log q_{\boldsymbol{\Phi}}\left(\mathbf{Z}\right)}{\partial \mathbf{Z}}\frac{\partial \mathbf{Z}}{\partial \mathbf{X}}\right)\frac{\partial \mathbf{X}}{\partial \boldsymbol{\Phi}} \\ &= \left(-\frac{\partial}{\partial \mathbf{X}}(\alpha_t \mathbf{X} - \mathbf{Z})^2 - \alpha_t \frac{\partial}{\partial \mathbf{Z}}(\alpha_t \mathbf{X} - \mathbf{Z})^2\right)\frac{1}{2\sigma_t^2}\frac{\partial \mathbf{X}}{\partial \boldsymbol{\Phi}} \\ &= \left(-(\alpha_t \mathbf{X} - \mathbf{Z}) + (\alpha_t \mathbf{X} - \mathbf{Z})\right)\frac{\alpha_t}{\sigma_t^2}\frac{\partial \mathbf{X}}{\partial \boldsymbol{\Phi}} \\ &= \left(-(\alpha_t \mathbf{X} - \alpha_t \mathbf{X} - \sigma_t \boldsymbol{\epsilon}) + (\alpha_t \mathbf{X} - \alpha_t \mathbf{X} - \sigma_t \boldsymbol{\epsilon})\right)\frac{\alpha_t}{\sigma_t^2}\frac{\partial \mathbf{X}}{\partial \boldsymbol{\Phi}} \\ &= \left(\frac{\alpha_t}{\sigma_t}\boldsymbol{\epsilon} - \frac{\alpha_t}{\sigma_t}\boldsymbol{\epsilon}\right)\frac{\partial \mathbf{X}}{\partial \boldsymbol{\Phi}} \\ &= 0\end{aligned} \tag{20}$$

which follows exactly Poole et al. [62], Appendix A.4, and analogously for $\nabla_{\boldsymbol{\Phi}} \log q_{\boldsymbol{\Phi}}\left(\tilde{\mathbf{Z}}\right)$. The first term in Eq. (20) corresponds to the gradient directly with respect to the variational parameters $\boldsymbol{\Phi}$ through the renderings $\mathbf{X}$, while the second term is the path derivative through the diffused samples $\mathbf{Z}$ [69]. Note that in score distillation, we take an expectation over $\boldsymbol{\epsilon}$ and after such expectation both terms are individually zero as the noise $\boldsymbol{\epsilon}$ has $\mathbf{0}$ mean.

We can now write our SDS gradient as

$$\nabla_{\boldsymbol{\Phi}} \mathcal{L}_{\text{SDS}}^{\text{AYG}}(\mathbf{X} = g(\boldsymbol{\Phi})) = \mathbb{E}_{t,\boldsymbol{\epsilon}}\left[w(t)\left(\gamma\, \hat{\boldsymbol{\epsilon}}^{\text{vid}}(\mathbf{Z}, v, t) + \kappa\, \hat{\boldsymbol{\epsilon}}^{\text{im}}(\tilde{\mathbf{Z}}, v, t)\right)\right]\frac{\partial \mathbf{X}}{\partial \boldsymbol{\Phi}}. \tag{21}$$

However, in SDS one typically includes the path derivative gradient (the second term in Eq. (20)) as a zero mean control variate to reduce the variance of the SDS gradient when using a sample-based approximation of the expectation, following Roeder et al. [69]. In other words, since

$$\mathbb{E}_{t,\boldsymbol{\epsilon}}\left[-w(t)\frac{\alpha_t}{\sigma_t}\boldsymbol{\epsilon}\right]\frac{\partial \mathbf{X}}{\partial \boldsymbol{\Phi}} = 0 \tag{22}$$

after the expectation of the diffusion noise $\boldsymbol{\epsilon}$, we can freely include such terms in the SDS gradient as control variates and scale with the temperatures γ and κ as needed. Now explicitly defining the different noise values $\boldsymbol{\epsilon}^{\text{vid}}$ and $\boldsymbol{\epsilon}^{\text{im}}$ as the diffusion noises for the perturbed renderings given to the video and the image model, respectively, we can write

$$\begin{aligned}\nabla_{\boldsymbol{\Phi}} \mathcal{L}_{\text{SDS}}^{\text{AYG}}(\mathbf{X} = g(\boldsymbol{\Phi})) &= \mathbb{E}_{t,\boldsymbol{\epsilon}}\left[w(t)\left(\gamma\, \hat{\boldsymbol{\epsilon}}^{\text{vid}}(\mathbf{Z}, v, t) + \kappa\, \hat{\boldsymbol{\epsilon}}^{\text{im}}(\tilde{\mathbf{Z}}, v, t)\right)\right]\frac{\partial \mathbf{X}}{\partial \boldsymbol{\Phi}} \\ &= \mathbb{E}_{t,\boldsymbol{\epsilon}^{\text{vid}},\boldsymbol{\epsilon}^{\text{im}}}\left[w(t)\left(\gamma\left(\hat{\boldsymbol{\epsilon}}^{\text{vid}}(\mathbf{Z}, v, t) - \boldsymbol{\epsilon}^{\text{vid}}\right) + \kappa\left(\hat{\boldsymbol{\epsilon}}^{\text{im}}(\tilde{\mathbf{Z}}, v, t) - \boldsymbol{\epsilon}^{\text{im}}\right)\right)\right]\frac{\partial \mathbf{X}}{\partial \boldsymbol{\Phi}}.\end{aligned} \tag{23}$$

Note that we have been using different noise values to perturb the different renderings from the very beginning of the derivation, but we have not been explicit about it in the notation so far in the interest of conciseness.

In practice, one typically employs classifier-free guidance (CFG) [26] with guidance weights $\omega_{\text{vid/im}}$ for the video and image model, respectively. Hence, we have that

$$\hat{\boldsymbol{\epsilon}}^{\text{vid}}(\mathbf{Z}, v, t) \rightarrow \hat{\boldsymbol{\epsilon}}^{\text{vid}}(\mathbf{Z}, v, t) + \omega_{\text{vid}} \left[\hat{\boldsymbol{\epsilon}}^{\text{vid}}(\mathbf{Z}, v, t) - \hat{\boldsymbol{\epsilon}}^{\text{vid}}(\mathbf{Z}, t)\right] \tag{24}$$

for the video model, and for the image model analogously ($\hat{\boldsymbol{\epsilon}}^{\text{vid}}(\mathbf{Z}, t)$ indicates the denoiser prediction without text conditioning). The no-CFG setting is recovered for $\omega_{\text{vid/im}} = 0$. Inserting above, we obtain

$$\begin{aligned}\nabla_{\boldsymbol{\Phi}} \mathcal{L}^{\text{AYG}}_{\text{SDS}}(\mathbf{X} = g(\boldsymbol{\Phi})) = \mathbb{E}_{t, \boldsymbol{\epsilon}^{\text{vid}}, \boldsymbol{\epsilon}^{\text{im}}} \Bigg[w(t) \Bigg\{ \gamma \Bigg(\omega_{\text{vid}} \left[\hat{\boldsymbol{\epsilon}}^{\text{vid}}(\mathbf{Z}, v, t) - \hat{\boldsymbol{\epsilon}}^{\text{vid}}(\mathbf{Z}, t)\right] + \underbrace{\hat{\boldsymbol{\epsilon}}^{\text{vid}}(\mathbf{Z}, v, t) - \boldsymbol{\epsilon}^{\text{vid}}}_{\boldsymbol{\delta}^{\text{vid}}_{\text{gen}}} \Bigg) \\ + \kappa \Bigg(\omega_{\text{im}} \left[\hat{\boldsymbol{\epsilon}}^{\text{im}}(\tilde{\mathbf{Z}}, v, t) - \hat{\boldsymbol{\epsilon}}^{\text{im}}(\tilde{\mathbf{Z}}, t)\right] + \underbrace{\hat{\boldsymbol{\epsilon}}^{\text{im}}(\tilde{\mathbf{Z}}, v, t) - \boldsymbol{\epsilon}^{\text{im}}}_{\boldsymbol{\delta}^{\text{im}}_{\text{gen}}} \Bigg) \Bigg\} \frac{\partial \mathbf{X}}{\partial \boldsymbol{\Phi}} \Bigg],\end{aligned} \tag{25}$$

which is exactly Eq. (3) from the main paper.

As discussed in the main paper, recently, ProlificDreamer [92] proposed a scheme where the noise-based control variate $\boldsymbol{\epsilon}$ ($\boldsymbol{\epsilon}^{\text{vid}}$ and $\boldsymbol{\epsilon}^{\text{im}}$ here) is replaced by a separate diffusion model that models the rendering distribution. More specifically, ProlificDreamer initializes this separate diffusion model from the diffusion model guiding the synthesis ($\hat{\boldsymbol{\epsilon}}^{\text{vid}}(\mathbf{Z}, v, t)$ and $\hat{\boldsymbol{\epsilon}}^{\text{im}}(\tilde{\mathbf{Z}}, v, t)$ here), and then slowly fine-tunes on the diffused renderings ($\mathbf{Z}$ or $\tilde{\mathbf{Z}}$ here). In our setting, one would initialize from the video and the image model, respectively, in the two terms. This means that, at the beginning of optimization, the terms $\boldsymbol{\delta}^{\text{vid/im}}_{\text{gen}}$ in Eq. (25) would be zero. Inspired by this observation and aiming to avoid ProlificDreamer's cumbersome fine-tuning, we instead propose to simply set $\boldsymbol{\delta}^{\text{vid/im}}_{\text{gen}} = 0$ entirely, and simply optimize with

$$\begin{aligned}\nabla_{\boldsymbol{\Phi}} \mathcal{L}^{\text{AYG}}_{\text{CSD}}(\mathbf{X} = g(\boldsymbol{\Phi})) &= \mathbb{E}_{t, \boldsymbol{\epsilon}^{\text{vid}}, \boldsymbol{\epsilon}^{\text{im}}} \left[w(t) \left\{ \omega_{\text{vid}} \underbrace{\left[\hat{\boldsymbol{\epsilon}}^{\text{vid}}(\mathbf{Z}, v, t) - \hat{\boldsymbol{\epsilon}}^{\text{vid}}(\mathbf{Z}, t)\right]}_{\boldsymbol{\delta}^{\text{vid}}_{\text{cls}}} + \omega_{\text{im}} \underbrace{\left[\hat{\boldsymbol{\epsilon}}^{\text{im}}(\tilde{\mathbf{Z}}, v, t) - \hat{\boldsymbol{\epsilon}}^{\text{im}}(\tilde{\mathbf{Z}}, t)\right]}_{\boldsymbol{\delta}^{\text{im}}_{\text{cls}}} \right\} \frac{\partial \mathbf{X}}{\partial \boldsymbol{\Phi}} \right] \\ &= \mathbb{E}_{t, \boldsymbol{\epsilon}^{\text{vid}}, \boldsymbol{\epsilon}^{\text{im}}} \left[w(t) \left\{ \omega_{\text{vid}} \boldsymbol{\delta}^{\text{vid}}_{\text{cls}} + \omega_{\text{im}} \boldsymbol{\delta}^{\text{im}}_{\text{cls}} \right\} \frac{\partial \mathbf{X}}{\partial \boldsymbol{\Phi}} \right],\end{aligned} \tag{26}$$

which is exactly Eq. (4) from the main paper. Moreover, we have absorbed γ and κ into $\omega_{\text{vid/im}}$. Interestingly, this exactly corresponds to the concurrently proposed classifier score distillation (CSD) [102], which points out that the above two terms $\boldsymbol{\delta}^{\text{vid/im}}_{\text{cls}}$ correspond to implicit classifiers predicting the text v from the video or images, respectively. CSD then proposes to use only $\boldsymbol{\delta}^{\text{vid/im}}_{\text{cls}}$ for score distillation, resulting in improved performance over SDS. We discovered that scheme independently, while aiming to inherit ProlificDreamer's strong performance.

The score for the first 3D stage looks exactly analogous, just that instead of the composition of the video and image diffusion model, we have the composition of the MVDream multiview image diffusion model and the regular text-to-image model.

D.3. AYG's Parameter Gradients—Putting it All Together

In practice, AYG additionally uses negative prompt guidance, motion amplification, JSD regularization, and rigidity regularization in the 4D stage, as well as view guidance in the initial 3D stage, see Sec. 3.4. Let us now put everything together.

Stage 1: 3D Synthesis. In the 3D stage, the entire gradient backpropagated into the 3D Gaussian splatting representation $\boldsymbol{\theta}$ is

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} \mathcal{L}^{\text{AYG}}_{\text{3D-stage}} = \mathbb{E}_{t, \boldsymbol{\epsilon}^{\text{3D}}, \boldsymbol{\epsilon}^{\text{im}}} \Bigg[w(t) \Bigg\{ &\omega_{\text{3D}} \boldsymbol{\delta}^{\text{3D}}_{\text{cls}} + \omega_{\text{im}} \boldsymbol{\delta}^{\text{im}}_{\text{cls}} \\ &+ \omega_{\text{vg}} \left(\hat{\boldsymbol{\epsilon}}^{\text{im}}(\tilde{\mathbf{Z}}, v^{\text{aug}}, t) - \hat{\boldsymbol{\epsilon}}^{\text{im}}(\tilde{\mathbf{Z}}, v, t) \right) \\ &+ \omega_{\text{neg}} \left(\hat{\boldsymbol{\epsilon}}^{\text{3D}}(\mathbf{Z}, t) - \hat{\boldsymbol{\epsilon}}^{\text{3D}}(\mathbf{Z}, v^{\text{neg}}, t) \right) \Bigg\} \frac{\partial \mathbf{X}}{\partial \boldsymbol{\theta}} \Bigg],\end{aligned} \tag{27}$$

where $\boldsymbol{\delta}^{\text{3D}}_{\text{cls}}$ corresponds to the implicit classifier score for the text-conditioned 3D-aware MVDream multiview diffusion model, analogous to the implicit classifier score for the video model above. Moreover, we added the view guidance and

negative prompting terms [102]. Our negative prompt v^{neg} in the 3D stage is *"ugly, bad anatomy, blurry, pixelated obscure, unnatural colors, poor lighting, dull, and unclear, cropped, lowres, low quality, artifacts, duplicate, morbid, mutilated, poorly drawn face, deformed, dehydrated, bad proportions"*, following Shi et al. [76]. We apply negative prompting only to the MVDream 3D multiview diffusion model (the negative prompt is taken from MVDream, after all). For view guidance, we attach ", front view" ($azm \leq 30°$ or $azm \geq 330°$), ", back view" ($150° \leq azm \leq 210°$), ", overhead view" ($elv \geq 60°$) and ", side view" (otherwise) at the end of a text prompt to construct the augmented prompt v^{aug}. Furthermore, $\boldsymbol{\epsilon}^{\text{3D}}$ and $\boldsymbol{\epsilon}^{\text{im}}$ denote the diffusion noise values used to perturb the renderings given to the 3D-aware diffusion model and the regular text-to-image model, respectively.

Stage 2: Adding Dynamics for 4D Synthesis. In the 4D stage, we apply our novel motion amplification mechanism (Sec. 3.4) after we combined the regular implicit classifier score of the video model $\boldsymbol{\delta}^{\text{vid}}_{\text{cls}}$ with the additional negative prompting term. Therefore, let us define

$$\hat{\boldsymbol{\delta}}^{\text{vid}}_{\text{cls+neg}} := \boldsymbol{\delta}^{\text{vid}}_{\text{cls}} + \omega_{\text{neg}} \left(\hat{\boldsymbol{\epsilon}}^{\text{vid}}(\mathbf{Z}, t) - \hat{\boldsymbol{\epsilon}}^{\text{vid}}(\mathbf{Z}, v^{\text{neg}}, t) \right). \tag{28}$$

We can now write out the entire gradient used to update the deformation field $\boldsymbol{\Phi}$ in AYG's main 4D stage in a concise manner as

$$\begin{aligned} \nabla_{\boldsymbol{\Phi}} \mathcal{L}^{\text{AYG}}_{\text{4D-stage}} = \mathbb{E}_{t, \boldsymbol{\epsilon}^{\text{vid}}, \boldsymbol{\epsilon}^{\text{im}}} \Bigg[w(t) \Bigg\{ \omega_{\text{vid}} \Bigg(\Big\langle \hat{\boldsymbol{\delta}}^{\text{vid}}_{\text{cls+neg}} \Big\rangle_{\text{frame-avg.}} + \omega_{\text{ma}} \Bigg[\hat{\boldsymbol{\delta}}^{\text{vid}}_{\text{cls+neg}} - \Big\langle \hat{\boldsymbol{\delta}}^{\text{vid}}_{\text{cls+neg}} \Big\rangle_{\text{frame-avg.}} \Bigg] \Bigg) + \omega_{\text{im}} \boldsymbol{\delta}^{\text{im}}_{\text{cls}} \Bigg\} \frac{\partial \mathbf{X}}{\partial \boldsymbol{\Phi}} \Bigg] \\ + \lambda_{\text{JSD}} \nabla_{\boldsymbol{\Phi}} \mathcal{L}_{\text{JSD-Reg.}} + \lambda_{\text{Rigidity}} \nabla_{\boldsymbol{\Phi}} \mathcal{L}_{\text{Rigidity-Reg.}}, \end{aligned} \tag{29}$$

where we inserted the motion amplification mechanism with scale ω_{ma}, negative prompting with weight ω_{neg}, as well as the additional regularizers. We only do negative prompting for the video model, using *"low motion, static statue, not moving, no motion"* as the negative prompt v^{neg}.

Use of Latent Diffusion Models. We would like to remind the reader that in the above derivations we have not explicitly included our diffusion models' encoders. All employed diffusion models are *latent* diffusion models [70, 86], see Appendix C.7. In practice, all score distillation gradients are calculated in latent space on encodings of the 4D scene renderings, and the gradients are backpropagated through the models' encoders before further backpropagated through the differentiable rendering process into the 4D scene representation. However, AYG's synthesis framework is general.

E. Experiment Details

E.1. Video Diffusion Model Training

We train a text-to-video diffusion model, following VideoLDM [7]. VideoLDM is a latent diffusion model that builds on Stable Diffusion [70] as text-to-image backbone model and fine-tunes it on text-video datasets by inserting additional temporal layers into the architecture. For details, see Blattmann et al. [7]. Here, we add conditioning on the frames-per-second (fps) frame rate, following [78, 79], scale the number of frames from 8 to 16 frames, and train the model on a larger dataset. Note that our video model was trained only for the purpose of 4D score distillation and is not meant to be a standalone video generation system, which would require additional upsampling and interpolation modules.

Datasets. Our training initially utilizes the WebVid-10M dataset [4], which comprises 10 million text-video pairs. In our analysis of the WebVid dataset, we noted the presence of many slow-motion videos. To address this, we implement a simple filtering algorithm to exclude such data. Initially, we calculate optical flow at 2 fps utilizing the iterative Lucas-Kanade method [49]. Following this, we filter out any videos where the average optical flow magnitude falls below a specified threshold. Specifically, we set this threshold to be 10. We further filter data with aesthetic score [73] lower than 4.0 and train our model for 100K iterations. To enhance the model's generalization capabilities, we incorporate the HDVG-130M dataset [90] in a subsequent fine-tuning phase for an additional 100K iterations. During fine-tuning, we specifically exclude data containing keywords such as 'talk', 'chat', 'game', 'screenshot', 'newspaper', 'screen', and 'microphone' in text captions. These keywords often indicate video recordings of interviews showing little scene dynamics or computer games with very unusual, non-photorealistic visual appearance. Since there are many such videos in the dataset and we are not interested in generating such videos or 4D scenes in this paper, we remove these videos from the training data. We also increase the optical flow slow motion filtering [49] threshold to 20. Moreover, since WebVid-10M is much smaller than HDVG-130M, but provides valuable training data with high-quality text annotations by human annotators, we slightly oversample the filtered WebVid-10M data. Specifically, we do not sample training data as if we just merged WebVid and HDVG and then randomly drew samples from the resulting dataset to form training batches. Instead, we sample in such a way as if we tripled the filtered

Table 3. Hyperparameters for the first stage (3D synthesis).

bs per GPU	# GPUs	# renders	lr_{position}	lr_{rgb}	lr_{sh}	lr_{opacity}	lr_{scaling}	ω_{vg}	ω_{neg}	ω_{3D}	ω_{im}
4	1	16	0.0002	0.01	0.0005	0.05	0.005	3.0	0.8	1.6	0.4

Table 4. Hyperparameters for the second stage (dynamic 4D synthesis).

bs per GPU	# GPUs	# renders	$lr_{\mathbf{\Phi}}$	num. hidden $\mathbf{\Phi}$	num. layers $\mathbf{\Phi}$	$\lambda_{\text{Rigidity}}$	λ_{JSD}	ω_{ma}	ω_{neg}	ω_{vid}	ω_{im}
1	4	64	0.001	128	5	100.0	30.0	24.0	0.8	1.0	1.0

WebVid data and only then merged with the filtered HDVG and sampled from the full training dataset randomly to construct training batches. We train the model on 128 NVIDIA-80G-A100 GPUs, utilizing fp16 precision and a gradient accumulation strategy with a batch size of 4 per-GPU. The total batch size, distributed across all GPUs, is 2048, achieved after complete gradient accumulation.

Frames-per-second (fps) Conditioning. We condition the model on the frames-per-second (fps) frame rate (by using cross-attention to the corresponding sinusoidal embedding). During training, the fps values are randomly sampled within a range of 2 to 16. Given that lower fps often indicates more motion and presents a greater learning challenge, we sample low fps more often. Specifically, we sample the training video fps based on the probability distribution $p(\text{fps}) \sim 1/\text{fps}^c$, where c is a hyperparameter. We choose $c = 1.6$ in our experiments.

Otherwise, our training exactly follows VideoLDM [7]. Samples from AYG's latent video diffusion model are shown in Figs. 15 to 17 in Appendix G.4 as well as in the supplementary video `ayg_new_video_model.mp4`, demonstrating the effect of the fps conditioning.

It is worth noting that we also explored training a larger latent video diffusion model with more parameters, improved temporal attention layers and additional cross-frame attention [38, 94] for higher performance. While this model indeed was able to generate noticeably higher quality videos, this did not translate into improved 4D distillation performance when used during score distillation. It would be valuable to study the effect of different video diffusion models during text-to-4D synthesis via score distillation in more detail. Based on these observations, we kept using the more memory efficient model described in this section above, which more directly follows the architecture from Blattmann et al. [7], apart from the fps conditioning.

E.2. Text-to-4D Hyperparameters

Stage 1. Table 3 summarizes the hyperparameters for the 3D optimization stage. bs denotes the batch size, and lr_{position}, lr_{rgb}, lr_{sh}, lr_{opacity} and lr_{scaling} denote the learning rates for the 3D Gaussians' position, color, spherical harmonics, opacity and scaling parameters, respectively. We use a single GPU for the optimization in the first 3D synthesis stage and a batch size of 4. This means that we use 4 independent sets of 4 images each, given to MVDream, which takes sets of 4 images as input. Each set of 4 images consists of renders with the correct relative camera angles for the multiview diffusion model MVDream. The 4 sets of images are then fed to MVDream, and all images are additionally fed to the regular text-to-image diffusion model (Stable Diffusion). Hence, in each optimization step, 16 different images are rendered in total from the 3D scene. For lr_{position}, we start from 0.001 and decay to the specified value by the 500-th iteration. We note that we followed the learning rate schedules used by DreamGaussian [83]. ω_{vg} and ω_{neg} denote the view guidance scale and negative prompt guidance scale, respectively. ω_{3D} is the weighting factor for the classifier score distillation term from MVDream [76] (p_{3D}) and ω_{im} is the weighting factor for the classifier score distillation term from the image diffusion model (p_{im}) (see Section 3.2). We sample the diffusion time t in the range $[0.02, 0.98]$ at the start of optimization and decay the range to $[0.02, 0.5]$ by the 6,000-th iteration for the image diffusion model (p_{im}). For MVDream [76] (p_{3D}), we directly follow their schedule which samples t from $[0.98, 0.98]$ at the start of optimization and decay the range to $[0.02, 0.5]$ by the 8,000-th iteration. We randomly choose black or white background during training. We run 10,000 optimization steps on average for this stage.

Stage 2. Table 4 summarizes the hyperparameters for the dynamic 4D optimization stage. bs again denotes the batch size, and $lr_{\mathbf{\Phi}}$, "num. hidden $\mathbf{\Phi}$" and "num. layers $\mathbf{\Phi}$" denote the learning rate, number of hidden units and number of layers for the deformation MLP, respectively. $\lambda_{\text{Rigidity}}$ and λ_{JSD} denote the weighting terms for the rigidity regularization (Appendix C.4) and the JSD-based dynamic 3D Gaussian distribution regularization (Appendix C.5). ω_{ma} specifies the motion amplification scale, while ω_{neg} is the negative prompt guidance scale for the video diffusion model. ω_{vid} denotes the weighting term for the video DM's (p_{vid}) classifier score distillation term and ω_{im} denotes the weighting term for the image model's (p_{im}) classifier

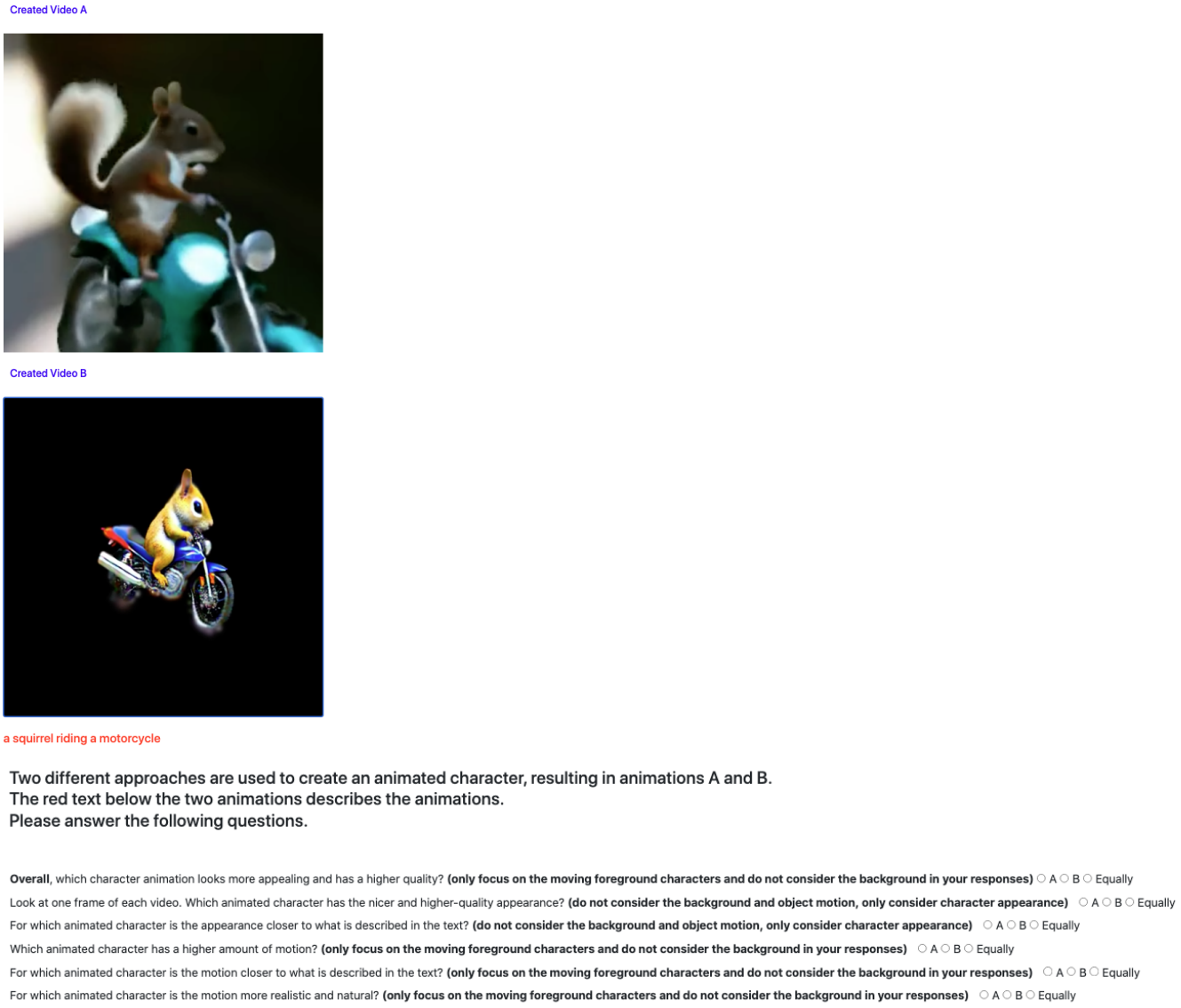

Figure 9. Screenshot of instructions provided to participants of the user studies for comparing AYG and MAV3D [79] as well as for the ablation studies.

score distillation term (see Section 3.2). Here, we used 4 GPUs per optimization with a batch size of 1 on each GPU. This means that on each GPU a single batch of 16 consecutive 2D images is rendered, consistent with the video diffusion model, which models 16-frame videos. Recall that, as discussed in Sec. 3.2, only four of those frames are also given to the regular text-to-image diffusion model. Hence, in each optimization step, 64 different images are rendered in total from the dynamic 4D scene. We sample the diffusion time t in the range $[0.02, 0.98]$ throughout the optimization process for the second stage. We also run 10,000 optimization steps on average for this stage.

E.3. Evaluation Prompts

For the baseline comparison to MAV3D [79], we used all the 28 prompts from MAV3D's project page:

"An alien playing the piano."; "Shark swimming in the desert."; "A dog wearing a Superhero outfit with red cape flying through the sky."; "A monkey eating a candy bar."; "A squirrel DJing."; "A cat singing."; "A bear driving a car."; "Chihuahua running on the grass."; "A human skeleton drinking wine."; "A yorkie dog eating a donut"; "A baby panda eating ice cream"; "A kangaroo cooking a meal."; "A humanoid robot playing the violin."; "A squirrel playing the saxophone.";

Table 5. **R-Precision comparison to MAV3D [79]** with the 300 text prompts also used by Singer et al. [78] and Singer et al. [79].

Method	AYG (*ours*) 3D-stage	AYG (*ours*) 4D-stage	MAV3D [79] 3D-stage	MAV3D [79] 4D-stage
R-Precision ↑	82.2	81.7	82.4	83.7

"An octopus is underwater."; "A silver humanoid robot flipping a coin."; "A goat drinking beer."; "A squirrel playing on a swing set."; "A panda playing on a swing set."; "A crocodile playing a drum set."; "A squirrel riding a motorcycle."; "3D rendering of a fox playing videogame."; "A dog riding a skateboard."; "An emoji of a baby panda reading a book."; "Clown fish swimming through the coral reef."; "A space shuttle launching."; "A corgi playing with a ball."; "A panda dancing."

For the ablation study, we selected the following 30 text prompts:

"A cat singing."; "A corgi playing with a ball."; "A cow running fast."; "A dog wearing a Superhero outfit with red cape flying through the sky."; "A fox dressed in a suit dancing."; "A monkey eating a candy bar."; "A monkey is playing bass guitar."; "A panda dancing."; "A panda surfing a wave."; "A pig running fast."; "A purple unicorn flying."; "A space shuttle launching."; "A squirrel DJing."; "A squirrel playing on a swing set."; "A squirrel playing the saxophone."; "A squirrel riding a motorcycle."; "A storm trooper walking forward and vacuuming."; "An alien playing the piano."; "an astronaut is playing the electric guitar."; "An astronaut riding a horse."; "An astronaut riding motorcycle."; "A panda reading a book."; "Beer pouring into a glass."; "Chihuahua running."; "Clown fish swimming."; "A dog riding a skateboard."; "Flying dragon on fire."; "Intricate butterfly flutters its wings."; "Waves crashing against a lighthous."; "Wood on fire."

E.4. User Study Details

We conducted human evaluations (user studies) through Amazon Mechanical Turk to assess the quality of our generated 4D scenes, comparing them with MAV3D [79] and performing ablation studies.

For the MAV3D comparison, we used the 28 rendered videos from MAV3D's project page (https://make-a-video3d.github.io/) and compared them against our model (AYG) using identical text prompts (see Appendix E.3 above). We rendered our dynamic 4D scenes from similar camera perspectives and created similar videos. We then asked the participants to compare the two videos with respect to 6 different categories and indicate preference for one of the methods with an option to vote for 'equally good' in a non-forced-choice format. The 6 categories measure overall quality, 3D appearance and 3D text alignment, as well as motion amount, motion text alignment and motion realism (see questions in Fig. 9).

In the ablation studies, we proceeded similarly. We showed participants 4D animations for 30 text prompts (see Appendix E.3 above) generated by the full AYG model and by the modified, ablated models. Again, participants were asked to choose the more favorable 4D character from each pair, with an option to vote for 'equally good'.

For a visual reference, see Fig. 9 for a screenshot of the evaluation interface. In all user studies, the order of video pairs (A-B) was randomized for each question. Note that since MAV3D uses an extra background model, while AYG does not, we asked participants to focus only on the moving foreground characters and to not consider the background in their responses. In all user studies, each video pair was evaluated by seven participants, totaling 196 responses for the MAV3D comparison and 210 for each setup in the ablation study. We selected participants based on specific criteria: they had to be from English-speaking countries, have an acceptance rate above 95%, and have completed over 1000 approved tasks on the platform.

F. Additional Quantitative Results

F.1. Comparisons to MAV3D and R-Precision Evaluation

The important MAV3D baseline [79] did not release any code or models and its 4D score distillation leverages the large-scale Make-A-Video [78] text-to-video diffusion model, which is also not available publicly. This makes comparisons to MAV3D somewhat difficult and this is why we compared to MAV3D by using the available results on their project page. As reported in the main text, we performed a user study comparing all their generated 4D scenes with AYG's generated scenes with the same text prompts. AYG outperforms MAV3D in our user study in all categories (see Table 1 in main text). Moreover, our comparisons to MAV3D do not leverage any fine-tuning as discussed in Appendix C.9. With this fine-tuning, our quality improvements over MAV3D are even larger, which would likely be reflected in an even higher preference for our 4D scenes in the user study.

Table 6. **Ablation study** by user study on synthesized 4D scenes with 30 text prompts. For each pair of numbers, the left number is the percentage that the full AYG model is preferred and the right number indicates preference percentage for ablated model as described in left column. The numbers do not add up to 100 and the difference is due to users voting "no preference" (table copied here from main paper for extended discussion in Appendix F.2).

Align Your Gaussians (full model)	Overall Quality	3D Appearance	3D Text Alignment	Motion Amount	Motion Text Alignment	Motion Realism
v.s. w/o rigidity regularization	**45.8**/13.3	**43.3**/19.2	**38.3**/15.0	**40.8**/15.0	**42.5**/18.3	**30.8**/26.7
v.s. w/o motion amplifier	**43.3**/23.3	**37.5**/28.3	**30.8**/26.7	**45.8**/10.8	**37.5**/26.7	**33.3**/31.7
v.s. w/o initial 3D stage	**67.5**/15.0	**57.5**/21.7	**64.2**/15.0	**60.8**/21.7	**60.8**/20.8	**59.2**/24.2
v.s. w/o JSD-based regularization	**40.0**/25.0	**40.0**/27.5	**36.7**/27.5	**41.7**/24.2	**39.2**/29.2	**45.0**/24.2
v.s. w/o image DM score in 4D stage	**42.5**/22.5	**39.2**/27.5	**36.7**/25.8	**33.3**/25.9	**37.5**/30.0	27.5/**40.0**
v.s. SDS instead of CSD	**44.2**/35.8	**40.0**/27.5	**35.8**/35.0	**35.0**/27.5	**35.0**/34.2	32.5/**35.8**
v.s. 3D stage w/o MVDream	**66.7**/21.7	**48.3**/34.2	**38.3**/34.2	**41.7**/22.5	**40.0**/27.5	**40.8**/27.5
v.s. 4D stage with MVDream	**50.8**/27.5	**38.3**/34.2	**41.6**/29.2	**39.2**/35.0	**44.2**/30.0	**39.2**/31.7
v.s. video model with only fps 4	**46.7**/15.8	27.5/**36.7**	**30.0**/23.3	**36.7**/30.0	**31.7**/26.7	**32.5**/28.3
v.s. video model with only fps 12	**48.3**/29.2	**30.8**/29.2	**29.2**/28.3	**35.0**/28.3	**35.0**/30.0	**39.2**/26.7
v.s. w/o dynamic cameras	**32.5**/25.0	**32.5**/31.7	**35.0**/33.3	**35.0**/32.5	**35.8**/33.3	**32.5**/25.0
v.s. w/o negative prompting	**44.2**/28.3	**38.3**/32.5	**31.7**/29.2	29.2/**31.6**	**33.3**/30.0	**37.5**/28.3

MAV3D also reports R-Precision [32, 58] in their paper. R-Precision is commonly used in the text-to-3D literature as an evaluation metric. In the R-Precision calculation, 2D renderings of the scene are given to a CLIP [65] image encoder and the CLIP encoding is then used to retrieve the closest text prompt among a set of text prompts used in the evaluation. The R-Precision value is then the top-1 retrieval accuracy, *i.e.*, the percentage of correctly retrieved text prompts (this is, the text prompt which was used to generate the 3D scene is correctly retrieved). However, R-Precision measures only 3D quality (and more specifically the alignment of the 3D appearance with the text prompt) and does not in any way capture dynamics at all. It is therefore a meaningless metric to evaluate *dynamic* scenes, which is the focus of our and also MAV3D's work.

Nevertheless, for future reference and simply to follow MAV3D we also provide R-Precision evaluation results in Table 5. We obtained the list of 300 prompts used in MAV3D's R-Precision evaluation by the authors of MAV3D. Note that this list of prompts was originally collected not for the evaluation of synthesized 4D dynamic scenes but for the evaluation of video diffusion models in Make-A-Video [78]. To calculate our R-Precision scores with the 300 text prompts, we used the evaluation protocol from `https://github.com/Seth-Park/comp-t2i-dataset`. Similarly to MAV3D, we evaluated R-Precision both after the initial 3D stage and at random times τ of the dynamic 4D scene after the full 4D stage optimization (MAV3D similarly first optimizes a static 3D scene and only then adds an additional temporal dimension to generate a full dynamic 4D scene). Specifically, for 3D objects, we render with 20 different azimuth angles with a fixed elevation of 15 degree, camera distance of 3 and field-of-view of 40. For dynamic 4D scenes, we sample 20 times τ together with 20 different azimuth angles. We use majority voting over the 20 views as top-1 retrieval results for the R-Precision calculation. The results are shown in Table 5.

We see that the two methods perform on par. Importantly, we do not know the exact evaluation protocol MAV3D used (*e.g.* camera poses used for rendering), and in our experience these details matter and can influence the metric non-negligibly. Hence, considering that the results of the two methods are extremely close, we conclude that the two methods perform approximately similarly with respect to R-Precision. We also see that, for both methods, performance does not meaningfully differ between the 3D and 4D stage. This means that both methods preserve the overall 3D object well when learning dynamics in their main 4D stage.

We would like to stress again that R-Precision is in the end not very useful for the evaluation of dynamic 4D scenes, as it completely misses the important temporal aspect and does not judge scene dynamics and object motion. We believe that user studies are a more meaningful way to evaluate dynamic 4D scenes (also MAV3D performs various user studies in their paper). Recall that in our user studies, we outperform MAV3D on all categories.

F.2. Extended Discussion of Ablation Studies

Here, we provide an extended discussion of our main ablation studies, originally presented in Table 2 in Sec. 4. We have carried out an extensive number of ablations and there is not enough space in the paper's main text to discuss all of them in detail. For convenience, we copied the table here, see Table 6, and we will now discuss the different settings one by one.

Also see the supplementary video `ayg_ablation_study.mp4`, which shows dynamic 4D scenes for all ablations. Note, however, that these 4D scenes were optimized with only 4,000 optimization steps in the second dynamic 4D optimization stage in the interest of efficiency, considering that we had to optimize many 4D scenes for all ablations. The quality of the shown 4D scenes is therefore somewhat lower than that of our main results shown elsewhere in the paper and on the project page.

Full AYG v.s. w/o rigidity regularization. We can see a clear preference for the full AYG model compared to a variant without rigidity regularization. Users strongly prefer the full model in all categories in Table 6. In the supplementary video we see unrealistic distortions of the object for the baseline without rigidity regularization. Such distortions are prevented by the regularization, as expected.

Full AYG v.s. w/o motion amplifier. Users prefer the full AYG variant that leverages the motion amplifier for all categories. The difference in preference is most pronounced in the "Motion Amount" category, which validates that the motion amplifier indeed amplifies motion. In the supplementary video, we can clearly observe enhanced motion compared to the baseline without the motion amplifier.

Full AYG v.s. w/o initial 3D stage. Without the initial 3D stage, simultaneously optimizing the 3D object itself and also distilling motion into the 4D scene results in unstable optimization behavior and no coherent dynamic 4D scenes can be produced. This is visible in the supplementary video and also validated in the user study. The full AYG model with the initial 3D stage is strongly preferred.

Full AYG v.s. w/o JSD-based regularization. We can also observe a clear preference for the full AYG model including the JSD-based regularization of the distribution of the dynamic 3D Gaussians over an ablated model that does not use it. This is visible in all categories in Table 6. In the supplementary video, we can see that without JSD-based regularization the 4D sequences show little motion and only some slow floating of the entire objects can be observed. We hypothesize that this slow global motion represents a local minimum of the video diffusion model, whose gradients are used to learn the dynamics. Falling into this local minimum is prevented through the JSD-based regularization. With JSD-based regularization, more complex, local motion is learnt instead of global translations or object size changes.

Full AYG v.s. w/o image DM score in 4D stage. A central design decision of AYG is the simultaneous use of both a text-to-image diffusion model and a text-to-video diffusion model for score distillation in the main 4D optimization stage. Hence, we compared to a variant that only uses the video model, but we find that the full AYG approach including the image diffusion model score in the 4D stage is preferred, except for "Motion Realism". However, this is a somewhat ambiguous category, as people might have subjective opinions on what constitutes a better motion, such as preferring slow but consistent motion versus large but artifact-prone motion. The full model wins on all other categories. The margin between the full and ablated model is especially large on the 3D appearance and 3D text alignment categories. This is exactly what we expected and why the text-to-image model is used, *i.e.*, to ensure that high visual and 3D quality is maintained while the video model is responsible for the optimization of the temporal dynamics. In line with that, in the supplementary video we can observe some degradation in the 3D quality without the text-to-image diffusion model in the 4D stage. This justifies AYG's design composing a text-to-image and a text-to-video diffusion model for score distillation in the 4D stage.

Full AYG v.s. SDS instead of CSD. We have a somewhat similar observation when replacing classifier score distillation (CSD) with regular score distillation sampling (SDS). The full model is preferred for all categories, except for the more ambiguous "Motion Realism". In the supplementary video we see slightly more distorted 3D objects and less motion when using SDS instead of CSD.

Full AYG v.s. 3D stage w/o MVDream. Another central design decision of AYG is the use of the 3D-aware multiview-image diffusion model MVDream [76] in its first 3D stage. In this ablation, we removed MVDream from the 3D stage and optimized the 3D assets only with the text-to-image Stable Diffusion model [70], and then performed the dynamic 4D distillation. We find that users prefer the full model by a large margin on all categories. In short, AYG is not able to produce high-quality static 3D assets without MVDream, and then also the dynamic 4D optimization struggles because of the poor 3D initialization. It is worth pointing out, however, that there is a rich literature on score distillation of 3D assets using only text-to-image models and arguably some of these techniques could help. As pointed out previously, the initial 3D assets used by AYG in its main 4D stage could potentially also be produced by other methods.

Full AYG v.s. 4D stage with MVDream. The previous ablation makes one wonder whether it would help to also include MVDream in the 4D optimization stage to ensure that geometric and multiview consistency is maintained during 4D optimization at all times τ of the 4D sequences. However, we find that this is not the case. Users prefer the dynamic 4D scenes generated by the regular AYG model over the one that also includes MVDream in the 4D stage for all categories in the table. In the supplementary video, we see that including MVDream in the 4D stage can lead to odd motion patterns or overall reduced motion. We hypothesize that the video diffusion model and the MVDream multiview diffusion model produce

Table 7. **Ablation study on view guidance** by user study on synthesized static 3D scenes from AYG's initial 3D stage. We used 30 text prompts, the same as in the other ablation studies. Numbers are percentages.

Method preference	AYG w/ view guidance preferred	AYG w/o view guidance preferred	Equal preference
Overall Quality	33.3	33.8	32.9
3D Appearance	28.6	29.0	42.4
3D Text Alignment	30.0	34.8	35.2

somewhat conflicting gradients, harming the 4D optimization process and suppressing the learning of smooth 4D dynamics. Hyperparameter tuning and a carefully chosen weighting between the MVDream and video models in the composed score distillation scheme could potentially address this, but we were not able to make this setting work better than the regular AYG approach without MVDream in the 4D stage. Together with the previous ablation, we conclude that MVDream is a crucial component of AYG, but only in its initial 3D stage.

Full AYG v.s. video model with only fps 4. Our newly trained latent video diffusion model is conditioned on the frames-per-second (fps) frame rate and when optimizing dynamic scenes during AYG's 4D stage we sample different fps values. In this ablation, we ask what would happen if we only sampled the low fps $= 4$ value, which corresponds to videos that show a lot of motion, but therefore are less smooth temporally (see the attached supplementary video `ayg_new_video_model.mp4`). We find that the full AYG model that samples different fps conditionings during score distillation is overall preferred, although there is an outlier in the 3D appearance category. However, the varying fps conditionings in AYG primarily aim at producing better motion and not at improving 3D quality, and for the motion categories the main AYG model is generally preferred over the ablated version with only one fps $= 4$ value. Visually, in the attached ablations video we observe slightly lower quality motion, in line with the results from the user study.

Full AYG v.s. video model with only fps 12. Here, we instead only use fps $= 12$, corresponding to videos with less but smoother motion. We again see that the full AYG model is preferred over this ablated variant, this time in all categories including the 3D ones. In the supplementary video we can see significantly reduced motion in the dynamic 4D scenes when using this ablated AYG version.

Note that these observations are in line with Singer et al. [79], who also used an fps-conditioned video diffusion model and different fps during score distillation. Sampling different fps during 4D optimization helps both them and our AYG to produce better dynamic 4D scenes with higher-quality motion.

Full AYG v.s. w/o dynamic cameras. We can also observe the benefit of dynamic cameras. The full AYG model that includes dynamic cameras is preferred in all categories over the ablated model that uses static cameras when rendering the video frames given to the video diffusion model during score distillation in the 4D stage. In the supplementary video, we see that the dynamic 4D sequences generated without dynamic cameras have less motion. This is also consistent with Singer et al. [79], who also observed a similar benefit of dynamic cameras.

Full AYG v.s. w/o negative prompting. Finally, we also studied the effect of negative prompt guidance during the 4D stage. Overall, the main model that includes negative prompting is preferred, although on "Motion Amount" the baseline without the negative prompt guidance is preferred. In the attached video, we observe lower quality dynamics without negative prompting.

Conclusions. Overall, our ablation studies show that all of AYG's components are important for generating high quality dynamic 4D scenes. On "Overall Quality", the main AYG model wins in all ablations with large margins. This justifies AYG's design choices.

F.3. View-guidance Ablation Study

Our main ablation studies focused primarily on the 4D stage and we wanted to study the effects of the different components when learning dynamic 4D scenes, which is the main novelty of our paper. Here, we show an additional ablation study that analyzes the effect of view-guidance (Sec. 3.4), which we used only in AYG's initial 3D stage. We performed a user study using the same prompts and following the exact same protocol as in our other user studies for the other ablations (see Appendix E.4). However, since view-guidance has only been used in the 3D stage, we only asked the users about overall quality, 3D appearance and 3D text alignment of the static 3D scenes synthesized after the initial 3D stage. We showed the users 3D objects generated with and without view guidance and asked them to compare and rate them. The results are shown in Table 7. We see that AYG with or without view-guidance in its 3D stage performs approximately similar according to the user ratings and there is no clear winner in any of the categories.

We nevertheless used view guidance in AYG, as we subjectively found in early experiments that it sometimes helped with overall 3D appearance and led to a small reduction of the Janus-face problem. However, as the results of the user study here demonstrate, view guidance is certainly not one of the crucial components of AYG that make or break synthesis. However, to the best of our knowledge view guidance is a new idea and maybe it can find applications in future work.

G. Additional Qualitative Results—More AYG Samples

Here, we show additional generated dynamic 4D scene samples from AYG. We also refer the reader to our supplementary video `ayg_text_to_4d.mp4`, which shows almost all of our dynamic 4D scene samples. We also share videos generated by AYG's newly trained latent video diffusion model for this work.

G.1. Text-to-4D Samples

In Figs. 10, 11 and 13, we show additional text-to-4D samples from AYG, similar to Fig. 6 in the main paper.

G.2. Autoregressively Extended and Looping Text-to-4D Synthesis

In Fig. 12, we show additional samples from AYG that are autoregressively extended to form longer sequences while changing the text prompt and that return to the initial pose to enable looping animations (similar to Fig. 7 in the main paper). For the first two rows (the *assassin*) in Fig. 12, we use the following sequence of text prompts: "Assassin with sword running fast, portrait, game, unreal, 4K, HD.", "Assassin walking, portrait, game, unreal, 4K, HD." and "Assassin dancing, portrait, game, unreal, 4K, HD.". For the next two rows (the *lion*), we use the following sequence of text prompts: "A lion running fast.", "A lion is jumping." and "A lion is eating.".

For reference, we also provide the prompts used for Fig. 7 in the main paper. For the first two rows (the *bulldog*), we use the following sequence of text prompts: "A bulldog is running fast." and "A bulldog barking loudly". For the next two rows (the *panda*), we use the following sequence of text prompts: "A panda running." and "A panda is boxing and punching.".

G.3. More Comparisons to Make-A-Video3D

In Fig. 14, we show additional visual comparisons to MAV3D [79], similar to Fig. 8 in the main paper.

G.4. Videos Generated by AYG's fps-conditioned Video Diffusion Model

In Figs. 15 to 17, we present videos generated by AYG's latent video diffusion model, showing the effect of the fps conditioning. We also refer to the attached supplementary video `ayg_new_video_model.mp4`, which shows more samples.

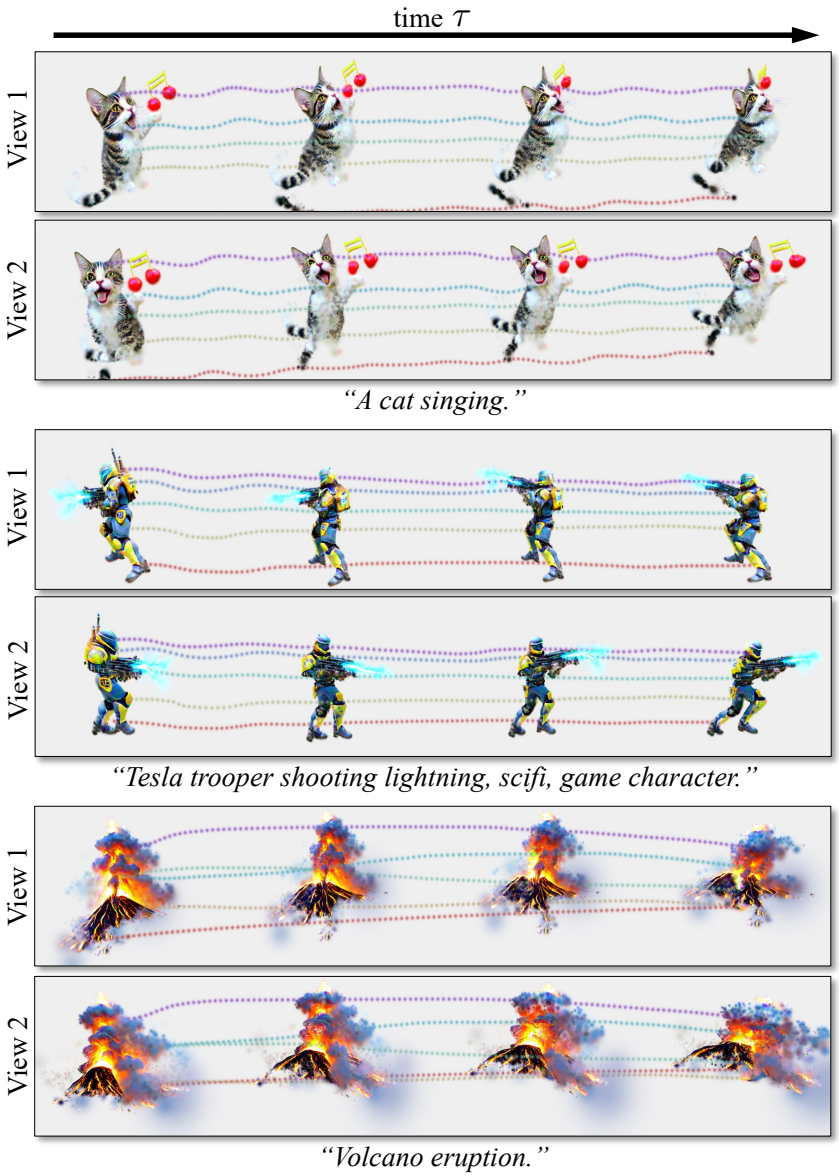

Figure 10. **Text-to-4D synthesis with AYG.** Various samples shown in two views each. Dotted lines denote deformation field dynamics (also see supplementary video `ayg_text_to_4d.mp4`, where the dynamics are much better visible).

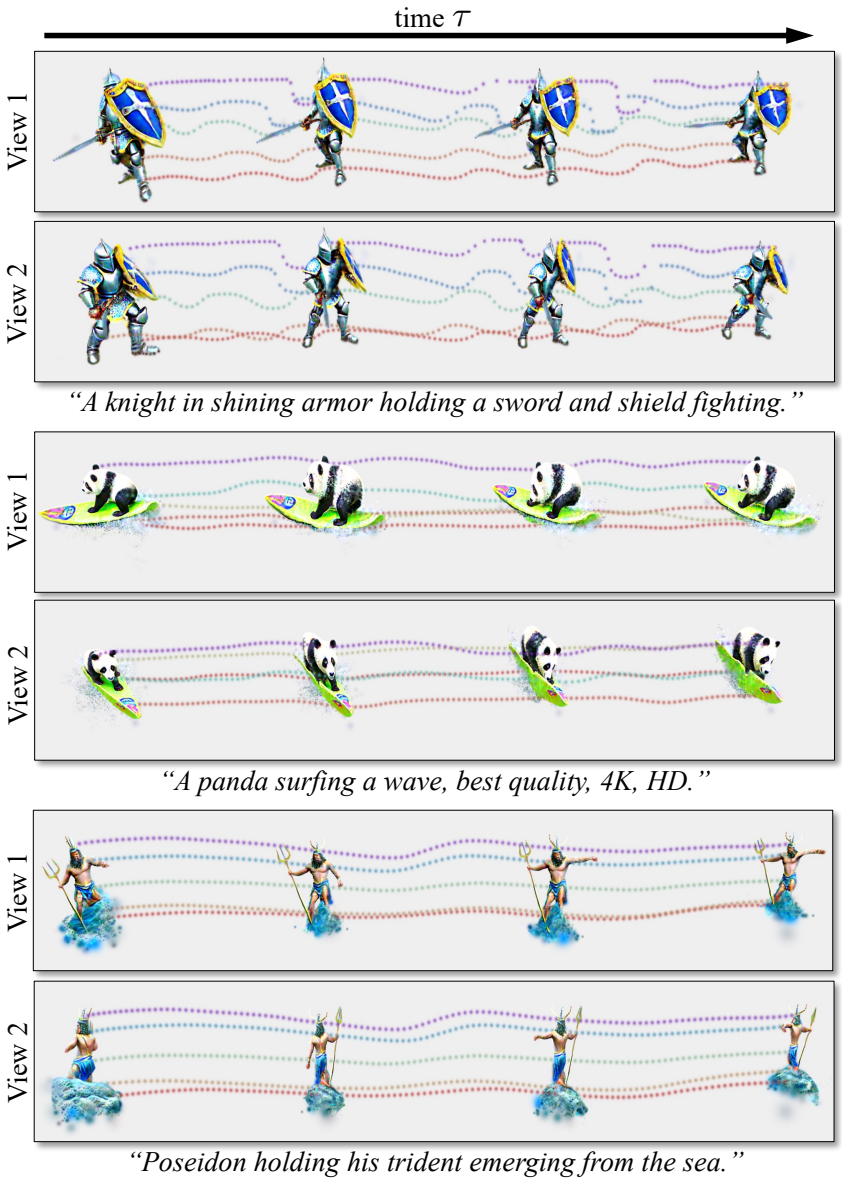

Figure 11. **Text-to-4D synthesis with AYG.** Various samples shown in two views each. Dotted lines denote deformation field dynamics (also see supplementary video `ayg_text_to_4d.mp4`, where the dynamics are much better visible).

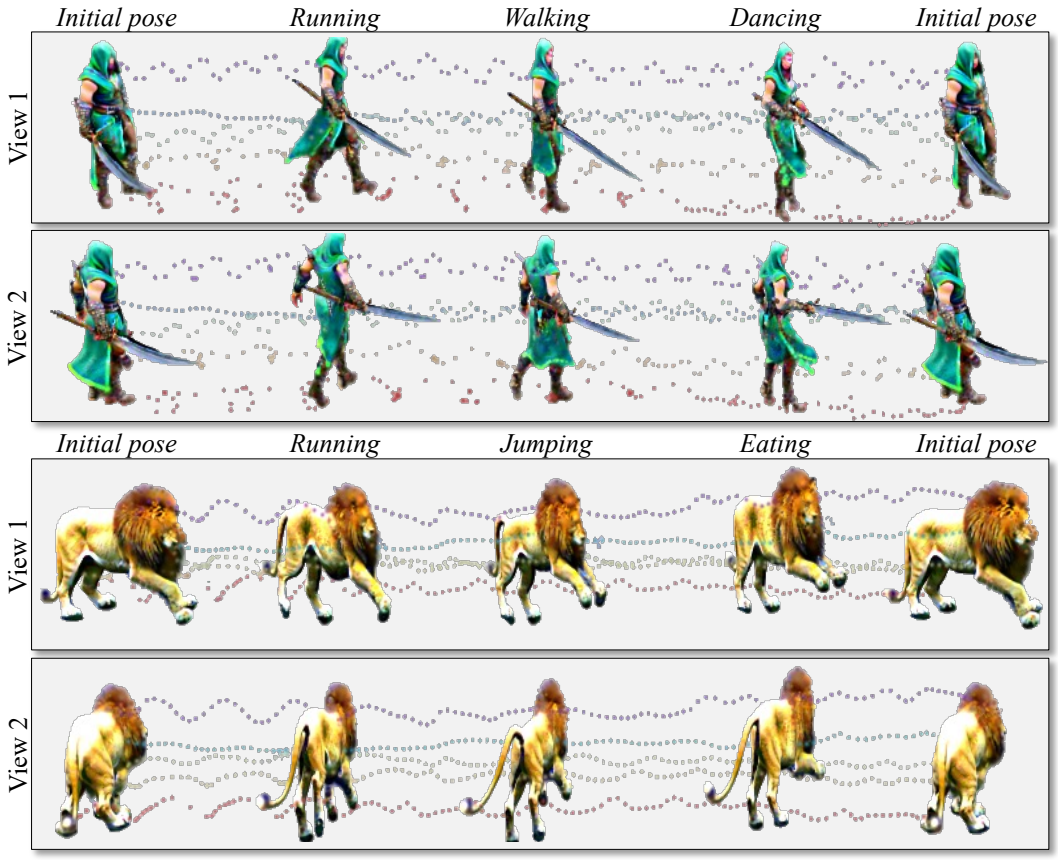

Figure 12. **Autoregressively extended text-to-4D synthesis.** AYG is able to autoregressively extend dynamic 4D sequences, combine sequences with different text-guidance, and create looping animations, returning to the initial pose (also see supplementary video `ayg_text_to_4d.mp4`, where the different actions are much better visible).

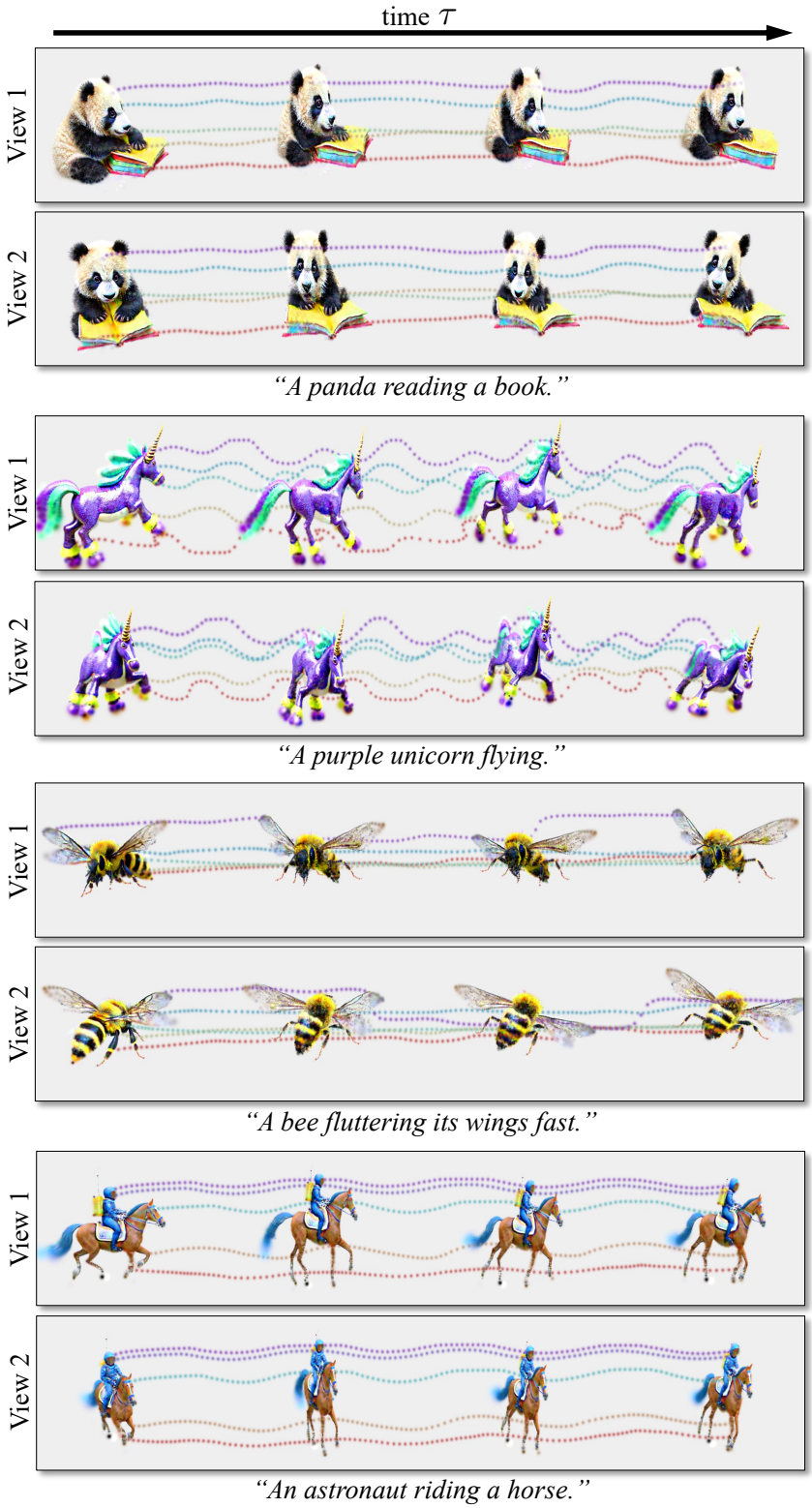

Figure 13. **Text-to-4D synthesis with AYG.** Various samples shown in two views each. Dotted lines denote deformation field dynamics (also see supplementary video `ayg_text_to_4d.mp4`, where the dynamics are much better visible).

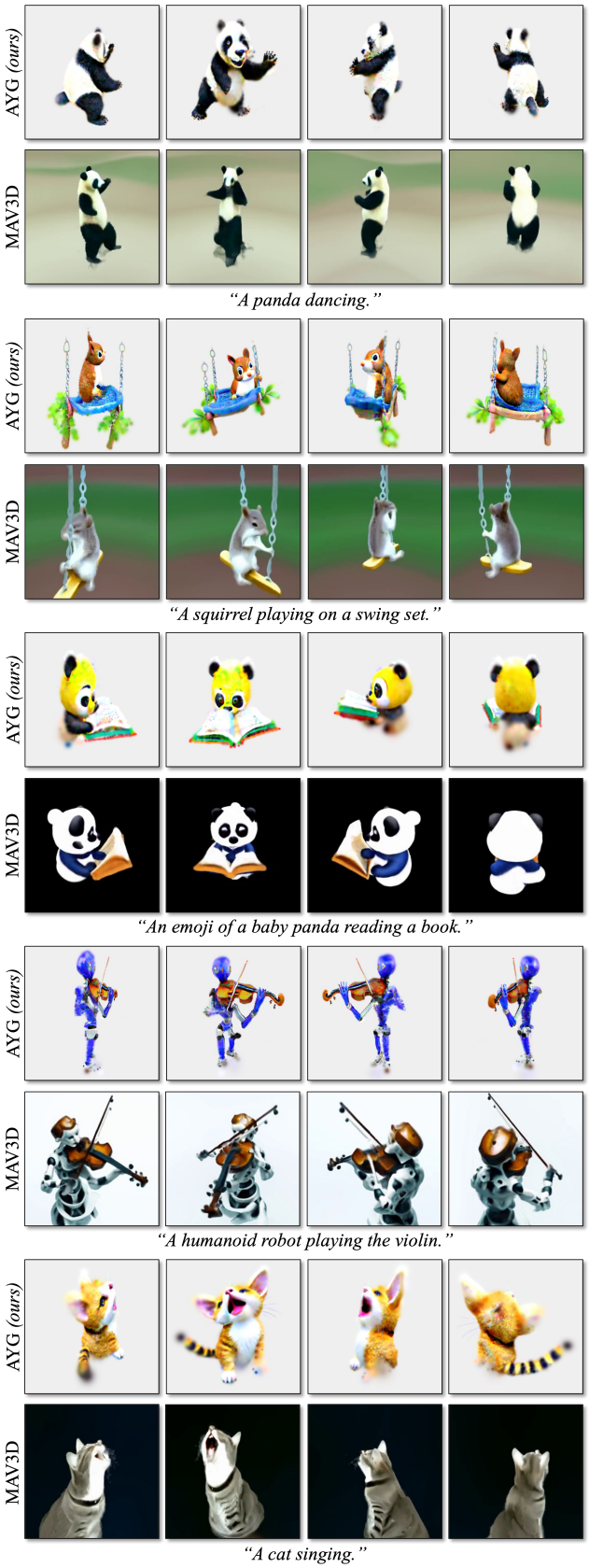

Figure 14. **AYG (*ours*) vs. MAV3D [79].** We show four 4D frames for different times and camera angles (also see supplementary video `ayg_text_to_4d.mp4`, where we also show comparisons to MAV3D and where the dynamics are much better visible).

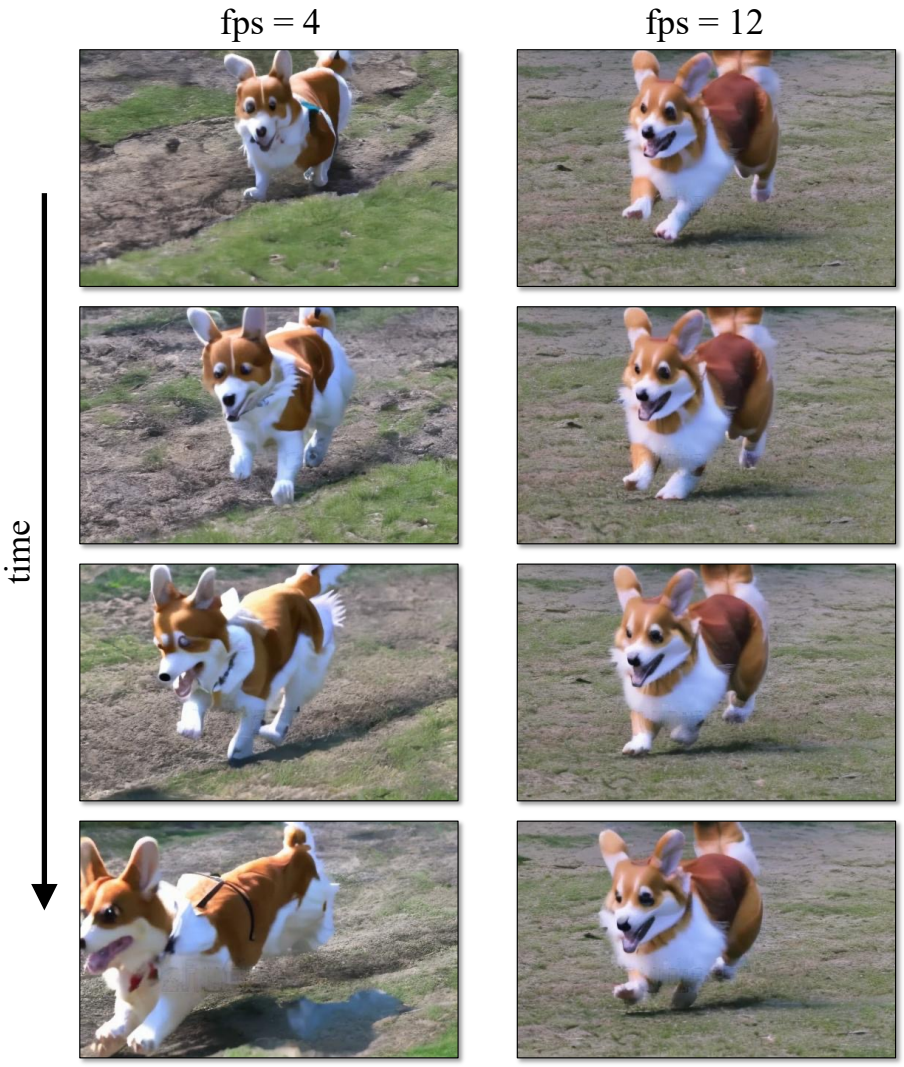

Figure 15. **Two video samples from AYG's newly trained latent text-to-video diffusion model** for the same text prompt *"A corgi running."* but with different fps conditionings fps $=$ 4 and fps $=$ 12. We see that, as expected, conditioning on the lower fps value generates a video with more motion for the same 4 frames (the model synthesizes 16 frames and we show the 1st, the 6th, the 11th, and the 16th frame). Conditioning on the higher fps value results in a video with less motion but good temporal consistency.

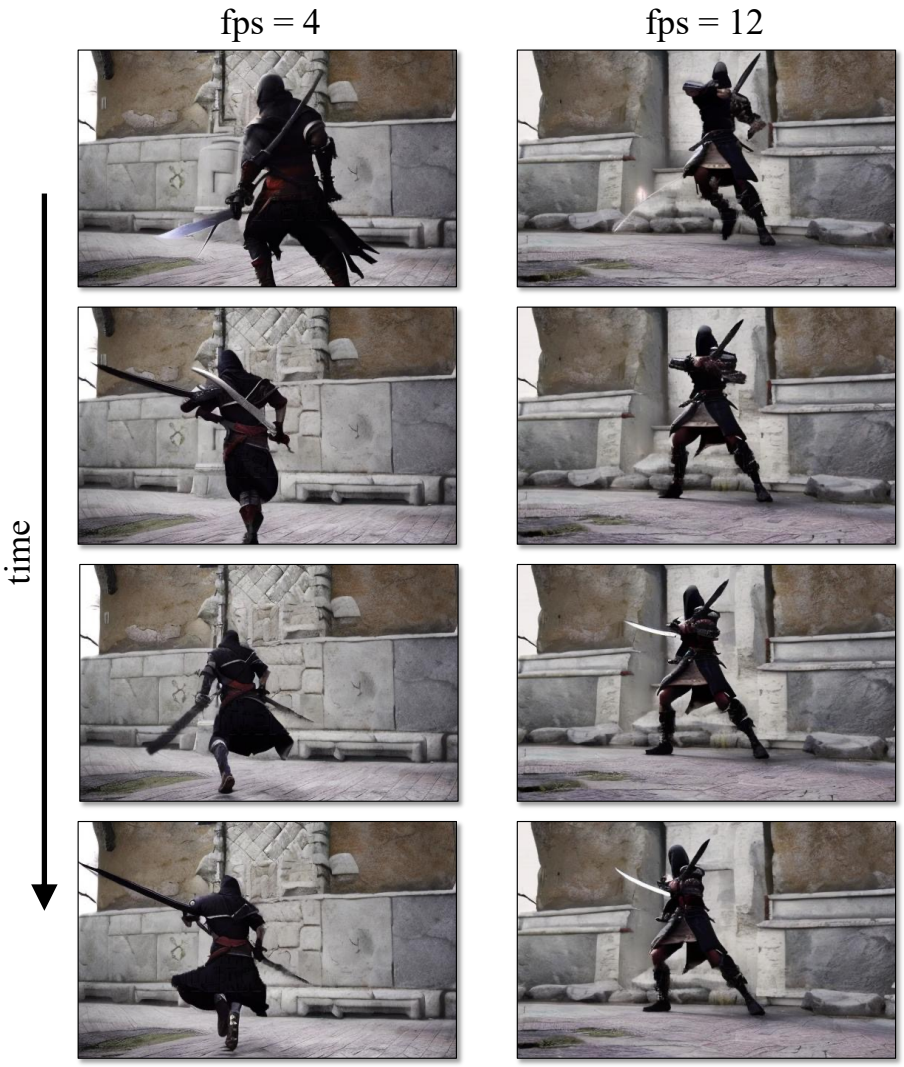

Figure 16. **Two video samples from AYG's newly trained latent text-to-video diffusion model** for the same text prompt *"Assassin with sword running fast, portrait, game, unreal, 4K, HD."* but with different fps conditionings fps $= 4$ and fps $= 12$. We see that, as expected, conditioning on the lower fps value generates a video with more motion for the same 4 frames (the model synthesizes 16 frames and we show the 1st, the 6th, the 11th, and the 16th frame). Conditioning on the higher fps value results in a video with less motion but good temporal consistency.

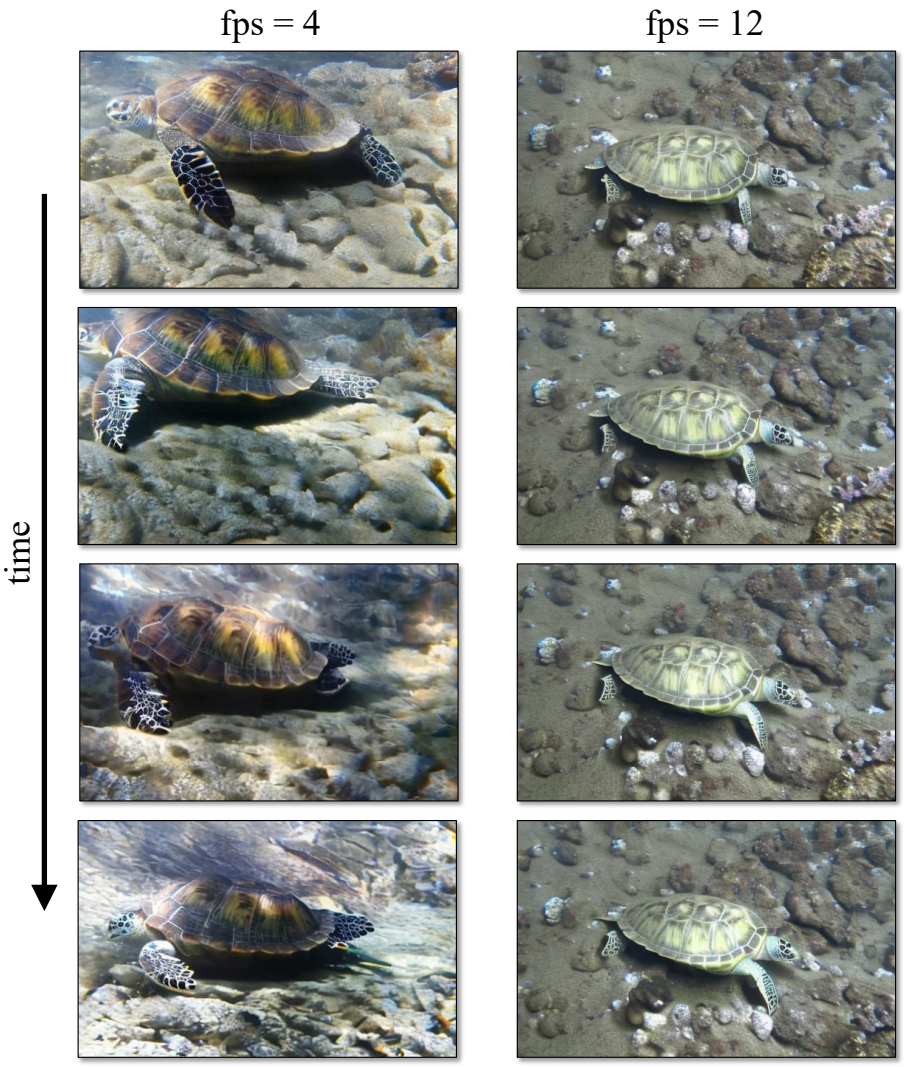

Figure 17. **Two video samples from AYG's newly trained latent text-to-video diffusion model** for the same text prompt *"A turtle swimming."* but with different fps conditionings fps $= 4$ and fps $= 12$. We see that, as expected, conditioning on the lower fps value generates a video with more motion for the same 4 frames (the model synthesizes 16 frames and we show the 1st, the 6th, the 11th, and the 16th frame). Conditioning on the higher fps value results in a video with less motion but good temporal consistency.