arXiv:2504.05296v3 [cs.GR] 17 Mar 2026

Let it Snow! Animating 3D Gaussian Scenes with Dynamic Weather Effects via Physics-Guided Score Distillation

Gal Fiebelman[1] Hadar Averbuch-Elor[2] Sagie Benaim[1]
[1]The Hebrew University of Jerusalem [2]Cornell University

https://galfiebelman.github.io/let-it-snow/



Figure 1. Our *Physics-Guided Score Distillation* framework enables dynamic scene-wide editing of static 3D Gaussian scenes introducing dynamic weather effects such as snowfall, rainfall, fog, and sandstorms. We do so by introducing a unified optimization framework where physics simulation guides Score Distillation to jointly refine the motion prior for photorealism while simultaneously optimizing appearance. Above, we show the effect of snow falling across different viewpoints at increasing timesteps over three different scenes.

Abstract

3D Gaussian Splatting has recently enabled fast and photorealistic reconstruction of static 3D scenes. However, dynamic editing of such scenes remains a significant challenge. We introduce a novel framework, Physics-Guided Score Distillation, to address a fundamental conflict: physics simulation provides a strong motion prior that is insufficient for photorealism, while video-based Score Distillation Sampling (SDS) alone cannot generate coherent motion for complex, multi-particle scenarios. We resolve this through a unified optimization framework where physics simulation guides Score Distillation to jointly refine the motion prior for photorealism while simultaneously optimizing appearance. Specifically, we learn a neural dynamics model that predicts particle motion and appearance, optimized end-to-end via a combined loss integrating Video-SDS for photorealism with our physics-guidance prior. This allows for photorealistic refinements while ensuring the dynamics remain plausible. Our framework enables scene-wide dynamic weather effects, including snowfall, rainfall, fog, and sandstorms, with physically plausible motion. Experiments demonstrate our physics-guided approach significantly outperforms baselines, with ablations confirming this joint refinement is essential for generating coherent, high-fidelity dynamics.

1. Introduction

Dynamic editing of static 3D scenes is typically a manual task that requires extensive knowledge of specialized software. A key challenge lies in creating temporally consistent, scene-wide modifications while maintaining both physical plausibility and photorealistic appearance. While static editing methods can enhance visual appearance, introducing dynamic elements that evolve over time and interact naturally across an entire scene remains largely un-

explored. Recent advances in neural scene representations, particularly 3D Gaussian Splatting (3DGS) [12], have enabled real-time, high-fidelity reconstruction and rendering of static 3D scenes from multi-view images. However, the development of effective approaches for dynamically editing these representations with scene-wide temporal effects presents a significant challenge.

Prior works for static editing of neural representations [2, 6, 16, 25, 29, 33, 40] cannot model temporal evolution or dynamic effects. Physics-based animation approaches [9, 15, 22, 34, 37] provide plausible motion but focus on manipulating existing scene elements; they cannot synthesize photorealistic appearance for new elements, and their motion-only guidance is insufficient for photorealism. Conversely, data-driven 4D generation methods [11, 14, 18, 20, 24, 31, 38] generate photorealistic appearance but struggle with physically plausible motion, often failing in scene-wide, multi-particle scenarios that require continuous particle emission.

Accordingly, we propose a framework for scene-wide, multi-particle dynamic editing that resolves this conflict through *Physics-Guided Score Distillation*. Our key insight is that physics simulation can serve as a powerful motion prior to guide video diffusion optimization, enabling Score Distillation Sampling (SDS) to jointly refine motion for photorealism while synthesizing appearance, rather than learning both from scratch, a task that proves intractable for complex multi-particle scenarios. Rather than rigidly separating motion generation from appearance optimization, we jointly optimize both within a unified framework.

Specifically, we learn a neural dynamics model that predicts particle trajectories and appearance parameters over time, supporting continuous particle emission and interaction throughout the temporal sequence. This model is optimized end-to-end using Video Score Distillation Sampling (Video-SDS), where physics guides the optimization through two complementary mechanisms: (1) the network is conditioned on reference trajectories from Material Point Method (MPM) simulation, providing physically-informed input features, and (2) a physics regularization loss encourages learned motion to remain close to simulated trajectories while permitting learned refinements for photorealism.

Our approach offers several advantages over existing editing methods. By leveraging physics as a guidance prior rather than a fixed constraint, we achieve temporally consistent edits with natural interaction patterns while allowing the diffusion model to refine both motion and appearance for visual quality. Our Video-SDS optimization, guided by physics regularization, ensures that dynamic elements achieve both photorealistic appearance and plausible motion that integrate seamlessly with the static scene.

We demonstrate our approach on scenes from both the MipNeRF 360 [1] and Tanks and Temples [13] datasets, showing its effectiveness across a range of dynamic scene modifications, including snowfall, rainfall, fog, and sandstorms with diverse appearance variations. Extensive experiments demonstrate that our physics-guided approach significantly outperforms existing methods in both motion plausibility and visual quality. Our ablations confirm that when motion is fixed without physics-guided refinement, Video-SDS cannot properly optimize appearance demonstrating that motion refinement through our joint optimization is essential for achieving photorealistic results. Without physics guidance, Video-SDS alone cannot reliably learn both motion and appearance for complex multi-particle scenarios with continuous particle emission. To summarize, our main contributions are:

- A *Physics-Guided Score Distillation* framework that leverages physics simulation as a motion prior to guide video diffusion optimization, enabling joint learning of motion and appearance for scene-wide, multi-particle dynamic editing.
- A neural dynamics model optimized through a combined loss integrating Video-SDS for photorealism with physics regularization for motion plausibility, supporting continuous particle emission over time.
- State-of-the-art results across diverse dynamic weather effects, with ablations confirming that physics guidance is essential for coherent multi-particle dynamics.

2. Related Work

Score Distillation for 4D Generation and Editing. Score Distillation Sampling (SDS) [21] revolutionized 3D content generation by leveraging 2D diffusion priors to optimize 3D representations. This concept was extended to the temporal domain with Video Score Distillation Sampling (Video-SDS), which uses supervision from text-to-video diffusion models [26] to optimize dynamic 3D content. Building on these foundations, recent 4D generation approaches [11, 18, 24, 38], works like Gaussians2Life [31], and 4D editing methods [14, 20] enable text-guided editing of dynamic scenes. However, while these data-driven approaches generate photorealistic appearance, they struggle with physically plausible motion. These methods operate on fixed Gaussian sets, making them unsuitable for weather effects that require continuous particle emission, accumulation, and removal throughout the temporal sequence.

Static 3D Scene Editing. Recent advances in static neural scene editing have enabled text-driven modifications of 3D representations. Methods such as [6, 19, 25, 40] leverage diffusion models to modify rendered content. For Gaussian-based scenes specifically, GaussianEditor [29] and GaussCtrl [33] enable text-guided editing. Environmental editing methods like ClimateNeRF [16] enable the addition of static weather effects to neural radiance fields. The key limita-

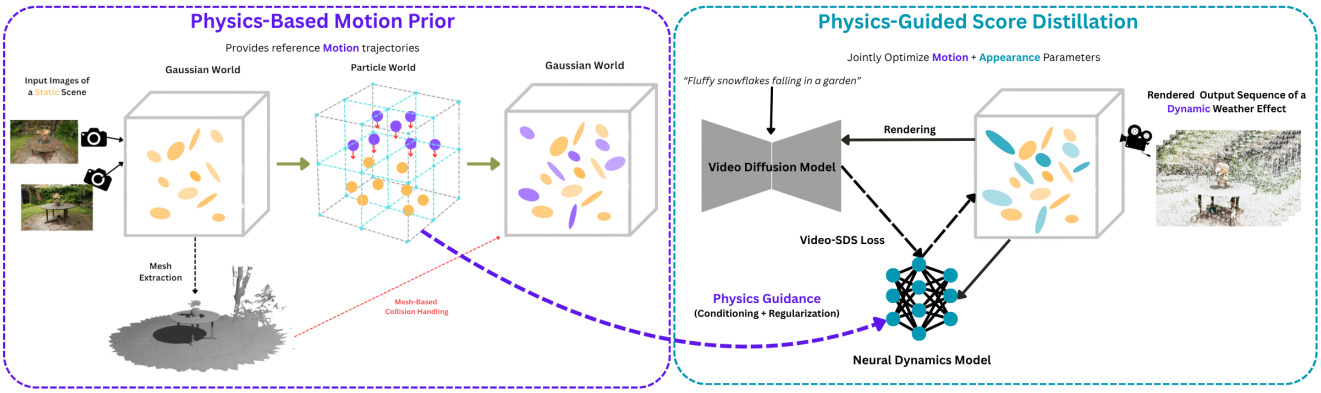


Figure 2. **Overview**. Left: Physics-Based Motion Prior. Given multi-view images of a static scene, we reconstruct the scene using 3DGS, map the static Gaussians to a particle representation, introduce dynamic particles, and simulate their motion using the Material Point Method (MPM). We then map these particles back to the Gaussian world and refine scene interactions with our mesh-based collision handling technique. The simulation provides reference motion trajectories that serve as guidance prior. Right: Physics-Guided Score Distillation. Our Neural Dynamics Model is conditioned on reference trajectories from physics simulation and optimized through Video-SDS with physics regularization losses. This joint optimization refines motion for photorealism while synthesizing appearance, maintaining physical plausibility through physics guidance. This produces photorealistic dynamic weather effects with continuous particle emission that integrate seamlessly with the static scene.

tion across these approaches is their inability to incorporate temporal dynamics; they either modify existing elements or introduce new static components, but cannot create scene-wide elements that evolve naturally over time.

Physics-Based Animation. Physics-based methods integrate simulation with neural representations, providing strong motion priors through physically accurate dynamics. Methods like PhysGaussian [34], PAC-NeRF [15], and DreamPhysics [9] animate existing scene elements through simulation. Gaussian Splashing [4] incorporates Position-Based Dynamics for fluid simulation. RainyGS [3] synthesizes rain effects through physics-based particle systems, however it can not generalize to additional weather effects. While these approaches generate physically-plausible motion, their guidance is derived purely from physics simulation. This motion-only prior is insufficient for photorealism, as these methods cannot synthesize the complex photorealistic appearance for newly introduced dynamic elements, a key limitation our joint optimization framework addresses.

3. Method

In this section we introduce our *Physics-Guided Score Distillation* framework for editing 3D Gaussian Splatting scenes scene-wide, multi-particle edits. As illustrated in Fig. 2, our approach consists of two components: First, we use Material Point Method (MPM) simulation to generate physically plausible reference trajectories for dynamic particles (Sec. 3.2). These trajectories serve as a guidance prior rather than fixed motion. Second, we learn a recurrent neural dynamics model that predicts dynamic Gaussian evolution over time, optimized through Physics-Guided Score Distillation (Sec. 3.3), a joint loss combining Video Score Distillation Sampling (Video-SDS) for photorealistic refinement of both motion and appearance with physics regularization for motion plausibility. We begin by providing a brief background about the foundational techniques (Sec. 3.1) that form the basis of our approach. See Appendix B.1 for implementation details.

3.1. Preliminaries

3D Gaussian Splatting. 3D Gaussian Splatting (3DGS) [12] represents scenes using a set of 3D anisotropic Gaussian kernels. Each Gaussian kernel $\mathcal{G}_i$ is defined by its center position $\mathbf{x}_i$, opacity σ_i, covariance matrix $\mathbf{\Sigma}_i$ which is decomposed into a scaling factor $\mathbf{S}_i \in \mathbb{R}^3$ and a rotation factor $\mathbf{R}_i \in \mathbb{R}^4$, and spherical harmonic coefficients $\mathbf{C}_i$ for view-dependent appearance. The color $\mathbf{C}$ of a pixel is computed by alpha-blending these 3D Gaussians when projected to the image plane:

$$\mathbf{C} = \sum_{i=1}^{N} T_i \alpha_i \mathbf{C}_i, \text{ with } \mathrm{T_i} = \prod_{\mathrm{j}=1}^{\mathrm{i}-1}(1-\alpha_\mathrm{j}), \quad (1)$$

where N is the set of depth-sorted Gaussian kernels affecting the pixel, and C_i and α_i represents the color and density of this point computed by a 3D Gaussian G with covariance $\mathbf{\Sigma}$ and opacity σ.

Material Point Method. The Material Point Method (MPM) [10, 27] is a numerical simulation technique for continuum dynamics. In MPM, a continuum is represented by a set of particles in a grid-based space. Following PhysGaussian [34], particles' positions are updated over time:

$$\mathbf{x}_i(t) = \Delta(\mathbf{x}_i, t) \quad (2)$$

where $\Delta(\cdot, t)$ updates the particle position. MPM is utilized for its ability to efficiently handle both particle dynamics and physical interactions.

Video Score Distillation Sampling. Score Distillation Sampling (SDS) [21] distills pre-trained 2D image diffusion models to optimize 3D representations. Video Score Distillation Sampling (Video-SDS) [26] extends this to pre-trained text-to-video diffusion model for dynamic 3D generation. Given a camera trajectory $\mathbf{r}(t)$, Video-SDS optimizes the rendered 3D video $V_{\mathbf{r}(t)}$ with predicted noise $\hat{\epsilon}_{\text{V}}$:

$$\nabla_\theta \mathcal{L}_{\text{Video-SDS}}(\theta) \triangleq \mathbb{E}\left[\omega(\mu)\left(\hat{\epsilon}_{\text{V}}(V_{\mathbf{r}(t)}; \mu, y) - \epsilon\right)\frac{\partial V_{\mathbf{r}(t)}}{\partial \theta}\right], \tag{3}$$

where μ is noise timestep and θ are optimizable parameters.

3.2. Physics-Based Motion Prior

We use MPM simulation to generate reference trajectories that serve as a guidance prior for our optimization. A static scene is first reconstructed from multi-view images using 3DGS to obtain a set of 3D Gaussians $\{\mathcal{G}_i\} = \{\mathbf{x}_i, \sigma_i, \mathbf{S}_i, \mathbf{R}_i, \mathbf{C}_i\}$. We extract a mesh representation of the scene and map the static Gaussians to a particle-based representation suitable for MPM simulation. These static particles act as obstacles in the simulation domain.

Dynamic particles representing weather effects are then introduced with appropriate physical parameters (emission region, rate, initial velocity, material properties). MPM simulation computes the motion trajectories $\{\mathbf{x}_p^{\text{sim}}(t), \mathbf{v}_p^{\text{sim}}(t)\}$ for these dynamic particles. For computational efficiency, we implement active particle tracking:

$$a_i^{t+1} = \begin{cases} 0, & \text{if } \|\boldsymbol{p}_i^{t+1} - \boldsymbol{p}_i^t\| < \delta \text{ or } \boldsymbol{p}_i^{t+1} \notin \Omega \\ a_i^t, & \text{otherwise} \end{cases} \tag{4}$$

where a_i^t represents the active state of particle i at time t, δ is a movement threshold, and Ω is the valid simulation domain. This enables the simulation of thousands of particles across timesteps by efficiently removing particles from active simulation when they come to rest after collision or leave the bounds of interest.

Due to the coarse grid resolution of MPM, we refine particle-scene interactions through mesh-based collision handling. For snow, we project particles onto mesh surfaces and interpolate with nearby surface Gaussians to create natural accumulation. For rain, we implement a 3D wetness grid that tracks moisture with Gaussian smoothing and temporal decay. For sandstorms, we displace particles along surface normals with anisotropic scaling to create thin accumulations. When particles become inactive ($a_i^t = 1$ to $a_i^{t+1} = 0$), their final positions and rotations are recorded as collision states. These simulated trajectories provide physically plausible reference motion that guides our neural dynamics model. See Appendix B.2 and B.3 for additional collision handling and simulation details.

3.3. Physics-Guided Score Distillation

Rather than treating physics simulation as fixed motion, we use it as a guidance prior to optimize a neural dynamics model through Video-SDS. For each dynamic particle from the MPM simulation, we create a corresponding dynamic Gaussian kernel $\mathcal{G}_g(t) = \{\mathbf{x}_g^{\text{init}}(t), \mathbf{R}_g^{\text{init}}, \mathbf{A}_g^{\text{active}}, \mathbf{A}_g^{\text{collided}}, \mathbf{v}_g^{\text{init}}(t)\}$, where $\mathbf{v}_g^{\text{init}}(t)$ is the velocity of each gaussian that is provided along the position $\mathbf{x}_g^{\text{init}}(t)$ directly from the motion simulation. Each Gaussian also maintains state-dependent appearance parameters based on its physical state determined by the physics simulation: $\mathcal{A}_g^{\text{active}} = \{sigma_g^{\text{active}}, \mathbf{S}_g^{\text{active}}, \mathbf{C}_g^{\text{active}}\}$ for actively moving Gaussians and $\mathcal{A}_g^{\text{collided}}$ for Gaussians that have accumulated on static geometry. For notational clarity, we omit the explicit state subscripts and use $\mathcal{A}_g = \{\sigma_g^{\text{collided}}, \mathbf{S}_g^{\text{collided}}, \mathbf{C}_g^{\text{collided}}\}$ to refer to the appropriate appearance parameters based on the Gaussian's current physical state. Appearance parameters are initialized using a large language model (LLM) that generates appropriate baseline values $\mathcal{A}_g^{\text{init}}$ from text prompts describing the desired weather effect. See Appendix B.4 for details on LLM initialization.

Recurrent Neural Dynamics Model. To model the continuous evolution of the dynamic Gaussians, we learn a recurrent neural dynamics model that predicts dynamic Gaussian evolution over time. At each timestep t, the model takes as input: (1) the previous rendered state $\{\mathbf{x}_g(t-1), \mathbf{R}_g(t-1), \mathcal{A}_g(t-1)\}$ from the previous timestep, where $\mathbf{x}_g(t-1)$ is the position of Gaussian g at time $t-1$, $\mathbf{R}_g(t-1)$ is the rotation of Gaussian g and $\mathcal{A}_g$ represents the appearance parameters of Gaussian g (opacity, scale, and color), (2) the physics velocity $\mathbf{v}_g^{\text{init}}(t)$ of Gaussian g at time t from simulation, and (3) the current timestep t. We implement separate MLPs for each physical state, with each MLP processing Gaussians independently. The positions, velocities, and timestep are processed through learned embedders (Fourier feature encoding for spatial inputs, sinusoidal encoding for time) before being concatenated with the current appearance parameters $\mathcal{A}_g(t-1)$ and fed to the appropriate state-specific MLP. The recurrence is initialized at $t = 0$ using physics simulation positions $\mathbf{x}_g^{\text{init}}(0)$ and the LLM-initialized appearance parameters $\mathcal{A}_g^{\text{init}}$. See Appendix B.1 for additional details on the network.

For each Gaussian g, the network predicts: (1) velocity correction $\boldsymbol{\Delta}\mathbf{v}_g$ relative to the simulated velocity, (2) angular velocity $\boldsymbol{\omega}_g$ for rotation updates, and (3) appearance deltas $\boldsymbol{\Delta}\mathcal{A}_g$. The refined motion and appearance are com-

puted as:

$$\begin{aligned}\mathbf{v}_g(t) &= \mathbf{v}_g^{\text{init}}(t) + \mathbf{\Delta v}_g,\\ \mathbf{x}_g(t) &= \mathbf{x}_g(t-1) + \mathbf{v}_g(t),\\ \mathbf{R}_g(t) &= \mathbf{R}_g(t-1) \circ \Delta\mathbf{R}(\boldsymbol{\omega}_g),\\ \mathcal{A}_g(t) &= \mathcal{A}_g(t-1) + \mathbf{\Delta}\mathcal{A}_g,\end{aligned} \tag{5}$$

where $\mathbf{v}_g^{\text{init}}(t)$ serves as a motion prior that is refined by the learned correction, $\circ$ denotes quaternion multiplication and $\Delta\mathbf{R}$ converts angular velocity to a quaternion delta, and appearance parameters evolve recurrently through additive updates. The recurrent model, however, introduces drift and error accumulation in the recurrent predictions as the learned trajectories $\mathbf{x}_g(t)$ diverge from the simulation trajectories, and $\mathbf{v}_g^{\text{init}}(t)$ may correspond to a different position in the physics simulation than the actual Gaussian position. The physics regularization losses introduced in the next section are therefore essential to mitigate this drift and maintain physical plausibility throughout optimization.

Physics-Guided Score Distillation Optimization. Next, we describe how our recurrent neural dynamics model is optimized through *Physics-Guided Score Distillation*, where the physics prior guides the optimization objective.

The optimization consists of three objectives. First, the *Video-SDS loss* $\mathcal{L}_{\text{Video-SDS}}$ (Eq. 3) provides photorealistic guidance from the video diffusion model, ensuring that rendered videos match the target text prompt. Second, *physics guidance losses* regularize the learned motion to remain close to the physics prior through three regularization terms:

$$\begin{aligned}\text{Position: } \mathcal{L}_{\text{xyz}} &= \|\mathbf{x}_g(t) - \mathbf{x}_g^{\text{sim}}(t)\|_2,\\ \text{Velocity: } \mathcal{L}_{\text{vel}} &= \|\mathbf{v}_g(t) - \mathbf{v}_g^{\text{sim}}(t)\|_2,\\ \text{Rotation: } \mathcal{L}_{\text{rot}} &= 2 \cdot \arccos(|\mathbf{R}_g(t) \cdot \mathbf{R}_g^{\text{init}}|,\end{aligned} \tag{6}$$

where $\mathbf{R}_g(t) \cdot \mathbf{R}_g^{\text{init}}$ is the dot product between two quaternion vectors. The position and velocity regularization losses calculate the $L2$ distance, preventing rendered trajectories from drifting away from the reference simulation trajectories. The rotation regularization loss calculates the angular distance between the two rotations, preventing excessive rotational drift from initial orientations.

Third, *appearance regularization* mitigates error accumulation inherent to the recurrent formulation by penalizing large appearance deltas:

$$\mathcal{L}_{\text{app}} = \|\mathbf{\Delta}\mathcal{A}_g\|_2 = \|\mathbf{\Delta}\sigma_g\|_2 + \|\mathbf{\Delta S}_g\|_2 + \|\mathbf{\Delta C}_g\|_2, \tag{7}$$

applied separately for active and collided Gaussian states. The complete objective combines these components with state-specific weights:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{Video-SDS}} + \lambda_{\text{xyz}}\mathcal{L}_{\text{xyz}} + \lambda_{\text{vel}}\mathcal{L}_{\text{vel}} + \lambda_{\text{rot}}\mathcal{L}_{\text{rot}} + \lambda_{\text{app}}\mathcal{L}_{\text{app}}. \tag{8}$$

Each regularization term has independent weight parameters to provide control over physics-appearance trade-off. See Appendix B.1 for additional details.

These regularization terms are critical for mitigating the drift and error accumulation discussed earlier: higher regularization weights pull the optimization closer to pure physics simulation (physically plausible but potentially lacking photorealism), while lower weights allow the optimization to be dominated by the Video-SDS objective (photorealistic but risking physical implausibility and trajectory drift). To balance this trade-off, we apply SDS-adaptive physics guidance, where all regularization weights are dynamically scaled by the instantaneous Video-SDS loss magnitude $\lambda_i \cdot |\mathcal{L}_{\text{Video-SDS}}|$. This creates an adaptive mechanism where physics guidance is amplified when the diffusion model exhibits high uncertainty (large $|\mathcal{L}_{\text{Video-SDS}}|$) and relaxes when the model is confident (small $|\mathcal{L}_{\text{Video-SDS}}|$), balancing physical plausibility with visual quality.

During training, we render video clips from random camera trajectories, compute $\mathcal{L}_{\text{total}}$, and backpropagate gradients to update network parameters. We employ progressive timestep annealing for the diffusion model to refine details. Through this framework, where physics both conditions network inputs and guides optimization, our model learns to generate physically plausible, photorealistic dynamic weather effects with continuous particle emission.

4. Experiments

We evaluate our framework for dynamic editing of static 3D Gaussian Splatting scenes through physics-guided score distillation. We present qualitative results across various scenes and effect types in Sec. 4.1, compare against prior work in Sec. 4.2, and provide ablation studies in Sec. 4.3. Finally, we discuss limitations in Sec. 4.4.

Scenes and dynamic weather effects. We demonstrate our framework on five diverse real-world static scenes: *Garden*, *Bicycle*, and *Stump* from the Mip-NeRF360 dataset [1], and *Playground* and *Truck* from the Tanks and Temples dataset [13]. For each scene, we create four types of dynamic weather effects: *snowfall*, *rainfall*, *fog*, and *sandstorm*. This diverse set of effects showcases our approach's ability to handle various dynamic scene modifications with different physical characteristics and temporal behaviors while maintaining physically plausible motion. Additionally, we demonstrate that our framework can generate creative appearance variations through text guidance (*purple snow*, *glittering yellow sand*, and *magic particles*).

4.1. Qualitative Results

Fig. 1 (*snowfall*) and Fig. 3 (all other dynamic weather effects) demonstrate our method's ability to create visually convincing and temporally consistent dynamic scene



Figure 3. Dynamic weather effects created with our physics-guided SDS framework across different scenes (input frames on left). Each column represents an increasing timestep, demonstrating temporal evolution with continuous particle emission and realistic scene interactions. Our approach enables scene-wide modifications where emitted particles interact naturally with static geometry and evolve coherently over time. The *snowfall* effect is presented in Fig. 1. As the *rainfall* effect is challenging to visualize, we provide zoomed-in regions to better illustrate the effect. Complete temporal visualizations for all effects are available in our project webpage.

modifications across different environments. Our physics-guided score distillation framework enables effects that exhibit physically plausible motion while maintaining photo-realistic appearance: snowflakes gently accumulate on surfaces through continuous particle emission, raindrops create wetness patterns, fog diffuses through the scene, and sandstorms envelope objects with physically-driven particle motion. Additionally, Fig. 4 showcases creative appearance variations generated by our framework through text guidance (e.g., *purple snow*) while maintaining physically plausible motion through our physics-guided optimization. Please refer to our project webpage for additional visualizations across different scenes, effects, and viewpoints.

4.2. Comparison To Baselines

Baselines. Our work addresses the challenge of introducing dynamic weather effects to static scenes, which differs from existing approaches. Recent 4D editing methods [14, 20] leverage score distillation for scene-wide modifications, however these methods operate in a different setting, applying edits to a fixed set of Gaussians. Similarly, 4D generation techniques like DreamGaussian4D [24] and

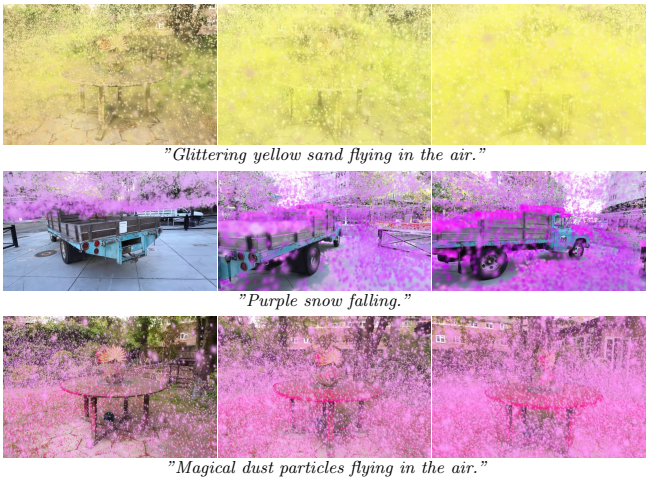


Figure 4. **Text Prompt Results.** Our physics-guided framework supports creative text prompts beyond standard weather. The method successfully synthesizes the target appearance while maintaining the physically plausible motion learned for the corresponding effect (e.g., sandstorm, snowfall).



Original ClimateNeRF GaussCtrl Ours (Small t) Ours (Large t)

Figure 5. **Qualitative comparison between our dynamic approach and static editing**. We show two different effects across separate rows, comparing the original scene, ClimateNeRF [16], GaussCtrl [33], and our method at both early (Small t) and later (Large t) timesteps.



w/o CH w/o Motion w/o App w/o PG Ours

Figure 6. **Qualitative Ablation Study**. We compare our full method against four variants. *(w/o CH)* Gaussians float above surfaces without collision handling; *(w/o Motion)* data-driven optimization alone yields incoherent and physically implausible motion; *(w/o App)* physics-only achieves plausible motion but lacks photorealistic appearance; *(w/o PG)* fixed physics motion without joint optimization prevents effective Video-SDS optimization, hindering appearance quality. Our full method achieves both physically plausible motion and photorealistic appearance through physics-guided score distillation. Highlighted regions contrast the successful surface accumulation in our method with the floating artifacts in w/o CH.

Table 1. **Quantitative Evaluation**. We compare against the static 3D editing methods ClimateNeRF [16] and GaussCtrl [33]. Results are shown for different edit categories, with F+S representing results for *fog* and *snow* edits (the only effects supported by ClimateNeRF) and All representing results across all edit types.

	Method	CLIP_{Sim} ↑	CLIP_{Dir} ↑	VQAScore ↑	ViCLIP-T ↑	VE-Bench ↑
F+S	ClimateNeRF	0.23	0.07	0.87	0.15	0.28
	GaussCtrl	0.25	0.08	0.71	0.16	0.24
	Ours	**0.29**	**0.12**	**0.92**	**0.20**	**0.45**
All	GaussCtrl	0.24	0.07	0.64	0.15	0.21
	Ours	**0.28**	**0.11**	**0.89**	**0.19**	**0.41**

Animate124 [38] primarily target object-centric animation rather than scene-wide effects. Physics-based methods such as PhysGaussian [34] and Feature Splatting [22] focus on manipulating existing scene elements through simulation but are not designed to synthesize new dynamic elements with photorealistic appearance. RainyGS [3] addresses rain synthesis through physics-based particle systems with rain-specific simulation and rasterization techniques, However it does not generalize to other weather effects. Additionally, we cannot provide direct comparisons as their code is not publicly available. Given these limitations, we compare our approach against static 3D editing methods that represent the most directly comparable baselines. Specifically, we evaluate against ClimateNeRF [16], which adds static weather effects to NeRFs, and GaussCtrl [33], a method for text-guided editing of 3D Gaussian Splatting scenes.

In the appendix, we compare our method against Instruct-4DGS [14], a 4D editing baseline. See Appendix A for the 4D editing baseline comparison details and results.

Evaluation Metrics. We employ metrics that assess visual fidelity and alignment with the intended edit. Our evaluation includes comparisons to static methods; therefore, we consider both image-based and video-based metrics: (1). *CLIP Similarity* [23], (CLIP_{Sim}), measures the CLIP similarity between the output rendered images and the target text prompts. (2). *CLIP Direction Similarity* [5] (CLIP_{Dir}), measures the cosine distance between the direction of the change from the input and output rendered images and the direction of the change from an input prompt to the edit prompt. (3). *VQAScore* [17], measures the probability that the answer to the question "Does the image show caption" is "Yes" by inputting the rendered image and caption into a VQA model. (4). *VE-Bench Score* [28], provides a score that evaluates video editing quality by measuring temporal consistency, visual quality, and edit fidelity for dynamic video content through automated metrics designed specifically for assessing video-based edits. (5). *ViCLIP-T*, as presented in [36], uses ViCLIP [30], a video CLIP model that incorporates temporal information when processing videos. ViCLIP-T calculates the cosine similarity between text and video embeddings, measuring how well the output video aligns with the target text prompt, providing a measure of text faithfulness for dynamic content.

Comparisons. A qualitative comparison to static 3D editing methods is shown in Fig. 5. We show our results at two different timesteps: one at an early stage of the simulation (Small t) and one at the end (Large t). In the *snowfall* example in the first row, the static methods cannot capture the temporal dynamics of continuous particle emission and accumulation, yielding only surface modifications (ClimateNeRF) or appearance changes (GaussCtrl). In the second row, for the *fog* example, the effect is either unnoticeable (ClimateNeRF) or unrealistic (GaussCtrl). Our physics-guided framework produces effects that exhibit both physically plausible temporal evolution and photorealistic appearance. See Appendix C for additional comparisons across different scenes, effects, and viewpoints.

In Tab. 1 we show a quantitative comparison to these static editing methods. We create moving camera videos from the static method results to enable fair comparison using video-based metrics. ClimateNeRF only performs modifications for the *fog* and *snow* effects, thus we show results over these effects (F+S) and also over all weather effects. As illustrated in the table, our method achieves higher scores across both video and image-based metrics, demonstrating that our approach produces high-quality effects even in static frames, while the temporal dimension, enabled by continuous particle emission and physics-guided optimization, further distinguishes our method.

4.3. Ablations

We conduct ablation studies to evaluate four variants: (1) without collision handling *(w/o CH)*, removing mesh-based collision handling techniques; (2) without appearance optimization *(w/o App)*, using only physics simulation with LLM-initialized appearance parameters and no Video-SDS optimization; (3) without motion simulation *(w/o Motion)*, using only data-driven optimization by initializing a 4D Gaussian Splatting representation [32] and optimizing its deformation field using Video-SDS; and (4) without physics guidance *(w/o PG)*, where motion trajectories from physics simulation are fixed and only appearance parameters are optimized through Video-SDS, eliminating the joint optimization, physics regularization losses, and physics conditioning that enable our approach.

Fig. 6 and Tab. 2 illustrate that each ablated component leads to a significant degradation in quality. The *w/o CH* ablation demonstrates that without proper collision handling, dynamic Gaussians fail to interact convincingly with static scene geometry, with snowflakes floating unnaturally above surfaces instead of accumulating realistically. While the *w/o App* variant achieves physically plausible motion, it produces suboptimal visual appearance due to simplified LLM-initialized parameters. The *w/o Motion* variant produces incoherent and noisy results, producing significantly lower results than other variants. These results demonstrate that video diffusion models alone cannot reliably generate physically plausible motion for complex multi-particle scenarios and are not designed to introduce new Gaussians. The *w/o PG* variant demonstrates that fixed motion prevents effective Video-SDS optimization. Without the joint optimization enabled by our physics-guidance prior, the model cannot learn the motion refinements required for the diffusion model to satisfy the text prompt.

Table 2. **Quantitative Ablation Results**. Comparison against variants without collision handling (w/o CH), appearance optimization (w/o App), without motion simulation (w/o Motion) or physical guidance (w/o PG).

Method	CLIP_{Sim} ↑	CLIP_{Dir} ↑	VQAScore ↑	ViCLIP-T ↑	VE-Bench ↑
w/o CH	0.24	0.10	0.83	0.16	0.34
w/o App	0.25	0.10	0.82	0.16	0.37
w/o Motion	0.18	0.03	0.35	0.08	0.13
w/o PG	0.26	0.10	0.85	0.17	0.37
Ours	**0.28**	**0.11**	**0.89**	**0.19**	**0.41**

In the appendix, we conduct additional experiments, ablating our SDS-adaptive weighting mechanism, our regularization losses and studying the threshold behavior of trajectory drift. See Appendix A for the additional ablations and results.

4.4. Limitations

While our physics-guided score distillation framework achieves high-quality results, several limitations remain. First, our current approach focuses on adding dynamic elements that interact with static scene geometry but does not enable bi-directional interactions where dynamic effects cause deformation or movement of existing scene elements. Extending our framework to support such interactions represents a promising direction for future work. Second, our framework does not currently update the appearance of static Gaussians to reflect environment changes induced by dynamic effects, such as shadows or ambient lighting variations. The modification of static Gaussian appearance to reflect these environmental effects represents an important direction for future research. Third, while physics simulation provides a strong initialization for optimization, our approach assumes that learned trajectories remain close to the reference simulation. If the optimized trajectories deviate significantly from the physics prior beyond a certain threshold, the guidance velocity signal becomes unreliable. Exploring ways to adapt or recalibrate the guidance is an interesting direction for future work.

5. Conclusion

We presented *Physics-Guided Score Distillation*, a framework for dynamic editing of 3D Gaussian Splatting scenes. Our key insight is to leverage physics simulation as a powerful motion prior, enabling the joint refinement of both motion and appearance within a unified Video-SDS optimiza-

tion. Our ablations confirm this joint refinement is essential for achieving coherent, photorealistic results, as simpler alternatives fail. This work bridges physics-based simulation with data-driven diffusion models, opening a new path for realistic, dynamic 3D content creation.

References

[1] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5470–5479, 2022. 2, 5

[2] Yiwen Chen, Zilong Chen, Chi Zhang, Feng Wang, Xiaofeng Yang, Yikai Wang, Zhongang Cai, Lei Yang, Huaping Liu, and Guosheng Lin. Gaussianeditor: Swift and controllable 3d editing with gaussian splatting. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 21476–21485, 2024. 2

[3] Qiyu Dai, Xingyu Ni, Qianfan Shen, Wenzheng Chen, Baoquan Chen, and Mengyu Chu. Rainygs: Efficient rain synthesis with physically-based gaussian splatting. *arXiv preprint arXiv:2503.21442*, 2025. 3, 7

[4] Yutao Feng, Xiang Feng, Yintong Shang, Ying Jiang, Chang Yu, Zeshun Zong, Tianjia Shao, Hongzhi Wu, Kun Zhou, Chenfanfu Jiang, and Yin Yang. Gaussian splashing: Unified particles for versatile motion synthesis and rendering. *arXiv preprint arXiv:2401.15318*, 2024. 3

[5] Rinon Gal, Or Patashnik, Haggai Maron, Amit H Bermano, Gal Chechik, and Daniel Cohen-Or. Stylegan-nada: Clip-guided domain adaptation of image generators. *ACM Transactions on Graphics (TOG)*, 41(4):1–13, 2022. 7

[6] Ayaan Haque, Matthew Tancik, Alexei A Efros, Aleksander Holynski, and Angjoo Kanazawa. Instruct-nerf2nerf: Editing 3d scenes with instructions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19740–19750, 2023. 2

[7] Yuanming Hu, Tzu-Mao Li, Luke Anderson, Jonathan Ragan-Kelley, and Frédo Durand. Taichi: a language for high-performance computation on spatially sparse data structures. *ACM Transactions on Graphics (TOG)*, 38(6): 1–16, 2019. 12

[8] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2d gaussian splatting for geometrically accurate radiance fields. In *ACM SIGGRAPH 2024 conference papers*, pages 1–11, 2024. 12

[9] Tianyu Huang, Haoze Zhang, Yihan Zeng, Zhilu Zhang, Hui Li, Wangmeng Zuo, and Rynson WH Lau. Dreamphysics: Learning physical properties of dynamic 3d gaussians with video diffusion priors. *arXiv preprint arXiv:2406.01476*, 2024. 2, 3

[10] Chenfanfu Jiang, Craig Schroeder, Joseph Teran, Alexey Stomakhin, and Andrew Selle. The material point method for simulating continuum materials. In *Acm siggraph 2016 courses*, pages 1–52. 2016. 3

[11] Yanqin Jiang, Chaohui Yu, Chenjie Cao, Fan Wang, Weiming Hu, and Jin Gao. Animate3d: Animating any 3d model with multi-view video diffusion. *Advances in Neural Information Processing Systems*, 37:125879–125906, 2025. 2

[12] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023. 2, 3, 12

[13] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36 (4):1–13, 2017. 2, 5

[14] Joohyun Kwon, Hanbyel Cho, and Junmo Kim. Efficient dynamic scene editing via 4d gaussian-based static-dynamic separation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 26855–26865, 2025. 2, 6, 7, 11, 12, 13

[15] Xuan Li, Yi-Ling Qiao, Peter Yichen Chen, Krishna Murthy Jatavallabhula, Ming Lin, Chenfanfu Jiang, and Chuang Gan. Pac-nerf: Physics augmented continuum neural radiance fields for geometry-agnostic system identification. *arXiv preprint arXiv:2303.05512*, 2023. 2, 3

[16] Yuan Li, Zhi-Hao Lin, David Forsyth, Jia-Bin Huang, and Shenlong Wang. Climatenerf: Extreme weather synthesis in neural radiance field. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3227–3238, 2023. 2, 7, 14, 15, 16, 17, 18

[17] Zhiqiu Lin, Deepak Pathak, Baiqi Li, Jiayao Li, Xide Xia, Graham Neubig, Pengchuan Zhang, and Deva Ramanan. Evaluating text-to-visual generation with image-to-text generation. In *European Conference on Computer Vision*, pages 366–384. Springer, 2024. 7

[18] Huan Ling, Seung Wook Kim, Antonio Torralba, Sanja Fidler, and Karsten Kreis. Align your gaussians: Text-to-4d with dynamic 3d gaussians and composed diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8576–8588, 2024. 2

[19] Xiangyue Liu, Han Xue, Kunming Luo, Ping Tan, and Li Yi. Genn2n: Generative nerf2nerf translation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5105–5114, 2024. 2

[20] Linzhan Mou, Jun-Kun Chen, and Yu-Xiong Wang. Instruct 4d-to-4d: Editing 4d scenes as pseudo-3d scenes using 2d diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20176–20185, 2024. 2, 6

[21] 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*, 2022. 2, 4

[22] Ri-Zhao Qiu, Ge Yang, Weijia Zeng, and Xiaolong Wang. Feature splatting: Language-driven physics-based scene synthesis and editing. *arXiv preprint arXiv:2404.01223*, 2024. 2, 7, 12

[23] 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, 2021. 7

[24] Jiawei Ren, Liang Pan, Jiaxiang Tang, Chi Zhang, Ang Cao, Gang Zeng, and Ziwei Liu. Dreamgaussian4d: Generative 4d gaussian splatting. *arXiv preprint arXiv:2312.17142*, 2023. 2, 6

[25] Etai Sella, Gal Fiebelman, Peter Hedman, and Hadar Averbuch-Elor. Vox-e: Text-guided voxel editing of 3d objects. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 430–440, 2023. 2

[26] Uriel Singer, Shelly Sheynin, Adam Polyak, Oron Ashual, Iurii Makarov, Filippos Kokkinos, Naman Goyal, Andrea Vedaldi, Devi Parikh, Justin Johnson, et al. Text-to-4d dynamic scene generation. In *Proceedings of the 40th International Conference on Machine Learning*, pages 31915–31929. PMLR, 2023. 2, 4

[27] Alexey Stomakhin, Craig Schroeder, Lawrence Chai, Joseph Teran, and Andrew Selle. A material point method for snow simulation. *ACM Transactions on Graphics (TOG)*, 32(4): 1–10, 2013. 3

[28] Shangkun Sun, Xiaoyu Liang, Songlin Fan, Wenxu Gao, and Wei Gao. Ve-bench: Subjective-aligned benchmark suite for text-driven video editing quality assessment. *arXiv preprint arXiv:2408.11481*, 2024. 7

[29] Junjie Wang, Jiemin Fang, Xiaopeng Zhang, Lingxi Xie, and Qi Tian. Gaussianeditor: Editing 3d gaussians delicately with text instructions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20902–20911, 2024. 2

[30] Yi Wang, Yinan He, Yizhuo Li, Kunchang Li, Jiashuo Yu, Xin Ma, Xinhao Li, Guo Chen, Xinyuan Chen, Yaohui Wang, Conghui He, Ping Luo, Ziwei Liu, Yali Wang, Limin Wang, and Yu Qiao. Internvid: A large-scale video-text dataset for multimodal understanding and generation, 2024. 7

[31] Thomas Wimmer, Michael Oechsle, Michael Niemeyer, and Federico Tombari. Gaussians-to-life: Text-driven animation of 3d gaussian splatting scenes. *arXiv preprint arXiv:2411.19233*, 2024. 2

[32] 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. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 20310–20320, 2024. 8, 15

[33] Jing Wu, Jia-Wang Bian, Xinghui Li, Guangrun Wang, Ian Reid, Philip Torr, and Victor Adrian Prisacariu. Gaussctrl: Multi-view consistent text-driven 3d gaussian splatting editing. In *European Conference on Computer Vision*, pages 55–71. Springer, 2024. 2, 7, 15, 16, 17, 18

[34] Tianyi Xie, Zeshun Zong, Yuxing Qiu, Xuan Li, Yutao Feng, Yin Yang, and Chenfanfu Jiang. Physgaussian: Physics-integrated 3d gaussians for generative dynamics. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4389–4398, 2024. 2, 3, 7

[35] Zhuoyi Yang, Jiayan Teng, Wendi Zheng, Ming Ding, Shiyu Huang, Jiazheng Xu, Yuanming Yang, Wenyi Hong, Xiaohan Zhang, Guanyu Feng, et al. Cogvideox: Text-to-video diffusion models with an expert transformer. *arXiv preprint arXiv:2408.06072*, 2024. 13

[36] Guy Yariv, Yuval Kirstain, Amit Zohar, Shelly Sheynin, Yaniv Taigman, Yossi Adi, Sagie Benaim, and Adam Polyak. Through-the-mask: Mask-based motion trajectories for image-to-video generation. *arXiv preprint arXiv:2501.03059*, 2025. 7

[37] Tianyuan Zhang, Hong-Xing Yu, Rundi Wu, Brandon Y Feng, Changxi Zheng, Noah Snavely, Jiajun Wu, and William T Freeman. Physdreamer: Physics-based interaction with 3d objects via video generation. In *European Conference on Computer Vision*, pages 388–406. Springer, 2024. 2

[38] 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. 2, 7

[39] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018. 12, 14

[40] Jingyu Zhuang, Chen Wang, Liang Lin, Lingjie Liu, and Guanbin Li. Dreameditor: Text-driven 3d scene editing with neural fields. In *SIGGRAPH Asia 2023 Conference Papers*, pages 1–10, 2023. 2

Appendix

A. Additional Ablations and Comparisons

4D Editing Baseline Comparison. We compare our method against Instruct-4DGS [14], a recent approach for scene-wide, text-guided editing of 4D Gaussian Splatting scenes. Instruct-4DGS operates in three stages: first generating consistent 2D target images by editing multiview renderings using an image-conditioned diffusion model, then optimizing the canonical 3D Gaussian Splatting representation to match these 2D priors through reconstruction loss, and finally refining the canonical representation using diffusion guidance while keeping the deformation field fixed. The method is designed for editing existing dynamic scenes captured from videos, where it modifies the canonical representation while freezing the pre-trained deformation field to preserve the original motion.

However, Instruct-4DGS is not directly suited for our task of introducing new dynamic weather effects to static scenes, as it assumes an existing dynamic scene and edits only the canonical representation while preserving the original deformation field. To enable comparison, we initialize the deformation field to zero and train it jointly during the refinement stage, allowing the method to learn motion dynamics from diffusion guidance. We use the code provided by the authors (https://github.com/juhyeon-kwon/Instruct-4DGS) with the default parameters, extending the optimization to 3,000 iterations for the first stage and 5,000 iterations for the second stage to ensure convergence.

As shown in Fig. 7 and Tab. 3, Instruct-4DGS produces incoherent results with physically implausible motion patterns. Without a physics-based motion prior to guide the optimization, the method struggles to learn realistic dynamics for complex multi-particle weather phenomena, resulting in unnatural motion and visual artifacts.

SDS-Adaptive Weight Ablations. We ablate our SDS-adaptive weighting mechanism by replacing all adaptive regularization weights with constant values across both physics guidance losses ($\mathcal{L}_{\text{xyz}}$, $\mathcal{L}_{\text{vel}}$, $\mathcal{L}_{\text{rot}}$) and appearance regularization ($\mathcal{L}_{\text{app}}$). We evaluate four fixed weight values: λ=1, λ=100, λ=10^4, and λ=10^6, comparing against our full method with SDS-adaptive scaling.

As shown in Fig. 8 and Tab. 4, low constant weights (1, 100) result in noisy and unstable optimization, with weight 1 producing severe visual artifacts and weight 100 showing reduced but still significant noise. At moderate weight (10^4), we observe color artifacts with unnatural pinkish tones, indicating that constant regularization fails to properly balance physics guidance with photorealistic optimization. Conversely, excessively high constant weight (10^6) over-regularizes the optimization, producing results similar to the *w/o App* ablation (Sec. 4.3) where appearance remains close to LLM-initialized parameters with minimal refinement. These results demonstrate that fixed regularization weights cannot adequately balance the competing objectives throughout optimization. Because the Video-SDS loss exhibits varying magnitude during training, regularization weights must adapt dynamically to maintain this balance. Our SDS-adaptive approach scales all regularization weights by the instantaneous Video-SDS loss magnitude, automatically adjusting the strength of physics guidance and appearance stability as the diffusion model's uncertainty changes throughout optimization.

Regularization Loss Ablations. We conduct ablation studies on each regularization loss component to validate their individual contributions to our physics-guided optimization. We evaluate four variants: (1) without position regularization *(w/o $\mathcal{L}_{xyz}$)*, (2) without velocity regularization *(w/o $\mathcal{L}_{vel}$)*, (3) without rotation regularization *(w/o $\mathcal{L}_{rot}$)*, and (4) without appearance regularization *(w/o $\mathcal{L}_{app}$)*.

As shown in Fig. 9 and Tab. 5, removing position or velocity regularization causes significant trajectory drift from the physics prior, as the learned motion deviates substantially from the simulated reference trajectories without constraints to maintain proximity. The rotation regularization ablation shows minimal impact, producing results similar to our full method, indicating that rotational corrections remain small throughout optimization. Without appearance regularization, we observe severe error accumulation in the recurrent predictions, resulting in completely noisy and incoherent results.

Trajectory Drift Ablations. As discussed in our limitations (Sec. 4.4), our physics guidance assumes that learned trajectories remain reasonably close to the simulation reference, thus when trajectories deviate significantly beyond a certain threshold, the guidance signal becomes unreliable. To investigate this threshold behavior, we replace our SDS-adaptive physics guidance with constant regularization weights for position loss $\mathcal{L}_{\text{xyz}}$. We evaluate four fixed weight values: λ_{xyz}=1, λ_{xyz}=100, λ_{xyz}=10^4, and λ_{xyz}=10^6, comparing against our full method with adaptive weighting. To quantify trajectory drift, we measure the average distance between rendered and simulated positions over all timesteps (reported as "Avg. Drift" in Tab. 6):

$$\text{Avg. Drift} = \frac{1}{T}\sum_{t=1}^{T}\frac{1}{N_t}\sum_{i=1}^{N_t}\|\mathbf{x}_i^{\text{rendered}} - \mathbf{x}_i^{\text{sim}}\|_2 \quad (9)$$

where T is the total timesteps and N_t is the number of moving Gaussians at timestep t. As shown in Fig. 10 and Tab. 6, low regularization weights (1, 100, 10^4) allow trajectories to deviate significantly from the simulation prior, causing high drift values that degrade physics guidance quality. Conversely, excessively high regularization weight (10^6) over-constrains the optimization, such that drift be-

Table 3. **4D Editing Baseline Comparison**. Quantitative comparison against Instruct-4DGS [14], a recent 4D editing method. Without physics-based motion guidance, Instruct-4DGS struggles to generate coherent multi-particle dynamics, resulting in lower scores across all metrics.

Method	CLIP_{Sim} ↑	CLIP_{Dir} ↑	VQAScore ↑	ViCLIP-T ↑	VE-Bench ↑
Instruct-4DGS	0.21	0.07	0.57	0.13	0.19
Ours	**0.28**	**0.11**	**0.89**	**0.19**	**0.41**

Table 4. **SDS-Adaptive Weight Ablations**. Quantitative comparison of constant regularization weights versus our SDS-adaptive approach. Low weights (λ=1, λ=100) produce noisy and unstable results, moderate weight (λ=10^4) shows color artifacts, and very high weight (λ=10^6) over-constrains optimization. Our SDS-adaptive approach outperforms all fixed weights by dynamically adjusting to Video-SDS loss magnitude.

Method	CLIP_{Sim} ↑	CLIP_{Dir} ↑	VQAScore ↑	ViCLIP-T ↑	VE-Bench ↑
λ=1	0.14	0.02	0.25	0.07	0.08
λ=100	0.21	0.05	0.43	0.09	0.18
λ=10^4	0.26	0.10	0.84	0.16	0.39
λ=10^6	0.25	0.10	0.83	0.16	0.36
Ours	**0.28**	**0.11**	**0.89**	**0.19**	**0.41**

Table 5. **Regularization Loss Ablations**. Quantitative comparison of variants removing individual regularization losses: without position regularization *(w/o $\mathcal{L}_{xyz}$)*, velocity regularization *(w/o $\mathcal{L}_{vel}$)*, rotation regularization *(w/o $\mathcal{L}_{rot}$)*, or appearance regularization *(w/o $\mathcal{L}_{app}$)*. Removing position or velocity regularization causes trajectory drift, while removing appearance regularization leads to severe error accumulation and incoherent results.

Method	CLIP_{Sim} ↑	CLIP_{Dir} ↑	VQAScore ↑	ViCLIP-T ↑	VE-Bench ↑
w/o $\mathcal{L}_{\text{xyz}}$	0.26	0.09	0.83	0.16	0.33
w/o $\mathcal{L}_{\text{vel}}$	0.26	0.10	0.84	0.16	0.36
w/o $\mathcal{L}_{\text{rot}}$	**0.28**	**0.11**	0.87	**0.19**	0.40
w/o $\mathcal{L}_{\text{app}}$	0.15	0.01	0.28	0.05	0.09
Ours	**0.28**	**0.11**	**0.89**	**0.19**	**0.41**

comes minimal but motion remains rigidly fixed to simulation trajectories, preventing the refinements necessary for photorealism (similar to the w/o PG ablation (Sec. 4.3). This demonstrates the existence of a critical threshold: insufficient regularization causes unreliable guidance due to excessive drift, while excessive regularization prevents effective optimization. Our SDS-adaptive approach aims to maintain trajectories within this viable range, though developing methods to dynamically recalibrate guidance as trajectories diverge remains an important direction for future work.

B. Additional Details

B.1. Implementation Details

For each scene we first use 3D Gaussian Splatting [12] to train our static 3D representation, using the default hyper parameters found in the code provided by the authors (https://github.com/graphdeco-inria/gaussian-splatting). To extract meshes we follow the mesh extraction method in [8] by using the code provided by the authors (https://github.com/hbb1/2d-gaussian-splatting/). We render depth maps of the training views using the depth value of the splats projected to the pixels and utilize truncated signed distance fusion (TSDF) to fuse the reconstruction depth maps, using Open3D [39]. We set the voxel size to 0.004, the truncated threshold to 0.02 during TSDF fusion and extract a mesh with resolution of 1024.

Table 6. **Trajectory Drift Ablations**. Investigation of the trajectory drift threshold using constant regularization weights for $\mathcal{L}_{\text{xyz}}$. Avg. Drift measures average distance between rendered and simulated positions. Results demonstrate a critical threshold: insufficient regularization causes excessive drift that degrades guidance quality, while excessive regularization prevents meaningful optimization.

Method	Avg. Drift	CLIP_{Sim} ↑	CLIP_{Dir} ↑	VQAScore ↑	ViCLIP-T ↑	VE-Bench ↑
λ_{xyz}=1	4.32	0.25	0.09	0.80	0.15	0.32
λ_{xyz}=100	2.37	0.26	0.09	0.82	0.15	0.33
λ_{xyz}=10^4	1.46	0.27	0.10	0.85	0.17	0.39
λ_{xyz}=10^6	0.03	0.26	0.10	0.83	0.16	0.38
Ours	0.24	**0.28**	**0.11**	**0.89**	**0.19**	**0.41**

For the simulation, we base our MPM simulation on Taichi [7], which supports various types of materials (e.g., *snow*, *water*, *sand*, etc.). We build on [22], using the code provided by the authors (https://github.com/vuer-ai/feature-splatting-inria). We first align the Gaussians to the particle system coordinates and then normalize them to fit in a unit cube and center them appropriately with the bottom aligned to a ground plane. We then add the static particles to the simulation as "stationary" material particles.

We then add particles based on their effect-specific motion simulation parameters (detailed in Sec. B.3). We run the simulation for 500 steps. We set the threshold for particle active state updates at 0.001. Our grid resolution is set to 64^3. After each simulation step we transform the positions back to the Gaussian world and save them for rendering. We also save the collided Gaussians position at the time of collision.

For physics-guided score distillation optimization, we implement separate neural dynamics models for active and collided Gaussian states. Each model consists of a shared MLP backbone with separate prediction heads. The models take as input: (1) XYZ positions encoded through Fourier feature embeddings (24 dimensions), (2) physics velocities encoded similarly (24 dimensions), (3) time encoded through sinusoidal embeddings (24 dimensions), and (4) the previous state's appearance parameters. The shared backbone consists of two hidden layers with 128 hidden dimensions and ReLU activations. For active Gaussians, the model predicts velocity corrections $\Delta\mathbf{v}$, angular velocities $\boldsymbol{\omega}$, and appearance deltas ($\Delta\sigma$, $\Delta\mathbf{S}$, $\Delta\mathbf{C}$). For col-



Figure 7. **Qualitative Comparison to 4D Editing Baseline**. We compare our method against Instruct-4DGS [14] across two dynamic weather effects over increasing timesteps. Top two rows show the *sandstorm* effect, bottom two rows show the *snowfall* effect. Instruct-4DGS produces noisy and incoherent results with physically implausible motion patterns, as the method lacks physics-based guidance to learn realistic multi-particle dynamics. Our physics-guided score distillation framework generates photorealistic weather effects with both plausible motion and photorealistic appearance.

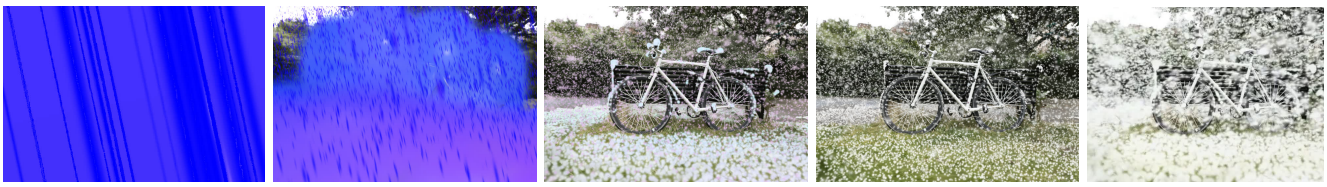


Figure 8. **SDS-Adaptive Weight Ablations**. Comparison of constant regularization weights versus our SDS-adaptive approach. λ=1 produces severe visual artifacts and noise due to insufficient regularization; λ=100 shows reduced but still significant noise and color distortions; λ=10^4 exhibits unnatural color artifacts with pinkish tones; λ=10^6 over-regularizes the optimization, preventing refinement and producing results similar to the w/o App ablation (Sec. 4.3), where appearance remains close to LLM-initialized parameters. These results demonstrate that fixed regularization weights cannot adapt to the varying and noisy Video-SDS loss throughout optimization. Our SDS-adaptive approach dynamically scales regularization based on instantaneous SDS loss magnitude, automatically balancing physics guidance and appearance stability.

lided Gaussians, the model predicts only appearance deltas as their motion is fixed at collision states.

We employ Video Score Distillation Sampling using CogVideoX-2B [35] as the video diffusion model. During optimization, we render video clips at 720×480 resolution with a length of 9 frames from randomly sampled training camera viewpoints. We use a guidance scale of 100 and optimize for 1000 iterations with an initial learning rate of 1×10^{-4}, using a cosine annealing schedule. For timestep sampling, we employ progressive annealing, sampling timesteps from $[t_{\text{min}}, t_{\text{max}}]$, where $t_{\text{min}} = 20$ and t_{max} decreases linearly from 980 to 740. To balance the gradients between the diffusion prior and regularization terms, we employ SDS-adaptive physics guidance where all regularization weights are dynamically scaled by the instantaneous magnitude of the Video-SDS loss ($|\mathcal{L}_{\text{SDS}}|$). Consequently, the regularization weights act as a percentage relative to the diffusion gradient magnitude. We set the physics regularization weights as follows: $\lambda_{\text{xyz}} = 0.1$, $\lambda_{\text{vel}} = 0.1$, $\lambda_{\text{rot}} = 0.1$. We set the appearance regularization weights as follows: for active particles, $\lambda_{\sigma}^{\text{active}} = 1.0$, $\lambda_{\mathbf{S}}^{\text{active}} = 1.0$, and $\lambda_{\mathbf{C}}^{\text{active}} = 35.0$; for collided particles, we enforce stronger regularization with $\lambda_{\sigma}^{\text{collided}} = 35.0$, $\lambda_{\mathbf{S}}^{\text{collided}} = 35.0$, and $\lambda_{\mathbf{C}}^{\text{collided}} = 35.0$.

Runtime. Our framework runs on a single NVIDIA A100 40GB GPU. Physics simulation takes approximately 2 min-



w/o $\mathcal{L}_{xyz}$ w/o $\mathcal{L}_{vel}$ w/o $\mathcal{L}_{rot}$ w/o $\mathcal{L}_{app}$ Ours

Figure 9. **Regularization Loss Ablations**. Comparison of our full method against variants removing individual regularization losses. *w/o* $\mathcal{L}_{xyz}$ and *w/o* $\mathcal{L}_{vel}$ cause trajectory drift from the physics prior, as the sandstorm passes above the bicycle and the bench; *w/o* $\mathcal{L}_{rot}$ shows minimal impact with results similar to our full method, indicating small rotational corrections; *w/o* $\mathcal{L}_{app}$ leads to severe error accumulation with noisy appearance. These results demonstrate that physics regularization is essential for maintaining plausible motion during Video-SDS optimization, while appearance regularization is critical for preventing error accumulation in our recurrent neural dynamics model.



λ_{xyz}=1 (4.32) λ_{xyz}=100 (2.37) λ_{xyz}=10^4 (1.46) λ_{xyz}=10^6 (0.03) Ours (0.24)

Figure 10. **Trajectory Drift Ablations**. Comparison of constant regularization weights for $\mathcal{L}_{xyz}$ demonstrating threshold behavior. Numbers in parentheses show Avg. Drift (mean Euclidean distance between rendered and simulated positions). At low weights (λ_{xyz}=1, λ_{xyz}=100, λ_{xyz}=10^4), active snow Gaussians drift significantly from simulation trajectories and fail to reach the ground, resulting in minimal or no visible falling snow (though accumulated snow remains visible as its positions are fixed at collision states). At extremely high weight (λ_{xyz}=10^6), drift is minimal but optimization is over-constrained, preventing photorealistic refinement. Our SDS-adaptive approach maintains low drift while enabling effective optimization.

utes and physics-guided optimization takes approximately 1 hour. Once optimized, dynamic effect rendering achieves 10 FPS.

B.2. Collision Handling Details

***Snowfall* Collision Handling**. For the *snowfall* effect we employ a dual approach. First, for surface accumulation, we compute the closest surface point and surface normal on the mesh to each interacting Gaussian using Open3D [39]. We project the Gaussians onto their closest surface point along the surface normal. Second, for background enhancement during the final rendering, we finetune our pretrained 3DGS model on images generated by a pretrained ClimateNeRF [16] model for snow. This finetuning process runs for 1000 iterations, optimizing only the scale, opacity, and color parameters of the background Gaussians (defined as all Gaussians outside our extracted mesh bounds). We then interpolate these parameters over the timesteps to create a gradual snow effect in the background environment.

***Sandstorm* Collision Handling**. For the *sandstorm* effect we perform the same surface projection as in the *snowfall* effect. We also use anisotropic scaling on the accumulated Gaussians in order to create thin, flat accumulations.

***Rainfall* Collision Handling**. For the *rainfall* effect, we also employ a dual approach. First, for surface accumulation, we perform the same surface projection as in the *snowfall* effect, but instead of accumulating visible Gaussians, we set these collided Gaussians' opacity to zero. Second, we utilize a volumetric grid-based approach to model surface wetness for the final rendering. When rain Gaussians collide with the mesh, we compute the closest surface points and increase wetness values in a coarse 3D grid surrounding the mesh, applying a Gaussian smoothing kernel to spread the effect naturally around impact points. The wetness grid undergoes temporal decay, simulating the gradual drying of surfaces. During rendering, we modify the appearance of the original Gaussians based on local wetness values, darkening affected areas to create realistic wet surface effects.

B.3. Motion Simulation Parameters

***Snowfall*.** Our *snowfall* simulation uses a Young's modulus of 0.14 and Poisson ratio of 0.2. We emit 1000 particles every 2 frames from random positions above the scene with initial vertical velocity of -0.5 and horizontal variation between -0.1 and 0.1. Standard gravity (0, -9.8, 0) is applied.

***Rainfall*.** Our *rainfall* simulation uses a Young's modulus of 0.08 and Poisson ratio of 0.45. We emit 1000 particles every 2 frames from random positions above the scene with initial vertical velocity of -0.5 and horizontal variation between -0.1 and 0.1. Standard gravity (0, -9.8, 0) is applied.

***Sandstorm*.** Our *sandstorm* simulation uses a Young's modulus of 0.08 and Poisson ratio of 0.3. We emit 1000 particles

every 2 frames from one side of the scene with strong horizontal velocity (0.8-1.2) and minor vertical variation (-0.2 to 0.2). Standard gravity (0, -9.8, 0) is applied.

***Fog*.** Our *fog* simulation differs by emitting 5000 water particles throughout the scene volume rather than from above. Particles have minimal initial velocity variation (-0.1 to 0.1) with slight horizontal drift (0.5 in x-direction). Nearly negligible gravity (0.5, -0.1, 0) is applied.

B.4. Appearance Initialization

We detail our appearance parameter LLM-initialization. We employ a large language model (Claude Sonnet 4, Anthropic, 2025) to generate baseline parameters A_{init}^{active} and $A_{init}^{collided}$ for each effect. We prompt the LLM with detailed descriptions to obtain appropriate scale, opacity, and color parameters.

For example, for snowfall we use:

```
"Generate appearance parameters for
realistic falling snow particles in a
3D Gaussian Splatting scene.
Provide: scale, opacity, and RGB color
values. Format as numerical values."
```

We then optimize these initialized parameters using Video-SDS with the following text prompts combining effect descriptions with scene-specific details:

Optimization Prompts:

- *Snowfall*: "Fluffy snowflakes are falling in a [SCENE DESCRIPTION], accumulating on surfaces. Photorealistic, high detail"
- *Rainfall*: "Rain falling in a [SCENE DESCRIPTION], wettening the surfaces. Photorealistic, high detail"
- *Sandstorm*: "A sandstorm engulfs the [SCENE DESCRIPTION], with swirling clouds of sand in the air and the sand accumulates on surfaces. Photorealistic, high detail"
- *Fog*: "Fog moving across the [SCENE DESCRIPTION], Photorealistic, high detail"

Scene Descriptions:

- *Bicycle*: "A bicycle leaning on a bench in the park"
- *Garden*: "A table with a vase on it in a garden"
- *Stump*: "An old, hollow tree stump covered in ivy lying on the forest floor"
- *Playground*: "A playground with slides and benches around it"
- *Truck*: "An old, light blue truck parked on a city street"

B.5. Comparisons and Ablation Details

Below we provide all the details needed to reproduce the comparisons and ablations shown in the paper.

Baseline Methods

ClimateNeRF. [16] We use the code provided by the authors (https://github.com/y-u-a-n-l-i/Climate_NeRF) with the default parameters.

GaussCtrl. [33] We use the code provided by the authors (https://github.com/ActiveVisionLab/gaussctrl) with the default parameters.

Ablation Details

w/o Collision Handling *(w/o CH)*. We disable the mesh-based collision handling techniques described in Sec. B.2. Dynamic Gaussians follow their simulated trajectories without surface projection or accumulation adjustments, resulting in Gaussians floating above surfaces rather than realistically interacting with scene geometry.

w/o Appearance Optimization *(w/o App)*. We skip the Video-SDS appearance optimization stage entirely and directly use the LLM-initialized parameters described in Sec. B.4. This variant uses only the baseline appearance parameters $\mathcal{A}_{init}^{active}$ and $\mathcal{A}_{init}^{collided}$ without any refinement through diffusion guidance.

w/o Motion Simulation *(w/o Motion)*. We bypass our physics-based motion simulation and instead use 4D Gaussian Splatting [32], with default parameters as in the code provided by the authors (https://github.com/hustvl/4DGaussians). We initialize a deformation field and optimize it directly using Video-SDS with the same prompts and optimization settings as our full method. This variant attempts to learn both motion and appearance simultaneously from the video diffusion model alone.

w/o Physics Guidance *(w/o PG)*. We fix the motion trajectories from physics simulation and optimize only the appearance matrices ($\mathcal{A}^{\text{active}}$, $\mathcal{A}^{\text{collided}}$) directly through Video-SDS. This variant eliminates both physics conditioning (the neural dynamics model does not receive velocities as input) and physics regularization losses ($\mathcal{L}_{\text{xyz}}$, $\mathcal{L}_{\text{vel}}$, $\mathcal{L}_{\text{rot}}$). The model predicts only appearance deltas ($\boldsymbol{\Delta}\mathcal{A}$) and does not predict velocity corrections ($\boldsymbol{\Delta}\mathbf{v}$) or angular velocities ($\boldsymbol{\omega}$), preventing joint optimization of motion and appearance. We retain only appearance regularization ($\mathcal{L}_{\text{app}}$) to prevent error accumulation in the recurrent predictions.

Evaluation Protocol

For quantitative evaluation of video-based metrics, we generate 15 camera trajectories per scene by interpolating between randomly sampled pairs of training viewpoints. We render videos along these trajectories for all methods and compute the metrics (ViCLIP-T, VE-Bench) on the resulting videos. For image-based metrics, we render images from all training viewpoints.

C. Additional Visualizations and Results

We present additional comparisons to the static 3D editing methods ClimateNeRF [16] and GaussCtrl [33] in Figure 11. We also show comparisons to GaussCtrl over the *sandstorm* and *rainfall* effects in Figure 12.



Figure 11. **Additional comparisons to static 3D editing techniques**. We show additional qualitative comparisons between static 3D editing methods and our dynamic approach. Only the *fog* (shown on the top row and the third row form the top) and *snowfall* (shown on the bottom row and the second row from the top) effects are shown, because ClimateNeRF only supports editing for these effects. We compare the original scene, ClimateNeRF [16], GaussCtrl [33], and our method at both early (Small t) and later (Large t) simulation timesteps.



Figure 12. **Additional Comparisons on the *sandstorm* and *rainfall* edits**. We show additional comparisons between our approach and GaussCtrl [33]. We show results for the *sandstorm* (shown on the bottom row and the second row from the top) and *rainfall* (shown on the top row and the second row from the bottom) effects. We compare the original scene, GaussCtrl [33], and our method at both early (Small t) and later (Large t) simulation timesteps. We note that ClimateNeRF [16] does not support editing for these effects. We zoom in on the results for *rainfall*, as these results are challenging to visualize.