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

Supplementary Material

6. Interactive Visualizations

We refer readers to the interactive visualizations at https://galfiebelman.github.io/let-it-snow/supp/index.html for full temporal sequences of dynamic weather effects, comparisons to 4D editing baselines, and comprehensive ablation studies.

7. Additional Ablations and Comparisons

4D Editing Baseline Comparison. We compare our method against Instruct-4DGS [16], 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 (quantitative results in Tab. 1 of the main paper), 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. 3, 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. 4, 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

Table 3. **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**

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. 5):

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

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. 5, 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 becomes 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.

Physics Prior Robustness. We evaluate the robustness of our framework to perturbations in key simulation parameters. We perturb four parameters by ±50%: emission rate, initial velocity, Young's modulus, and movement threshold δ. Experiments are conducted on all scenes across snowfall, rainfall, and sandstorm effects. As shown in Tab. 6, our method maintains stable performance across all perturbations and motion remains stable with average trajectory drift (Eq. 9) of 0.28 ± 0.03 compared to a baseline drift of 0.26 on these three effects.

Table 4. **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**

Table 5. **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**

Table 6. **Physics Prior Robustness**. Performance under ±50% perturbation of key simulation parameters (emission rate, initial velocity, Young's modulus, movement threshold), averaged across all scenes and three effects (snowfall, rainfall, sandstorm). Our method maintains stable performance across all perturbations.

Setting	CLIP$_{Sim}$ ↑	CLIP$_{Dir}$ ↑	VQAScore ↑	ViCLIP-T ↑	VE-Bench ↑
±50% Perturbation	0.27±0.02	0.10±0.01	0.87±0.04	0.17±0.01	0.38±0.03
Default (Ours)	**0.28**	**0.11**	**0.88**	**0.19**	**0.39**

Regularization Weight Sensitivity. We vary each regularization weight independently by $\times 0.1$ and $\times 10$ while keeping others at their default values. Experiments are conducted on all scenes across snowfall and sandstorm effects. As shown in Tab. 7, physics regularization weights (λ_{xyz}, λ_{vel}, λ_{rot}) are robust to order-of-magnitude changes, with metrics remaining within 0.01-0.02 of default values. Appearance regularization weights are more sensitive when reduced by $10\times$ but degrade gracefully when increased, confirming a stable operating range around our defaults.

Scheduler Comparison. We compare our SDS-adaptive scaling against three fixed regularization weight decay schedules on all scenes across snowfall and sandstorm effects: (1) linear decay (λ: $10^6 \rightarrow 10^2$ over 1000 iterations), (2) exponential decay (λ=$10^6 \cdot 0.995^{\text{iter}}$), and (3) cosine annealing (λ: $10^6 \rightarrow 10^2$ following a cosine curve). As shown in Tab. 8, our SDS-adaptive scaling outperforms all fixed schedules. Fixed schedules fail because Video-SDS loss

Figure 7. **Qualitative Comparison to 4D Editing Baseline**. We compare our method against Instruct-4DGS [16] 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.

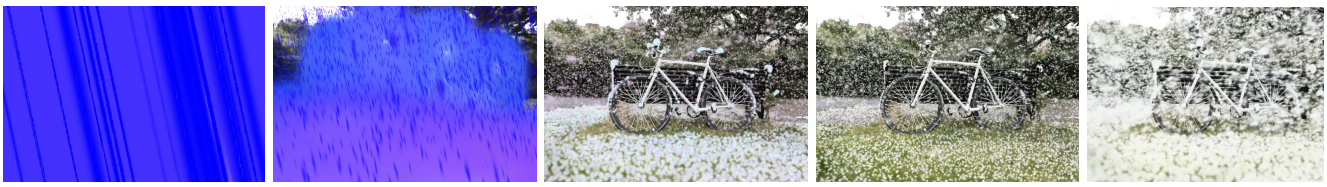

Figure 8. **SDS-Adaptive Weight Ablations**. Comparison of constant regularization weights versus our SDS-adaptive approach. λ=1 produces severe visual artifacts and noise due to insufficient regularization; λ=100 shows reduced but still significant noise and color distortions; λ=10^4 exhibits unnatural 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.

magnitude varies unpredictably during training, while our adaptive scaling strengthens physics guidance when SDS is uncertain and relaxes it when the model is confident.

Per-Particle Appearance and Rotation Ablation. We ablate the necessity of per-particle appearance variation and angular velocity prediction. We evaluate two variants on all scenes across snowfall and sandstorm effects: (1) *fixed appearance*, where all active particles share one optimized appearance and all collided particles share another, and (2) *fixed rotation*, where the angular velocity prediction is disabled. The variance of learned parameters across particles, averaged over timesteps, scenes, and effects, confirms meaningful per-particle variation for appearance: $\text{Var}(\sigma)_{\text{active}}$=0.12, $\text{Var}(\mathbf{S})_{\text{active}}$=0.21, $\text{Var}(\mathbf{C})_{\text{active}}$=0.05, $\text{Var}(\sigma)_{\text{collided}}$=0.01, $\text{Var}(\mathbf{S})_{\text{collided}}$=0.03, $\text{Var}(\mathbf{C})_{\text{collided}}$=0.005, while angular velocity variance is near-zero ($\text{Var}(\omega)\approx$0).

Tab. 9 demonstrates that removing per-particle appearance clearly degrades performance across all metrics, confirming its importance. Fixed rotation shows no degradation, consistent with the near-zero angular velocity variance ($\text{Var}(\omega)\approx$0). We retain this capacity for generality and for effects where rotation may be more prominent.

SDS Variant Comparison. We evaluate two improved score distillation variants to assess compatibility with our framework, Noise-Free Score Distillation (NFSD) [13],

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

Table 7. **Regularization Weight Sensitivity**. Each weight is varied independently by $\times 0.1$ and $\times 10$ on all scenes across snowfall and sandstorm effects. Physics weights are robust to order-of-magnitude changes. Appearance weights are sensitive when reduced but degrade gracefully when increased.

Weight	Value	$\text{CLIP}_{Sim}\uparrow$	$\text{CLIP}_{Dir}\uparrow$	VQAScore ↑	ViCLIP-T ↑	VE-Bench ↑
λ_{xyz} (0.1)	×0.1	0.26	0.09	0.84	0.16	0.35
	×10	0.27	0.10	0.86	0.17	0.37
λ_{vel} (0.1)	×0.1	0.26	0.10	0.85	0.16	0.37
	×10	0.27	0.10	0.86	0.17	0.39
λ_{rot} (0.1)	×0.1	0.28	**0.11**	0.88	**0.19**	0.39
	×10	0.28	**0.11**	0.88	**0.19**	0.41
$\lambda_{\sigma}^{\text{active}}$ (1.0)	×0.1	0.24	0.08	0.79	0.15	0.32
	×10	0.27	0.10	0.86	0.18	0.39
$\lambda_{\mathbf{S}}^{\text{active}}$ (1.0)	×0.1	0.19	0.04	0.52	0.09	0.19
	×10	0.26	0.10	0.83	0.17	0.37
$\lambda_{\mathbf{C}}^{\text{active}}$ (35.0)	×0.1	0.20	0.06	0.57	0.11	0.23
	×10	0.26	0.10	0.84	0.18	0.37
$\lambda_{\sigma}^{\text{collided}}$ (35.0)	×0.1	0.22	0.07	0.74	0.13	0.30
	×10	0.27	0.10	0.85	0.18	0.38
$\lambda_{\mathbf{S}}^{\text{collided}}$ (35.0)	×0.1	0.17	0.03	0.47	0.07	0.14
	×10	0.26	0.10	0.84	0.18	0.38
$\lambda_{\mathbf{C}}^{\text{collided}}$ (35.0)	×0.1	0.21	0.06	0.61	0.12	0.26
	×10	0.27	0.10	0.85	0.18	0.37
Default (Ours)		**0.29**	**0.11**	**0.89**	**0.19**	**0.42**

which decomposes the score into interpretable components and removes an undesired noise term to reduce over-smoothing, and Bridge-SDS [22], which formulates score distillation as optimal transport between source and target distributions to reduce oversaturation artifacts. We replace our vanilla SDS objective with each variant and evaluate on all scenes across snowfall and sandstorm effects, keeping all other components identical.

Table 8. **Scheduler Comparison**. Comparison of fixed regularization weight decay schedules versus our SDS-adaptive scaling on all scenes across snowfall and sandstorm effects. Our adaptive approach outperforms all fixed schedules.

Schedule	$\text{CLIP}_{Sim}\uparrow$	$\text{CLIP}_{Dir}\uparrow$	VQAScore ↑	ViCLIP-T ↑	VE-Bench ↑
Linear Decay	0.25	0.09	0.82	0.15	0.35
Exponential Decay	0.24	0.08	0.80	0.14	0.34
Cosine Annealing	0.26	0.10	0.84	0.17	0.38
Ours (SDS-Adaptive)	**0.29**	**0.11**	**0.89**	**0.19**	**0.42**

Table 9. **Per-Particle Appearance and Rotation Ablation**. Removing per-particle appearance degrades all metrics. Removing angular velocity shows no degradation, consistent with near-zero learned rotation variance for weather effects.

Variant	$\text{CLIP}_{Sim}\uparrow$	$\text{CLIP}_{Dir}\uparrow$	VQAScore ↑	ViCLIP-T ↑	VE-Bench ↑
Fixed Appearance	0.26	0.09	0.83	0.17	0.36
Fixed Rotation	**0.29**	**0.11**	0.88	**0.19**	**0.43**
Ours	**0.29**	**0.11**	**0.89**	**0.19**	0.42

As shown in Tab. 10, both improved variants yield higher scores across all metrics, with Bridge-SDS achieving the strongest results. While our physics-guided optimization drives the primary quality improvements over baselines (Tab. 1), these results confirm that orthogonal advances in

Table 10. **SDS Variant Comparison**. Our framework is compatible with improved score distillation methods. Both NFSD [13] and Bridge-SDS [22] improve over vanilla SDS when integrated into our physics-guided optimization.

SDS Variant	CLIP_{Sim} ↑	CLIP_{Dir} ↑	VQAScore ↑	ViCLIP-T ↑	VE-Bench ↑
Vanilla SDS	0.29	0.11	0.89	0.19	0.42
NFSD	0.30	0.11	0.91	0.20	0.44
Bridge-SDS	**0.32**	**0.12**	**0.93**	**0.22**	**0.45**

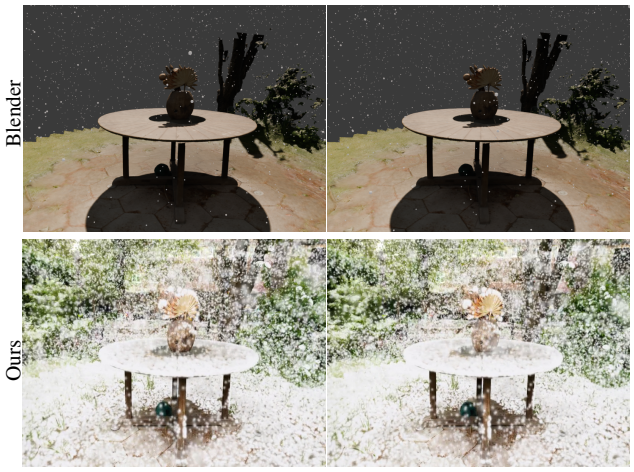

Figure 11. **Comparison to Manual VFX**. Two consecutive frames of a snowfall effect on the Garden scene created manually in Blender [3] (∼10 hours, top) versus our automated pipeline (∼1 hour, bottom). Our automated pipeline produces results of comparable visual quality while requiring an order of magnitude less time and no manual expertise.

score distillation provide additional gains when integrated into our framework.

Comparison to Manual VFX. To contextualize the quality-automation tradeoff, we created an equivalent snowfall effect in Blender [3] on the Garden scene. As a novice user, manual setup in Blender required approximately 10 hours (particle system configuration, material setup, lighting adjustment, rendering), compared to our automated pipeline of approximately 1 hour. A visual comparison is provided in Fig. 11. Our results are comparable or better than the manually created Blender effect, suggesting that little to no realism is lost for the sake of automation while requiring significantly less time and expertise.

8. Additional Details

8.1. Implementation Details

For each scene we first use 3D Gaussian Splatting [14] 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 [9] 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 [42]. 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.

For the simulation, we base our MPM simulation on Taichi [8], which supports various types of materials (e.g., *snow*, *water*, *sand*, etc.). We build on [25], 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. 8.3). 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 collided Gaussians, the model predicts only appearance deltas as their motion is fixed at collision states.

We employ Video Score Distillation Sampling using CogVideoX-2B [38] 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 instanta-

neous 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. A runtime comparison against baselines is provided in Tab. 11.

Table 11. **Runtime Comparison**. Per scene-effect runtime on a single NVIDIA A100 40GB GPU.

Method	Runtime
ClimateNeRF [18]	∼180 min
GaussCtrl [36]	∼30 min
Instruct-4DGS [16]	∼45 min
Ours	∼60 min

8.2. Collision Handling Details

Following Sec. 3.2, we describe our mesh-based collision handling.

***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 [42]. 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 [18] 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.

8.3. Motion Simulation Parameters

Following Sec. 3.2, we detail our 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.

8.4. Appearance Initialization

Following Sec. 3.3, 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"

8.5. Comparisons and Ablation Details

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

Baseline Methods

ClimateNeRF. [18] We use the code provided by the authors (https://github.com/y-u-a-n-l-i/Climate_NeRF) with the default parameters.

GaussCtrl. [36] 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. 8.2. 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. 8.4. 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 [35], 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 ($\mathbf{\Delta}\mathcal{A}$) and does not predict velocity corrections ($\mathbf{\Delta v}$) 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.

9. Additional Visualizations and Results

We present additional comparisons to the static 3D editing methods ClimateNeRF [18] and GaussCtrl [36] in Figure 12. We also show comparisons to GaussCtrl over the *sandstorm* and *rainfall* effects in Figure 13.

Figure 12. **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 [18], GaussCtrl [36], and our method at both early (Small t) and later (Large t) simulation timesteps.

Figure 13. **Additional Comparisons on the *sandstorm* and *rainfall* edits**. We show additional comparisons between our approach and GaussCtrl [36]. 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 [36], and our method at both early (Small t) and later (Large t) simulation timesteps. We note that ClimateNeRF [18] does not support editing for these effects. We zoom in on the results for *rainfall*, as these results are challenging to visualize.