EXPLORATION MATTERS FOR ESCAPING THE BLUR TRAP IN 3D GAUSSIAN SPLATTING

Anonymous authors
Paper under double-blind review

ABSTRACT

3D Gaussian Splatting (3DGS) employs Gaussian primitives for explicit scene representation, facilitating real-time, high-fidelity reconstruction and novel view synthesis of complex scenes. However, the explicit modeling inherent in 3DGS introduces a gradient bias during optimization, rendering its non-convex optimization process highly susceptible to convergence toward local suboptimal solutions. ***This constitutes a fundamental limitation in 3DGS optimization, which we term the Blur Trap***. To address this limitation, we integrate simple explicit exploration into the 3DGS optimization framework. First, through rigorous mathematical analysis of the 3DGS optimization formulation, we identify the underlying optimization bias responsible for the Blur Trap and categorize it into two distinct subtypes: the Far-Side Blur Trap and the Near-Side Blur Trap. Subsequently, we propose two highly straightforward exploration strategies (*Random Seeding* and *Random Splitting*) to mitigate the far-side and near-side blur traps, respectively. Experimental validation demonstrates that the incorporation of these exploration operators effectively and complementarily overcome the Blur Trap, achieving high-quality rendering performance across multiple datasets.

1 INTRODUCTION

3D Gaussian Splatting (3DGS) (Kerbl et al., 2023) has rapidly become a leading approach for novel view synthesis, offering real-time rendering together with photorealistic fidelity through an explicit representation built from anisotropic Gaussian primitives. Unlike implicit neural representations such as NeRF (Mildenhall et al., 2021), 3DGS owes much of its efficiency to a physically grounded design choice: each Gaussian is updated by gradients derived directly from the camera projection model, with no intermediate neural network in the loop. This design has driven impressive progress on standard benchmarks. Yet a recurring failure pattern can be observed across 3DGS reconstructions. ***Certain regions in the rendered scene remain visibly blurry no matter how many training views are provided.*** As shown in Figure 1, this is not random noise. It appears consistently in two kinds of regions: distant content such as mountains and skylines, and occluded near-field regions such as the grass behind the rack. The persistence of these artifacts even under abundant supervi-

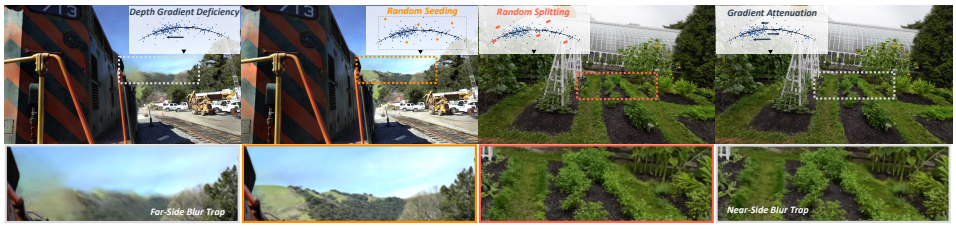

Figure 1: **Escaping the Blur Trap via Stochastic Exploration.** We identify the *Blur Trap* as a fundamental limitation intrinsic to 3DGS, rooted in an inherent exploitation-only optimization bias. It manifests as two variants: the Far-Side Blur Trap triggered by depth-gradient deficiency and the Near-Side Blur Trap driven by 2D-gradient attenuation. To break these, we introduce minimal exploration operators: *Random Seeding* and *Random Splitting*. These straightforward interventions bypass gradient-dependent optimization, consistently recovering structural details and proving that simple explicit exploration is highly effective for escaping Blur Trap without complex modifications.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

sion indicates that the cause is not insufficient data but something rooted in the optimization process itself. We refer to this systematic failure as the **Blur Trap**.

To understand why the Blur Trap forms, we trace the issue back to the gradient that drives 3DGS optimization. The 3D position update of each Gaussian is composed of three branches that correspond to the 2D projected position, the 2D covariance, and the spherical harmonic coefficients. Empirically, the 2D position component dominates the other two by two to three orders of magnitude across the entire training process, so ***the 3D position update is effectively governed by 2D reprojection error alone***. We then ask what this dominant component looks like geometrically. Through a careful derivation of the camera projection pipeline, we prove that ***the 3D position update derived from the 2D gradient is strictly orthogonal to the viewing ray from the camera center to the Gaussian***. The practical consequence is that the optimizer moves primitives within the plane perpendicular to the viewing ray, and only negligibly along the depth direction of the viewing ray. Compounding this, the same 2D gradient also serves as the criterion for adaptive densification, and the α-blending pipeline systematically attenuates this signal for occluded Gaussians, since ***later-ranked primitives in the depth-sorted sequence receive transmittance-weighted gradients that decay rapidly***. The Blur Trap therefore decomposes into two subtypes. The *Far-Side Blur Trap* arises in distant regions where there is no depth-directed optimization signal to drive Gaussians to their correct depth. The *Near-Side Blur Trap* arises in occluded foreground regions where the densification trigger is suppressed before it can fire.

Existing efforts to improve 3DGS have largely circled around the Blur Trap without naming or resolving it. A first line of work refines the densification criterion. AbsGS (Ye et al., 2024) addresses gradient direction conflicts among sub-pixel signals, Pixel-GS (Zhang et al., 2024) reweights gradients by pixel coverage, and similar variants (Mallick et al., 2024; Fang & Wang, 2024) adjust when and how clone or split operations fire. These methods all consume the same input: a gradient signal that already carries the two biases described above. Adjusting the threshold or the weighting does not change the fact that the signal itself lacks depth information and is attenuated by α-blending. A second line of work, exemplified by HoGS (Liu et al., 2025), reparameterizes Gaussian positions in homogeneous coordinates to better express far-field content. This changes how distant primitives are represented but does not introduce any new optimization signal in the depth direction, so the underlying gradient deficiency persists. A third line of work, most notably 3DGS-MCMC (Kheradmand et al., 2024), injects Langevin-style noise into Gaussian positions and reinterprets the entire pipeline as MCMC sampling. This is the closest in spirit to our perspective. However, the noise there is applied uniformly across the scene and is not tied to a specific diagnosis of why 3DGS fails. As a result, the method treats all primitives with the same exploratory mechanism, even though, as we show, the failure has two distinct geometric origins that call for two different remedies.

These observations suggest a simple reframing. 3DGS optimization is fundamentally a high-dimensional non-convex problem, and the broader optimization literature has long recognized that escaping local optima in such landscapes requires some form of exploration, whether through stochastic gradient noise (Welling & Teh, 2011; Jin et al., 2017), perturbed gradient descent (Ge et al., 2015), or simulated annealing. Standard neural network training enjoys several natural sources of such exploration, including the noise inherent in minibatch stochastic gradients and implicit regularization from dropout or weight decay. 3DGS enjoys almost none of these. Training proceeds with a batch size of one, the update rule contains no implicit regularization, and as established above, the physically derived gradient is itself structurally biased in ways that prevent the optimizer from probing either depth or occluded regions. From this vantage point, the Blur Trap is not a surprise but the predictable outcome of running a deterministic, gradient-dominated optimizer on a landscape whose dominant signal cannot point in the directions that matter. The natural remedy is to put exploration back in, and to design it so that it directly compensates for what the gradient cannot supply.

Guided by this principle, we introduce two minimal exploration operators, each tied to one subtype of the Blur Trap. **Random Seeding** addresses the Far-Side Blur Trap by periodically injecting candidate Gaussians at randomly sampled 3D positions across the scene. This bypasses the orthogonality constraint, since new primitives can land at depths that no existing Gaussian could have reached through gradient updates alone. Seeds that fall in geometrically invalid regions are removed by standard opacity pruning, and seeds that land in plausible regions are picked up by the regular optimization loop and refined. **Random Splitting** addresses the Near-Side Blur Trap by periodically splitting a random subset of Gaussians regardless of their accumulated 2D gradient, thereby bypassing the densification gate that α-blending suppresses. Both operators are deliberately simple.

We choose this minimal design because our goal is to verify that explicit exploration, on its own, is what 3DGS optimization is missing. A more sophisticated exploration strategy is certainly possible, but the burden of proof should first rest on the principle itself rather than on engineering.

Experiments on Mip-NeRF 360, Tanks & Temples, Deep Blending, OMMO and DL3DV show that each operator resolves its targeted form of blur and that the two are complementary. The combined operators consistently improve rendering fidelity over the 3DGS baseline at a training cost below that of the competing baselines. The advantage grows with scene scale: on the 135 scenes of DL3DV, they beat 3DGS and the exploration-based 3DGS-MCMC (Kheradmand et al., 2024) on PSNR, SSIM and LPIPS simultaneously on 100% and 78% of scenes respectively.

Contributions of this paper can be summarized as:

1. We identify and formalize the **Blur Trap**, a fundamental optimization limitation of 3DGS that arises from gradient biases inherent to its physically derived backpropagation. We further categorize the Blur Trap into two subtypes, the Far-Side Blur Trap and the Near-Side Blur Trap, each rooted in a different facet of the gradient bias.
2. We provide a rigorous theoretical analysis of the Blur Trap. In particular, we prove that the 3D position update derived from the 2D gradient is orthogonal to the viewing ray, which directly explains the absence of depth-directed optimization, and we characterize how α-blending systematically attenuates the densification signal for occluded primitives.
3. We propose two minimal exploration operators, **Random Seeding** and **Random Splitting**, targeting the Far-Side Blur Trap and Near-Side Blur Trap respectively. The minimal design is intentional, isolating the contribution of exploration itself from incidental engineering choices.
4. Extensive experiments on five different datasets demonstrate that each operator resolves its targeted form of blur, and that the two operators are complementary, jointly establishing exploration as a missing and essential ingredient in 3DGS optimization.

2 Related Work

2.1 Exploitation-Driven Gaussian Splatting Optimization

Adaptive Density Control. Adaptive density control (ADC) is the cornerstone of 3D Gaussian Splatting (3DGS), growing and pruning primitives to trade geometric fidelity against cost. It rests on one signal, the periodically averaged view-space positional gradient. A first line of work remedies the criterion while leaving the signal in place. AbsGS (Ye et al., 2024) replaces the summed gradient with a homodirectional one to resolve collisions among sub-pixel contributions. GDAGS (Zhou et al., 2026) measures directional coherence instead, separating over-reconstructed from under-densified regions. Pixel-GS (Zhang et al., 2024) normalizes gradients to remove scale bias.

Modeling Distant Content in Unbounded Scenes. Far-field content imposes a second constraint, with scales and viewing angles outside bounded, object-centric parameterizations. HoGS (Liu et al., 2025) answers by reparameterization, using homogeneous coordinates for position and scale to unify near-far representation. Scaffold-GS (Lu et al., 2024) and Proxy-GS (Gao et al., 2026) answer with structural priors, preventing redundant growth behind occluders.

These methods refine the criterion, the budget, or the parameterization, yet all of their densification fires on a periodically averaged 2D positional gradient, which carries gradient biase introduced by the physical projection model. None reduces or bypasses that dependency. The failures they suppress are symptoms of the biases, and none steps outside the deterministic exploit loop.

2.2 Exploration in Machine Learning

Exploration as a learning principle. Exploration is a recurring theme wherever a greedy update rule meets a complex landscape: the rule commits prematurely to whatever direction the local signal indicates, missing solutions hidden behind barriers or in under-visited regions. Simulated annealing (Kirkpatrick et al., 1983), Stochastic Gradient Langevin Dynamics (SGLD; Welling & Teh, 2011) and perturbed gradient descent (Ge et al., 2015; Jin et al., 2017) all instantiate this idea, and reinforcement learning treats exploration as a first-class algorithmic problem rather than an auxiliary mechanism (Sutton et al., 1998).

Two families of exploration mechanisms. These mechanisms divide into two complementary families. *Noise-based* methods inject stochasticity directly into the update rule, from ϵ-greedy action selection to parameter-space perturbation of network weights (Plappert et al., 2017; Fortunato et al., 2018). *Coverage-driven* methods instead steer exploration by explicit estimates of novelty or uncertainty, such as count-based bonuses (Bellemare et al., 2016), prediction-error curiosity (Pathak et al., 2017) and random network distillation (Burda et al., 2018). The two are not interchangeable. The former refines locally around the current state. The latter allocates effort toward regions the model has never reached (Osband et al., 2019), so which is better depends on the failure mode at hand.

Exploration in 3DGS optimization. Despite this rich literature, exploration remains a niche thread in 3DGS, where mainstream training follows a deterministic densify-and-prune cycle from a Structure-from-Motion initialization (Kerbl et al., 2023). 3DGS-MCMC (Kheradmand et al., 2024) is the closest predecessor of our perspective: it reinterprets Gaussians as samples from a distribution and converts position updates into SGLD steps via Langevin noise. A contemporaneous work, Opt3DGS (Huang et al., 2026), makes the exploration–exploitation framing explicit by alternating an adaptive SGLD phase with a curvature-aware exploitation phase. Both apply a single, uniform perturbation across all Gaussians without distinguishing between distinct failure modes. In contrast, we trace the difficulty back to two specific gradient biases identified in Section 3.3, and design two minimal operators each tied to one bias: **Random Seeding** acts as a coverage-driven mechanism that introduces new primitives in depth intervals unreachable from existing Gaussians, while **Random Splitting** acts as a noise-based mechanism that bypasses the densification gate suppressed by α-blending. Our work thereby treats exploration in 3DGS as a diagnosis-driven design problem rather than a uniform perturbation strategy.

3 Understanding the Blur Trap in 3D Gaussian Splatting

The Blur Trap follows from two properties of the 3DGS optimizer: its positional update is dominated by screen-space reprojection (Section 3.1), and this gradient lacks a depth-directed component (Section 3.2). Their combination produces the failure mode characterized in Section 3.3.

3.1 Screen-Space Gradient Dominance in Optimization

3D Gaussian Splatting (3DGS) reconstructs 3D scenes through a differentiable rendering framework. The forward pass builds a rendering pipeline via a ***physical camera model***, producing pixel colors through α-blending. Crucially, backpropagation derives gradients through this physical model, enabling direct optimization of Gaussian attributes against the image-based loss. Among these attributes, the 3D position of each Gaussian primitive is paramount: it determines where the primitive projects in screen space, governs how it blends with neighboring Gaussians, and ultimately defines the geometric structure of the reconstructed scene (Kotovenko et al., 2026; Lan et al., 2025). The 3D position gradient of a Gaussian, $\mathbf{g}_{\text{all}}$, decomposes into three components:

$$\mathbf{g}_{\text{all}} = \mathbf{g}_{\text{2d}} + \mathbf{g}_{\text{cov2d}} + \mathbf{g}_{\text{sh}}, \tag{1}$$

where $\mathbf{g}_{\text{2d}}$ originates from the 2D screen-space projection, $\mathbf{g}_{\text{cov2d}}$ stems from the 2D covariance governing the image-plane footprint, and $\mathbf{g}_{\text{sh}}$ derives from the spherical harmonic coefficients encoding view-dependent appearance. $\mathbf{g}_{\text{2d}}$ drives the displacement that reduces reprojection error, while $\mathbf{g}_{\text{cov2d}}$ and $\mathbf{g}_{\text{sh}}$ supply the geometric and appearance signals.

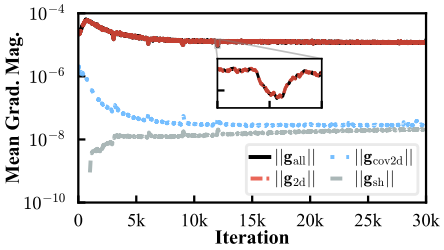

Figure 2: **Position gradient magnitude analysis.** $\|\mathbf{g}_{\text{2d}}\|$ stays almost identical to $\|\mathbf{g}_{\text{all}}\|$, while $\|\mathbf{g}_{\text{cov2d}}\|$ and $\|\mathbf{g}_{\text{sh}}\|$ are 2–3 orders of magnitude smaller.

These three components differ drastically in magnitude. As shown in Figure 2, $\mathbf{g}_{\text{2d}}$ is consistently orders of magnitude larger than $\mathbf{g}_{\text{cov2d}}$ and $\mathbf{g}_{\text{sh}}$. This disparity is structural: the 2D position gradient directly captures reprojection sensitivity to spatial displacements, whereas the covariance and SH gradients derive from less direct transformations. Consequently, $\mathbf{g}_{\text{2d}}$ dominates parameter updates, reducing the three-component optimization to greedily minimizing 2D reprojection error. To verify this dominance, we ablate individual gradient components during primitive updates. Retaining $\mathbf{g}_{\text{2d}}$ matches

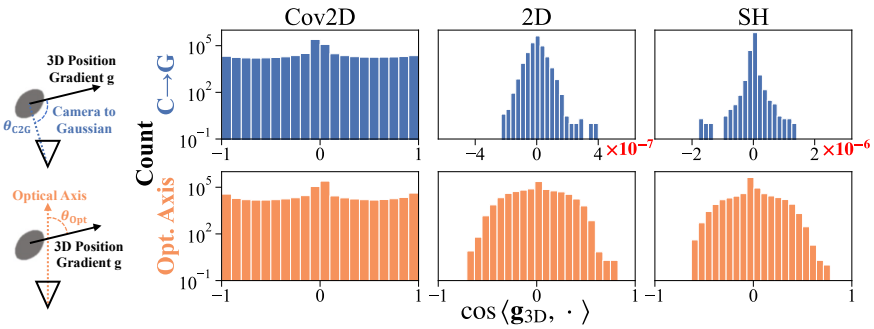

Figure 3: **Single-view analysis: orthogonality of the position-gradient branches.** Cosine similarity between each branch of the 3D position gradient and two reference directions, measured over the training run. **Top:** against the Camera-to-Gaussian ray, over all primitives. **Bottom:** against the optical axis, restricted to primitives inside the field of view. ($\theta \in [60°, 120°] \rightarrow \cos\theta \in [-0.5, 0.5]$)

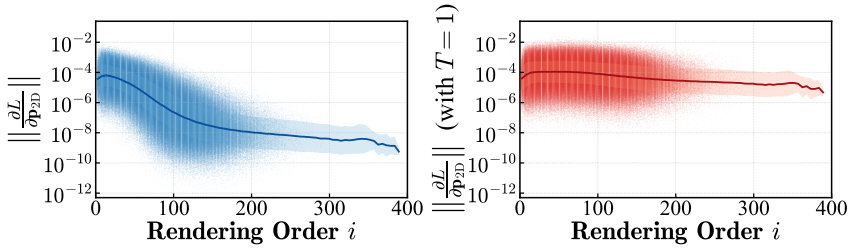

Figure 4: **Transmittance dependence of the 2D gradient magnitude.** Scatter of the 2D gradient magnitude $\|\frac{\partial L}{\partial \mathbf{p}_{2D}}\|$ against the rendering order i, over all pixels of the whole training process. **Left:** under standard backpropagation, $\|\frac{\partial L}{\partial \mathbf{p}_{2D}}\|$ decays rapidly as i increases. **Right:** recomputing the same gradient with T_i fixed to 1 eliminates the decay.

standard 3DGS, whereas excluding it degrades quality sharply; removing $\mathbf{g}_{\text{cov2d}}$ or $\mathbf{g}_{\text{sh}}$ individually is close to inconsequential (Appendix B.1). The optimization is therefore driven by screen-space positional signals alone, with the remaining components marginal.

In summary, although the 3D position update formally combines three gradient contributions, **their severe magnitude disparity degenerates the optimization into a greedy, 2D screen-space gradient exploitation process.** The optimizer minimizes reprojection errors through $\mathbf{g}_{\text{2d}}$ while effectively ignoring the complementary geometric and appearance signals from $\mathbf{g}_{\text{cov2d}}$ and $\mathbf{g}_{\text{sh}}$. As a result, Gaussian primitives rapidly converge to configurations that locally minimize pixel error. This is a classic exploitation behavior that prioritizes immediate screen-space fidelity at the expense of global structural integrity. ***This analytical insight naturally leads to the central question: does such a greedy 2D optimization strategy introduce systematic reconstruction failures in 3DGS?***

3.2 Depth Gradient Deficiency and Blending-Induced Densification Failure

The 2D position gradient in 3DGS fulfills a dual role: it drives 3D positional updates and serves as the core criterion for adaptive density control (ADC). Consequently, this single optimization branch exerts disproportionate control over the entire reconstruction pipeline. Both roles are compromised: the positional update carries no depth-directed component, and the densification signal is diluted by α-blending before it can trigger a split. *All derivations are given in Appendix A.*

3.2.1 Depth Gradient Deficiency

We prove that **the component of the 3D position gradient contributed by the 2D position branch is strictly orthogonal to the viewing ray from the camera center to the primitive**:

$$\mathbf{g}_{\text{2d}} \cdot (\mathbf{P}_{\text{3D}} - \mathbf{P}_{\text{cam}}) = 0, \tag{2}$$

where $\mathbf{g}_{\text{2d}}$ is the 2D-branch component of the 3D position gradient. $\mathbf{P}_{\text{3D}}$ is the Gaussian's 3D position and $\mathbf{P}_{\text{cam}}$ the camera center.



Figure 5: **The Blur Trap under dense view coverage.** Each panel pairs a rendered crop with the training-view distribution that supervises it. **Left:** distant content stays structurally smoothed although the dashed region receives dense coverage. **Right:** the same coverage still leaves detail missing in an occluded near-field region.

A direct consequence is that a primitive is free to move within the plane orthogonal to its ray, but not along it. We validate this both empirically and geometrically. Figure 3 shows that the 2D position branch stays orthogonal to the Camera-to-Gaussian ray and clusters tightly around zero against the optical axis; since this branch also dominates the gradient magnitude , $\mathbf{g}_{\text{all}}$ inherits that orthogonality. Depth information therefore has to come from elsewhere: each view constrains a primitive only within the plane orthogonal to its own ray, so the ray direction is pinned solely by views taken from other angles. Angular diversity, not view count, is what supplies the depth constraint Figure 9; the illustration and the branch-wise summary are given in Appendix B. Distant content removes exactly this recourse: relative to a far-away primitive the camera trajectory subtends almost no parallax, so its viewing rays stay nearly parallel however many views are captured, and the radial component of its position gradient along $\boldsymbol{x}$ degenerates (Figure 10, Appendix B.3). Skylines and far terrain, which cannot be surrounded with viewpoints, are the extreme case (Liu et al., 2025). Depth-directed momentum is therefore absent by construction, not by insufficient supervision.

3.2.2 Blending-Induced Densification Failure

The 2D position gradient also serves as the screening metric for adaptive densification: a primitive is densified once its periodically accumulated magnitude exceeds τ_{split}.

$$C_p = \sum_{i=1}^{N} c_i \alpha_i T_i, \quad T_i = \prod_{j=1}^{i-1} (1 - \alpha_j) \tag{3}$$

Under this compositing rule, where i is a Gaussian's rank in the depth-sorted sequence, a primitive ranked late contributes little to the pixel colour and, in proportion, receives a weak gradient: the transmittance T_i weights its contribution to both the rendered output and the loss (Wang et al., 2025). **Because T_i decays rapidly with i, the 2D position signal available to occluded primitives is progressively suppressed.**

The suppression also crosses structures. Hundreds of primitives composite one pixel (Liu & Han, 2026), and they need not belong to one coherent surface (Gao et al., 2026), so the pixel is not consistent with the underlying geometry (Zhang et al., 2026; Chen et al., 2024). A background Gaussian hidden behind a different structure therefore shares that pixel and inherits the same attenuation. Figure 11 profiles one pixel at a single iteration (Appendix B.4), where the gradient falls with rendering order while the contributing primitives span three distinct depth intervals. Figure 4 repeats the measurement over all pixels of the whole training run and shows that the decay disappears when T_i is fixed to 1 during backpropagation (Equation 20). The attenuation is therefore systematic, and α-blending is what causes it. A primitive that is occluded in one view is typically foreground in others, and there it is optimized normally. The periodically accumulated gradient, however, sums over views: the near-zero values contributed by the views in which it is occluded are averaged in alongside its normal values, so the accumulated magnitude stays below τ_{split} and the split never fires. The occluded regions are thus not starved of training views but penalized by the ones in which they are hidden, which is the Near-Side variant of the Blur Trap.

3.3 Formation of the Blur Trap

Collectively, screen-space gradient dominance, depth gradient deficiency, and blending-induced densification failure establish an **exploitation-only optimization dynamic**, which we formally define as the **Blur Trap**: a systematic stagnation of 3DGS optimization in local optima intrinsic to the

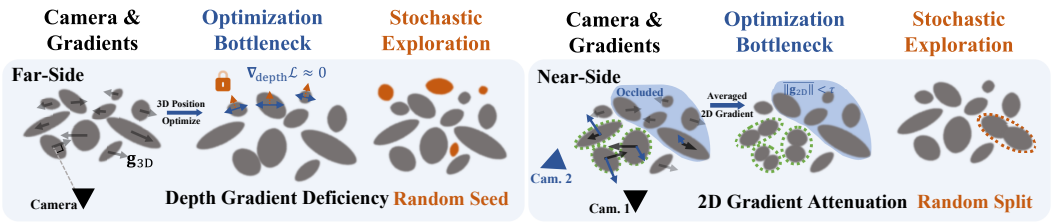

Figure 6: **Stochastic Exploration for Escaping the Blur Trap. Far-Side (Left):** The 3D position gradient $\mathbf{g}_{\text{all}}$ is approximately orthogonal to the viewing direction, so distant zones receive almost no update along the depth. *Random Seeding* injects sparse Gaussian primitives directly to probe unexplored depths. **Near-Side (Right):** In occluded regions, α-blending attenuation suppresses the per-view 2D position gradient, so its view average ($\overline{\|\partial L/\partial \mathbf{p}_{2D}\|}$) stays below the split threshold. Standard gradient-based splitting therefore never reaches the occluded primitives that need capacity, which *Random Splitting* restores by splitting them regardless of their gradient.

physically derived gradient, rather than a consequence of insufficient supervision or model capacity. Figure 5 shows the evidence that separates the two explanations: the blurring persists in distant and occluded regions even where the training views cover them densely.

Two variants follow from which degraded signal dominates. The *Far-Side Blur Trap* is the absence of any depth-directed component in the position gradient: primitives cannot converge to their correct locations along the viewing ray, so distant content stays structurally smoothed. The *Near-Side Blur Trap* is the attenuation of that same gradient under occlusion: a primitive that is occluded in some views still receives transmittance-weighted gradients from the occluded views, and the densification criterion averages those suppressed values together with the normal ones, which holds the accumulated magnitude below τ_{split} and keeps the local geometry at its suboptimal configuration.

Because the required depth-probing signal is suppressed by the rendering formulation itself rather than by the optimizer's state, deterministic gradient descent has no route out of this basin; escaping it requires explicit perturbations.

4 Methodology

Our design principle follows directly from the diagnosis of Section 3: to **bypass purely gradient-driven exploitation by injecting explicit exploration**, perturbing the optimization independently of the 2D gradient magnitude rather than waiting for the suppressed signal to recover. To validate the efficacy of exploration itself rather than of engineering refinements, we deliberately adopt the simplest operator for each requirement: *Random Seeding* for Far-Side trap and *Random Splitting* for Near-Side one. They can be enabled independently. Appendix C gives the full schedule.

Random Seeding As illustrated in Figure 6, the *Random Seeding* operator periodically samples a sparse set of seed Gaussians directly in 3D space and injects them into the existing primitive collection. These seeds are not displaced by gradients, so the ray orthogonality of Section 3.2.1 does not restrict where they can land: the operator places primitives at depths that gradient-based updates cannot reach. The sparse injection schedule leaves well-reconstructed regions largely undisturbed. Seeds that do not decrease the photometric loss are removed by the standard opacity pruning; the rest lie in geometrically plausible regions and are refined by the 2D reprojection loss. They move within the plane orthogonal to the viewing ray, eventually triggering ADC. This establishes a complementary optimization loop: ***Random Seeding* supplies global depth exploration, while deterministic gradients drive local planar exploitation**.

Random Splitting The *Random Splitting* operator bypasses the gradient-dependent densification criterion by targeting Gaussians for exploratory splitting regardless of their accumulated gradient. Specifically, we sample primitives with probability proportional to their mean scale and split at most N_{split} of them per iteration to preserve training stability. This strategy brings occluded regions into a densification process that gradient magnitudes would otherwise exclude. Once these regions are populated with finer primitives, the views in which they appear as foreground refine them normally.

Table 1: **Reconstruction quality across five datasets.** Red, orange and yellow mark the best, second-best and third-best value. *The same primitive budget as *Seed & Split Exploration*.

Method	Mip-NeRF360			Tanks & Temples			Deep Blending			OMMO			DL3DV Benchmark		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
3DGS	27.52	0.816	0.215	23.74	0.853	0.169	29.76	**0.907**	0.238	30.50	0.920	0.142	27.16	0.861	0.167
AbsGS	27.48	0.818	0.197	23.65	0.852	0.163	29.60	0.901	0.241	30.11	0.917	0.137	27.40	0.871	0.149
HoGS (50K Iter.)	27.55	0.818	0.201	24.23	0.859	0.160	29.21	0.898	0.244	30.66	0.919	0.135	28.16	0.878	0.140
GS-MCMC*	**28.04**	**0.831**	**0.191**	24.28	0.865	0.149	29.42	0.895	0.266	30.42	0.921	0.138	28.13	0.886	0.132
Seed Explore	27.70	0.818	0.214	24.30	0.860	0.163	29.78	0.906	**0.237**	30.85	0.924	0.131	27.75	0.868	0.155
Split Explore	27.95	0.829	0.192	24.30	0.868	0.141	30.01	**0.907**	0.248	**31.29**	0.928	0.121	28.45	0.888	0.126
Seed & Split Explore	27.96	0.829	0.192	**24.37**	**0.871**	**0.138**	**30.03**	**0.907**	0.249	31.27	**0.930**	**0.119**	**28.46**	**0.889**	**0.125**



Figure 7: **Complementarity of Random Seeding and Random Splitting.** *Random Seeding* clears the far-side blur, *Random Splitting* the near-side blur. *Random Splitting* alleviates the Far-Side Blur Trap but does not eliminate it. The distant hillside and the cloud remain smoothed, and are restored only when seeding is also enabled.

Random Splitting thus restores densification in occluded regions by decoupling structural refinement from the transmittance-attenuated gradient of Section 3.2.2.

5 Experiments

5.1 Experimental Settings

Implementation Details. We integrate both operators into standard 3DGS under minimal intervention: *Random Seeding* injects a small number of candidate positions at fixed intervals within the bounding box of the existing Gaussians, and *Random Splitting* splits a small number of large-scale Gaussians. Both are fixed-rate and non-adaptive, and the full schedule, hyperparameters and training loop are given in Appendix C.

Dataset and Metrics. We comprehensively evaluate our method across multiple 3D reconstruction benchmarks, including Mip-NeRF 360 (Barron et al., 2022), Tanks & Temples (Knapitsch et al., 2017), Deep Blending (Hedman et al., 2018), OMMO (Lu et al., 2023), and DL3DV Benchmark (Ling et al., 2024). We randomly select a subset of scenes from the DL3DV Benchmark for detailed evaluation with $\frac{1}{4}$ resolution images [1]. Reconstruction performance is quantified using PSNR, SSIM, and LPIPS. Additionally, the number of Gaussian primitives is monitored.

5.2 Experiments Results

Quantitative results across five datasets confirm the efficacy of both operators. Table 1 reports means over seeds 1–3; the standard deviations are given in Appendix D.1. *Seed & Split Exploration* is the best or tied-best of our variants on four of the five datasets, and *Seed Exploration* alone already improves every unbounded dataset while slightly reducing the primitive count.

On Deep Blending the ordering changes: *Seed & Split Exploration* attains the highest PSNR and SSIM, but the LPIPS is the worst. Appendix D.2 explains the trade: on this dataset splitting ends at 0.79M primitives, 32% of 3DGS. A third of the model size is worth a small perceptual margin, and we therefore read this cell as a trade.

[1] DL3DV Scene IDs: 1, 101, 104, 109, 11, 21, 24, 26, 34, 42, 69, 7, 8, 82, 90, 97

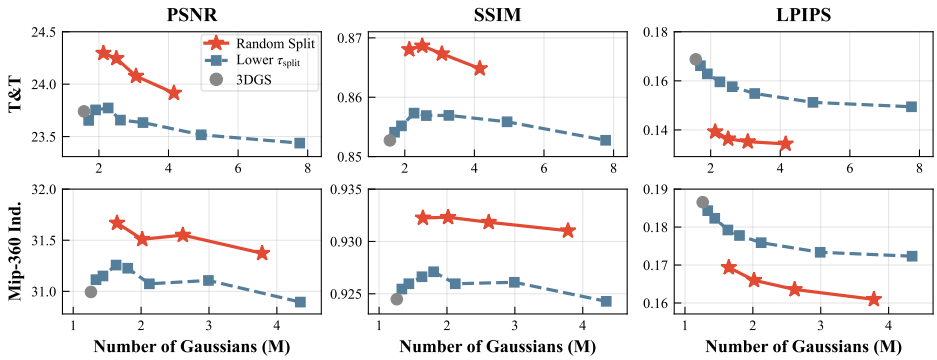

Figure 8: **Fidelity against primitive count.** The dashed line lowers the splitting threshold τ_{split} in standard 3DGS; the solid lines apply *Random Splitting* at varying N_{split}.

We further compare against GS-MCMC. Its opacity-based relocation moves primitives whose opacity falls below a threshold into high-opacity regions, implicitly bypassing gradient-gated densification and thus yielding a slight edge on near-side-dominated Mip-NeRF360. Yet both this relocation and its position noise are anchored to what the primitives have already reached: relocation targets occupied locations, and noise only perturbs locally. Under-explored regions, the sparsely initialized distant intervals where the Far-Side Blur Trap forms, therefore remain untouched (Figure 15). On unbounded scenes such as OMMO and DL3DV, GS-MCMC is consequently overtaken by *Seed & Split Exploration*.

Figure 7 and Table 13 decompose the effect by region. The deficit is located where Section 3 predicts: on DL3DV the farthest 5% of the depth range scores 0.35 dB below the rest under standard 3DGS, and both operators recover it. Seeding by 1.07 dB with 2% *fewer* primitives than 3DGS, splitting by 2.05 dB at 58% more. Their contributions are not additive ($+0.11$ dB far-side), as expected when distant content is both parallax-poor and frequently occluded. We explain in Appendix D.3.5 why splitting cannot place primitives at depths nothing occupies.

We compare *Random Splitting* directly against standard gradient-driven densification along the fidelity–capacity trade-off. We sweep *Random Splitting* over $N_{\text{split}} \in \{20, 50, 100, 200\}$, and separately lower the 2D position gradient threshold τ_{split} in 3DGS so that its own densification fires more readily. Figure 8 contrasts the two. Simply lowering τ_{split} brings no meaningful fidelity gain and instead introduces excessive redundant primitives. Sparse random splitting, by contrast, improves reconstruction quality while keeping a markedly leaner primitive allocation. The gain therefore comes from targeted structural exploration rather than from brute-force densification.

6 Conclusion

We identify the *Blur Trap* as a fundamental optimization limitation in 3DGS, rooted in viewing-ray gradient orthogonality and blending-induced gradient attenuation that systematically suppress depth-directed signals. Because both biases arise from the rendering formulation rather than from the optimizer's state, deterministic gradient descent has no route out of this exploitation-only deadlock. The deficiency takes two forms: the *Far-Side Blur Trap*, in which distant content receives no signal that can move it along the viewing ray, and the *Near-Side Blur Trap*, in which occluded geometry is never refined because the suppressed signal fails to trigger densification. To break the deadlock, we introduce two deliberately minimal exploration operators that bypass gradient-dependent criteria: *Random Seeding* places candidate primitives at depths gradient updates cannot reach, and *Random Splitting* refines occluded regions independently of the gradient. Experiments across five benchmarks show that each operator resolves the degradation it targets while their combination is complementary, improving fidelity over the 3DGS baseline at a training-time cost below that of the competing baselines. These results are consistent with the view that reconstruction bottlenecks stem from optimization dynamics rather than model capacity. Our work establishes explicit exploration as a necessary component for differentiable rendering, advocating a shift toward balanced exploration-exploitation dynamics.

AI USE STATEMENT

In this work, we used generative AI tools for one task with required disclosure: **translation and language polishing of individual sentences and paragraphs** of the author-written manuscript. All substantive content was produced by the authors without generative AI: the conceptual framework of the Blur Trap and its two subtypes, the hypotheses, the mathematical claims and their proofs, the experimental design and methodology, and the interpretation of the results were all developed and carried out by the authors. We have not used generative AI to generate synthetic datasets, to clean or reformat datasets, or to support qualitative or thematic data analysis, and the task of "help develop theoretical models or conceptual frameworks" does not apply to this work.

Additionally, we used generative AI tools for the following tasks with recommended disclosure: editing the manuscript for readability, and creating and editing code, specifically debugging support and the plotting scripts used to produce the figures. The two exploration operators were designed by the authors, and AI assistance was confined to implementation and debugging support.

All mathematical claims were independently re-derived by hand and matched against the closed-form expressions given in the appendix. The AI-assisted code was tested by the authors on all five datasets, and the numerical results it produced were checked against the values reported in the tables. All AI-assisted edits were reviewed by the authors to confirm that the intended meaning was preserved. We take responsibility for the final content of this work, including all text, claims, figures, and artifacts produced with the aid of generative AI.

REFERENCES

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*, pp. 5470–5479, 2022.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

Danpeng Chen, Hai Li, Weicai Ye, Yifan Wang, Weijian Xie, Shangjin Zhai, Nan Wang, Haomin Liu, Hujun Bao, and Guofeng Zhang. Pgsr: Planar-based gaussian splatting for efficient and high-fidelity surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 31 (9):6100–6111, 2024.

Guangchi Fang and Bing Wang. Mini-splatting: Representing scenes with a constrained number of gaussians. In *European conference on computer vision*, pp. 165–181. Springer, 2024.

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Matteo Hessel, Ian Osband, Alex Graves, Volodymyr Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. Noisy networks for exploration. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=rywHCPkAW.

Yuanyuan Gao, Yuning Gong, Yifei Liu, Jingfeng Li, Dan Xu, Yanci Zhang, Dingwen Zhang, Xiao Sun, and Zhihang Zhong. Proxy-gs: Unified occlusion priors for training and inference in structured 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7330–7339, 2026.

Rong Ge, Furong Huang, Chi Jin, and Yang Yuan. Escaping from saddle points—online stochastic gradient for tensor decomposition. In *Conference on learning theory*, pp. 797–842. PMLR, 2015.

Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (ToG)*, 37(6):1–15, 2018.

Ziyang Huang, Jiagang Chen, Jin Liu, and Shunping Ji. Opt3dgs: Optimizing 3d gaussian splatting with adaptive exploration and curvature-aware exploitation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 5230–5238, 2026.

Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M Kakade, and Michael I Jordan. How to escape saddle points efficiently. In *International conference on machine learning*, pp. 1724–1732. PMLR, 2017.

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

Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Yang-Che Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 3d gaussian splatting as markov chain monte carlo. *Advances in Neural Information Processing Systems*, 37:80965–80986, 2024.

Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.

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.

Dmytro Kotovenko, Olga Grebenkova, and Björn Ommer. Edgs: Eliminating densification for efficient convergence of 3dgs. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 41065–41076, 2026.

Lei Lan, Tianjia Shao, Zixuan Lu, Yu Zhang, Chenfanfu Jiang, and Yin Yang. 3dgs2: Near second-order converging 3d gaussian splatting. In *Proceedings of the Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*, pp. 1–10, 2025.

Tianye Li, Mira Slavcheva, Michael Zollhoefer, Simon Green, Christoph Lassner, Changil Kim, Tanner Schmidt, Steven Lovegrove, Michael Goesele, Richard Newcombe, et al. Neural 3d video synthesis from multi-view video. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 5521–5531, 2022.

Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, et al. Dl3dv-10k: A large-scale scene dataset for deep learning-based 3d vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 22160–22169, 2024.

Jiaqi Liu and Zhizhong Han. Speeding up the learning of 3d gaussians with much shorter gaussian lists. *arXiv preprint arXiv:2603.09277*, 2026.

Xinpeng Liu, Zeyi Huang, Fumio Okura, and Yasuyuki Matsushita. Hogs: Unified near and far object reconstruction via homogeneous gaussian splatting. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 26714–26722, 2025.

Chongshan Lu, Fukun Yin, Xin Chen, Wen Liu, Tao Chen, Gang Yu, and Jiayuan Fan. A large-scale outdoor multi-modal dataset and benchmark for novel view synthesis and implicit scene reconstruction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7557–7567, 2023.

Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-gs: Structured 3d gaussians for view-adaptive rendering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 20654–20664, 2024.

Saswat Subhajyoti Mallick, Rahul Goel, Bernhard Kerbl, Markus Steinberger, Francisco Vicente Carrasco, and Fernando De La Torre. Taming 3dgs: High-quality radiance fields with limited resources. In *SIGGRAPH Asia 2024 Conference Papers*, pp. 1–11, 2024.

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

Ian Osband, Benjamin Van Roy, Daniel J Russo, and Zheng Wen. Deep exploration via randomized value functions. *Journal of machine learning research*, 20(124):1–62, 2019.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pp. 2778–2787. PMLR, 2017.

Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter space noise for exploration. *arXiv preprint arXiv:1706.01905*, 2017.

Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Chengbo Wang, Guozheng Ma, Yifei Xue, and Yizhen Lao. Faster and better 3d splatting via group training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2025.

Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688, 2011.

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*, pp. 20310–20320, 2024.

Zongxin Ye, Wenyu Li, Sidun Liu, Peng Qiao, and Yong Dou. Absgs: Recovering fine details in 3d gaussian splatting. In *Proceedings of the 32nd ACM international conference on multimedia*, pp. 1053–1061, 2024.

Baowen Zhang, Chuan Fang, Rakesh Shrestha, Yixun Liang, Xiao-Xiao Long, and Ping Tan. Rade-gs: Rasterizing depth in gaussian splatting. *ACM Transactions on Graphics*, 45(2):1–14, 2026.

Zheng Zhang, Wenbo Hu, Yixing Lao, Tong He, and Hengshuang Zhao. Pixel-gs: Density control with pixel-aware gradient for 3d gaussian splatting. In *European Conference on Computer Vision*, pp. 326–342. Springer, 2024.

Zheng Zhou, Yu-Jie Xiong, Jia-Chen Zhang, Chun-Ming Xia, Xihe Qiu, and Hongjian Zhan. Gradient-direction-aware density control for 3d gaussian splatting. In *The Fourteenth International Conference on Learning Representations*, 2026. URL `https://openreview.net/forum?id=6qDxK4Gz7F`.

APPENDIX: PROOFS, ANALYSES AND ADDITIONAL RESULTS

A Ray Orthogonality and Transmittance Attenuation of the 2D Position Gradient **14**

A.1 Ray Orthogonality of the 2D Position Branch 14

A.1.1 Preliminaries 14

A.1.2 Projection-Path Gradient Decomposition 15

A.1.3 Proof of Orthogonality 16

A.1.4 Physical Interpretation 17

A.2 Transmittance Bound on the Depth-Sorted Position Gradient 17

A.2.1 m-Dependence of Each Factor 17

A.2.2 Derivation of Factor B $\partial \hat{\mathbf{c}}_c^{(p)} / \partial \alpha_m^{(p)}$ 18

A.2.3 Gradient Attenuation 18

B Mechanism Evidence **19**

B.1 Gradient-Branch Magnitudes and Directions 19

B.2 Component Ablation on the 2D Position Gradient 19

B.3 Multi-View Depth Constraints: Geometry and Statistics 19

B.4 Single-Pixel Profile of the Blending-Induced Attenuation 20

C Pseudocode and Hyper-parameters **20**

C.1 Notation 20

C.2 Algorithm Pseudocode 21

C.3 Oversized-Gaussian Pruning 21

D Experimental Results **22**

D.1 Main Results 23

D.2 Model Complexity 26

D.3 Ablations 27

D.3.1 Depth Supervision versus Random Seeding 27

D.3.2 Relaxed Pruning versus Random Seeding 27

D.3.3 Positional Perturbation along the Viewing Ray 28

D.3.4 Split Hyper-parameter Sweeps 28

D.3.5 Region-wise Effect of Each Exploration Mode 29

D.4 Qualitative Comparisons 30

D.4.1 Under-Explored Regions: Where GS-MCMC Falls Short 30

D.4.2 The Blur Trap Is Inherited by 4DGS 31

D.4.3 Per-Method Comparisons on Far and Near Regions 31

A Ray Orthogonality and Transmittance Attenuation of the 2D Position Gradient

Both results below concern the same object: the 2D position gradient branch, which drives the positional update and also serves as the densification criterion. Result A.1 shows that its direction carries no component along the camera-to-primitive ray, and Result A.2 that the blending-dependent factor of its magnitude is bounded by the accumulated transmittance of the primitives in front.

A.1 Ray Orthogonality of the 2D Position Branch

We prove that the 2D position gradient $\mathbf{g}_{2\mathrm{d}} = \frac{\partial L}{\partial \mathbf{p}_{2\mathrm{D}}} \frac{\partial \mathbf{p}_{2\mathrm{D}}}{\partial \mathbf{P}_{3\mathrm{D}}}$ is always orthogonal to the viewing direction $\mathbf{D}_{\mathrm{C2G}} = \mathbf{P}_{3\mathrm{D}} - \mathbf{P}_{\mathrm{cam}}$. This reveals a fundamental limitation: the 2D position update alone provides **no gradient along the line of sight**. The proof proceeds by decomposing $\mathbf{g}_{2\mathrm{d}}$ through the rendering pipeline via the chain rule and showing that the inner product $\mathbf{g}_{2\mathrm{d}} \cdot \mathbf{D}_{\mathrm{C2G}}$ collapses to zero through two geometric facts: (i) ${}^{h}\mathbf{p}$ lies in the null space of the perspective division Jacobian, and (ii) the viewport Jacobian annihilates the surviving z-component.

A.1.1 Preliminaries

Symbol Definition To rigorously derive the orthogonality property, we first establish the notation used throughout this proof. We denote the camera position in world coordinates as $\mathbf{P}_{\mathrm{cam}}$, with pose $[\mathbf{R}, \mathbf{t}]$. For a Gaussian primitive G, let $\mathbf{P}_{3\mathrm{D}}$ be its world-space position and $\mathbf{p}_{\mathrm{cam}}$ its camera-space position. The rendering pipeline transforms $\mathbf{P}_{3\mathrm{D}}$ through the following stages:

1. The view matrix $\mathbf{V}$ maps world coordinates to camera space: $[\mathbf{p}_{\mathrm{cam}}^{\mathrm{T}}, 1]^{\mathrm{T}} = \mathbf{V}[\mathbf{P}_{3\mathrm{D}}^{\mathrm{T}}, 1]^{\mathrm{T}}$.
2. The projection matrix $\mathbf{P}$ maps camera coordinates to homogeneous clip space: ${}^{h}\mathbf{p} = \mathbf{P}[\mathbf{p}_{\mathrm{cam}}^{\mathrm{T}}, 1]^{\mathrm{T}}$.
3. Perspective division yields Normalized Device Coordinates (NDC): $\mathbf{p}_{\mathrm{ndc}} = [\frac{p_x}{p_w}, \frac{p_y}{p_w}, \frac{p_z}{p_w}]^{\mathrm{T}}$.
4. The viewport transformation with image dimensions $W \times H$ produces 2D pixel coordinates $\mathbf{p}_{2\mathrm{D}}$.

The composite matrix $\mathbf{M} = \mathbf{PV}$ encodes the combined camera–projection mapping. These symbols are summarized below.

$$
\begin{aligned}
\mathbf{P}_{3\mathrm{D}} &= [x, y, z]^{\mathrm{T}} \\
\mathbf{p}_{\mathrm{cam}} &= [p_x^{\mathrm{G}}, p_y^{\mathrm{G}}, p_z^{\mathrm{G}}]^{\mathrm{T}} \\
{}^{h}\mathbf{p} &= [p_x, p_y, p_z, p_w]^{\mathrm{T}} \\
\mathbf{p}_{\mathrm{ndc}} &= [x_n, y_n, z_n]^{\mathrm{T}} \\
\mathbf{p}_{2\mathrm{D}} &= [u, v]^{\mathrm{T}} \\
\mathbf{M} &= \mathbf{PV} \\
\mathbf{V} &= \begin{bmatrix} R_{00} & R_{01} & R_{02} & t_x \\ R_{10} & R_{11} & R_{12} & t_y \\ R_{20} & R_{21} & R_{22} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
\mathbf{P} &= \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix}
\end{aligned}
\tag{4}
$$

Forward Mapping The forward mapping proceeds through four stages:

World to camera.

$$
\left[\mathbf{p}_{\mathrm{cam}}^{\mathrm{T}}, 1\right]^{\mathrm{T}} = \left[p_x^{\mathrm{G}}, p_y^{\mathrm{G}}, p_z^{\mathrm{G}}, 1\right]^{\mathrm{T}} = \mathbf{V}\left[\mathbf{P}_{3\mathrm{D}}^{\mathrm{T}}, 1\right]^{\mathrm{T}}, \tag{5}
$$

Camera to clip.

$$ {}^{h}\mathbf{p} = [p_x, p_y, p_z, p_w]^{\mathrm{T}} = \mathbf{M}\left[\mathbf{P}_{3\mathrm{D}}^{\mathrm{T}}, 1\right]^{\mathrm{T}} = \mathbf{P}\left[\mathbf{p}_{\mathrm{cam}}^{\mathrm{T}}, 1\right]^{\mathrm{T}}, \tag{6} $$

Clip to NDC.

$$ \mathbf{p}_{\mathrm{ndc}} = \begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix} = \begin{bmatrix} p_x/p_w \\ p_y/p_w \\ p_z/p_w \end{bmatrix}, \tag{7} $$

NDC to pixel.

$$ \mathbf{p}_{2\mathrm{D}} = \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} (x_n + 1) \cdot \frac{W}{2} \\ (y_n + 1) \cdot \frac{H}{2} \end{bmatrix}. \tag{8} $$

A.1.2 Projection-Path Gradient Decomposition

The gradient of the loss with respect to the 3D Gaussian position comprises three terms:

$$ \frac{\partial L}{\partial \mathbf{P}_{3\mathrm{D}}} = \frac{\partial L}{\partial \mathbf{Cov2D}} \frac{\partial \mathbf{Cov2D}}{\partial \mathbf{P}_{3\mathrm{D}}} + \frac{\partial L}{\partial \mathbf{p}_{2\mathrm{D}}} \frac{\partial \mathbf{p}_{2\mathrm{D}}}{\partial \mathbf{P}_{3\mathrm{D}}} + \frac{\partial L}{\partial \mathbf{SH}} \frac{\partial \mathbf{SH}}{\partial \mathbf{P}_{3\mathrm{D}}}. \tag{9} $$

These correspond to the optimization paths through the 2D covariance, the 2D screen-space position, and the spherical harmonics coefficients, respectively. Among them, the 2D position term dominates in magnitude; we therefore focus on this branch and decompose it via the chain rule into the loss derivative and the three Jacobians of Equation 10.

The 2D position gradient further decomposes via the chain rule through the rendering pipeline:

$$ \mathbf{g}_{2\mathrm{d}} = \underbrace{\frac{\partial L}{\partial \mathbf{p}_{2\mathrm{D}}}}_{\text{loss derivative}} \underbrace{\frac{\partial \mathbf{p}_{2\mathrm{D}}}{\partial \mathbf{p}_{\mathrm{ndc}}}}_{\text{viewport}} \underbrace{\frac{\partial \mathbf{p}_{\mathrm{ndc}}}{\partial {}^{h}\mathbf{p}}}_{\text{perspective}} \underbrace{\frac{\partial {}^{h}\mathbf{p}}{\partial \mathbf{P}_{3\mathrm{D}}}}_{\text{projection}} \tag{10} $$

The core factor $\frac{\partial \mathbf{p}_{2\mathrm{D}}}{\partial \mathbf{P}_{3\mathrm{D}}}$ is the product of three Jacobian matrices, which we now derive explicitly.

$$ \frac{\partial \mathbf{p}_{2\mathrm{D}}}{\partial \mathbf{p}_{\mathrm{ndc}}} = \begin{bmatrix} \frac{\partial u}{\partial x_n} & \frac{\partial u}{\partial y_n} & \frac{\partial u}{\partial z_n} \\ \frac{\partial v}{\partial x_n} & \frac{\partial v}{\partial y_n} & \frac{\partial v}{\partial z_n} \end{bmatrix} = \begin{bmatrix} \frac{W}{2} & 0 & 0 \\ 0 & \frac{H}{2} & 0 \end{bmatrix}. \tag{11} $$

$$ \begin{aligned} \frac{\partial \mathbf{p}_{\mathrm{ndc}}}{\partial {}^{h}\mathbf{p}} &= \begin{bmatrix} \frac{\partial x_n}{\partial p_x} & \frac{\partial x_n}{\partial p_y} & \frac{\partial x_n}{\partial p_z} & \frac{\partial x_n}{\partial p_w} \\ \frac{\partial y_n}{\partial p_x} & \frac{\partial y_n}{\partial p_y} & \frac{\partial y_n}{\partial p_z} & \frac{\partial y_n}{\partial p_w} \\ \frac{\partial z_n}{\partial p_x} & \frac{\partial z_n}{\partial p_y} & \frac{\partial z_n}{\partial p_z} & \frac{\partial z_n}{\partial p_w} \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{p_w} & 0 & 0 & -\frac{p_x}{p_w^2} \\ 0 & \frac{1}{p_w} & 0 & -\frac{p_y}{p_w^2} \\ 0 & 0 & \frac{1}{p_w} & -\frac{p_z}{p_w^2} \end{bmatrix}. \end{aligned} \tag{12} $$

$$ \begin{aligned} \frac{\partial {}^{h}\mathbf{p}}{\partial \mathbf{P}_{3\mathrm{D}}} &= \begin{bmatrix} \frac{\partial p_x}{\partial x} & \frac{\partial p_x}{\partial y} & \frac{\partial p_x}{\partial z} \\ \frac{\partial p_y}{\partial x} & \frac{\partial p_y}{\partial y} & \frac{\partial p_y}{\partial z} \\ \frac{\partial p_z}{\partial x} & \frac{\partial p_z}{\partial y} & \frac{\partial p_z}{\partial z} \\ \frac{\partial p_w}{\partial x} & \frac{\partial p_w}{\partial y} & \frac{\partial p_w}{\partial z} \end{bmatrix} \\ &= \begin{bmatrix} M_{00} & M_{01} & M_{02} \\ M_{10} & M_{11} & M_{12} \\ M_{20} & M_{21} & M_{22} \\ M_{30} & M_{31} & M_{32} \end{bmatrix} \\ &= \mathbf{M}_{[:,:3]} = \mathbf{P}\mathbf{V}_{[:,:\mathbf{3}]}. \end{aligned} \tag{13} $$

For the orthogonality proof we also need the world-space direction from the camera to the Gaussian, $\mathbf{P}_{3D} - \mathbf{P}_{cam}$. Applying $\mathbf{V}_{[:,:3]}$ to this direction and then $\mathbf{P}$ yields:

$$\mathbf{P}\mathbf{V}_{[:,:3]}(\mathbf{P}_{3D} - \mathbf{P}_{cam}) = \mathbf{P}\left[\mathbf{p}_{cam}^{T}, 0\right]^{T}, \tag{14}$$

which, after perspective division, gives the vanishing point of all lines parallel to the camera-to-Gaussian ray $\mathbf{p}_{cam}$ in camera space.

A.1.3 Proof of Orthogonality

We prove that $\mathbf{g}_{2d}$ is always orthogonal to the viewing direction from the camera center to the primitive. This orthogonality property is formalized in Equation 16.

Let $\mathbf{D}_{C2G} = \mathbf{P}_{3D} - \mathbf{P}_{cam}$ be the viewing direction from the camera to the Gaussian. Using the gradient decomposition from Equation 10, their inner product expands as

$$\begin{aligned}
\mathbf{g}_{2d} \cdot \mathbf{D}_{C2G} &= \mathbf{g}_{2d} \cdot (\mathbf{P}_{3D} - \mathbf{P}_{cam}) \\
&= \frac{\partial L}{\partial \mathbf{p}_{2D}} \frac{\partial \mathbf{p}_{2D}}{\partial \mathbf{p}_{ndc}} \frac{\partial \mathbf{p}_{ndc}}{\partial {}^{h}\mathbf{p}} \frac{\partial {}^{h}\mathbf{p}}{\partial \mathbf{P}_{3D}} (\mathbf{P}_{3D} - \mathbf{P}_{cam}) \\
&= \frac{\partial L}{\partial \mathbf{p}_{2D}} \frac{\partial \mathbf{p}_{2D}}{\partial \mathbf{p}_{ndc}} \frac{\partial \mathbf{p}_{ndc}}{\partial {}^{h}\mathbf{p}} \mathbf{P}\mathbf{V}_{[:,:3]} (\mathbf{P}_{3D} - \mathbf{P}_{cam}) \\
&= \frac{\partial L}{\partial \mathbf{p}_{2D}} \frac{\partial \mathbf{p}_{2D}}{\partial \mathbf{p}_{ndc}} \frac{\partial \mathbf{p}_{ndc}}{\partial {}^{h}\mathbf{p}} \mathbf{P}\left[\mathbf{p}_{cam}^{T}, 0\right]^{T} \\
&= \frac{\partial L}{\partial \mathbf{p}_{2D}} \frac{\partial \mathbf{p}_{2D}}{\partial \mathbf{p}_{ndc}} \frac{\partial \mathbf{p}_{ndc}}{\partial {}^{h}\mathbf{p}} \begin{bmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} p_x^{G} \\ p_y^{G} \\ p_z^{G} \\ 0 \end{bmatrix} \\
&= \frac{\partial L}{\partial \mathbf{p}_{2D}} \frac{\partial \mathbf{p}_{2D}}{\partial \mathbf{p}_{ndc}} \frac{\partial \mathbf{p}_{ndc}}{\partial {}^{h}\mathbf{p}} \begin{bmatrix} \frac{2n}{r-l} p_x^{G} + \frac{r+l}{r-l} p_z^{G} \\ \frac{2n}{t-b} p_y^{G} + \frac{t+b}{t-b} p_z^{G} \\ -\frac{f+n}{f-n} p_z^{G} \\ -p_z^{G} \end{bmatrix} \\
&= \frac{\partial L}{\partial \mathbf{p}_{2D}} \frac{\partial \mathbf{p}_{2D}}{\partial \mathbf{p}_{ndc}} \frac{\partial \mathbf{p}_{ndc}}{\partial {}^{h}\mathbf{p}} \begin{bmatrix} p_x \\ p_y \\ p_z + \frac{2fn}{f-n} \\ p_w \end{bmatrix} \\
&= \frac{\partial L}{\partial \mathbf{p}_{2D}} \frac{\partial \mathbf{p}_{2D}}{\partial \mathbf{p}_{ndc}} \begin{bmatrix} \frac{1}{p_w} & 0 & 0 & -\frac{p_x}{p_w^2} \\ 0 & \frac{1}{p_w} & 0 & -\frac{p_y}{p_w^2} \\ 0 & 0 & \frac{1}{p_w} & -\frac{p_z}{p_w^2} \end{bmatrix} \begin{bmatrix} p_x \\ p_y \\ p_z + \frac{2fn}{f-n} \\ p_w \end{bmatrix} \\
&= \frac{\partial L}{\partial \mathbf{p}_{2D}} \begin{bmatrix} \frac{W}{2} & 0 & 0 \\ 0 & \frac{H}{2} & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ \frac{2fn}{(f-n)p_w} \end{bmatrix} \\
&= \frac{\partial L}{\partial \mathbf{p}_{2D}} \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\
&= 0
\end{aligned} \tag{15}$$

$$\mathbf{g}_{2d} \cdot \mathbf{D}_{C2G} = \mathbf{g}_{2d} \cdot (\mathbf{P}_{3D} - \mathbf{P}_{cam}) = 0 \tag{16}$$

The derivation above reveals two geometric reasons behind this orthogonality. First, since $\mathbf{p}_{ndc}$ is homogeneous of degree zero in ${}^{h}\mathbf{p}$, the homogeneous coordinate vector ${}^{h}\mathbf{p}$ lies in the null space of $\frac{\partial \mathbf{p}_{ndc}}{\partial {}^{h}\mathbf{p}}$. Combined with $\mathbf{P}[\mathbf{p}_{cam}^{T}, 0]^{T} = {}^{h}\mathbf{p} + [0,\ 0,\ \frac{2fn}{f-n},\ 0]^{T}$, this forces the intermediate vector after perspective division to carry only a z-component, i.e., $\left[0,\ 0,\ \frac{2fn}{(f-n)p_w}\right]^{T}$. Second, left-multiplying by the viewport Jacobian $\frac{\partial \mathbf{p}_{2D}}{\partial \mathbf{p}_{ndc}}$, whose third column is identically zero, annihilates this z-component

and yields $[0,\ 0]^{\mathrm{T}}$. Multiplying by $\frac{\partial L}{\partial \mathbf{p}_{2\mathrm{D}}}$ then trivially gives zero. Consequently, $\mathbf{D}_{\mathrm{C2G}}$ is always orthogonal to $\mathbf{g}_{2\mathrm{d}}$. **In essence, the orthogonality arises from two geometric facts: ${}^{h}\mathrm{p}$ lies in the null space of the perspective division Jacobian, forcing the intermediate vector to a pure z-component; the viewport Jacobian then annihilates this component.** □

A.1.4 Physical Interpretation

Equation 2 has a direct reading in image space. Translating a Gaussian along its viewing ray leaves its projected 2D screen position unchanged, so the projection loss carries no information about where along the ray the primitive sits, and the optimizer directs the 2D branch of the update orthogonally to that ray. For the narrow fields of view and centrally framed subjects typical of capture, viewing rays lie close to the optical axis, so these orthogonal updates are nearly parallel to the image plane and the optimization is starved of depth-directed signals. Depth is therefore determined not by any single view but by the intersection of rays from different angles, which is why the angular spread of the camera trajectory, rather than the number of views, governs depth recovery. The geometric construction is drawn in Figure 9.

A.2 Transmittance Bound on the Depth-Sorted Position Gradient

We prove that along a pixel's ray, the magnitude of the 2D position gradient $\partial L/\partial \mathbf{p}_{2\mathrm{D}}$ of a depth-sorted Gaussian G_m — its u_m component in particular — decays rapidly as the sorted index m increases, making rear Gaussians increasingly unable to surpass the densification threshold τ. This decay is driven by the accumulated transmittance $T_{m-1} = \prod_{j=1}^{m-1}(1-\alpha_j)$ of all front Gaussians: T_{m-1} is monotonically non-increasing in m, so the gradient bound compresses toward zero—the direct cause of **blending-induced densification failure**. We formalize this by decomposing the gradient into factors, identifying the sole m-dependent factor, and bounding its magnitude by T_{m-1}.

For a pixel p, let the depth-sorted Gaussians along its ray be $\mathrm{G}_1, \ldots, \mathrm{G}_N$, and let $[u_m, v_m]^{\top}$ be the 2D position $\mathbf{p}_{2\mathrm{D}}$ of G_m in the viewport-transformed pixel coordinates of Section A.1. The rendered color follows the α-blending equation:

$$\begin{aligned} \hat{\mathbf{c}}^{(p)} &= \sum_{i=1}^{N} \mathbf{c}_i \alpha_i^{(p)} T_{i-1}^{(p)} + \mathbf{c}_{\mathrm{bg}}\, T_N^{(p)}, \\ T_k^{(p)} &= \prod_{j=1}^{k} (1-\alpha_j^{(p)}), \end{aligned} \tag{17}$$

where $\alpha_i^{(p)} = o_i \cdot G_i^{2\mathrm{D}}(p)$ is the effective opacity of G_i at pixel p, with learned opacity $o_i \in (0,1)$ and 2D Gaussian $G_i^{2\mathrm{D}}(p) \in (0,1]$; colors satisfy $\mathbf{c}_i, \mathbf{c}_{\mathrm{bg}} \in [0,1]^3$.

We decompose $\partial L/\partial u_m$ via the chain rule:

$$\frac{\partial L}{\partial u_m} = \sum_{p} \sum_{c=1}^{C} \underbrace{\frac{\partial L}{\partial \hat{\mathbf{c}}_c^{(p)}}}_{\text{Factor A}} \cdot \underbrace{\frac{\partial \hat{\mathbf{c}}_c^{(p)}}{\partial \alpha_m^{(p)}}}_{\text{Factor B}} \cdot \underbrace{o_m}_{\text{Factor C}} \cdot \underbrace{\frac{\partial G_m^{2\mathrm{D},(p)}}{\partial u_m}}_{\text{Factor D}}, \tag{18}$$

and identically for $\partial L/\partial v_m$ with $\partial G_m^{2\mathrm{D},(p)}/\partial v_m$ in Factor D. Equation 18 and its v_m-counterpart are therefore the two components of the 2D position gradient $\partial L/\partial \mathbf{p}_{2\mathrm{D}}$; the magnitude $\|\partial L/\partial \mathbf{p}_{2\mathrm{D}}\|$ accumulated over views is the quantity adaptive densification thresholds on (Section 3.2.2).

A.2.1 m-Dependence of Each Factor

We examine which factors in Equation 18 are influenced by the depth-sorted index m.

Factor A ($\partial L/\partial \hat{\mathbf{c}}_c^{(p)}$): *independent of* m. This term is the derivative of the loss with respect to the rendered pixel color; it does not involve m and is therefore independent of depth ordering.

Factor C (o_m): *independent of* m. The opacity is a learned intrinsic parameter of Gaussian G_m. Its value is attached to the primitive itself and does not change when the view-dependent sorting reassigns which Gaussian occupies slot m.

Factor D ($\partial G_m^{2\mathrm{D},(p)}/\partial u_m$): *independent of* m. This spatial derivative of the 2D Gaussian kernel depends only on the screen-space covariance Σ_m and the pixel offset $\mathbf{\Delta}^{(p)} = [u_m - u_p,\, v_m - v_p]^\top$—quantities determined by the intrinsic 3D position and covariance of G_m, neither of which is affected by the depth-sorted ordering.

Factor B ($\partial \hat{\mathbf{c}}_c^{(p)}/\partial \alpha_m^{(p)}$): *depends on* m. This factor couples G_m to every other Gaussian through the α-blending equation. It is the sole channel through which the depth-sorted index m enters the gradient. We now derive its explicit form.

A.2.2 Derivation of Factor B $\partial \hat{\mathbf{c}}_c^{(p)}/\partial \alpha_m^{(p)}$

Dropping the pixel superscript (p) for brevity, we split Equation 17 at index m to isolate terms involving α_m:

$$\hat{\mathbf{c}} = \underbrace{\sum_{i=1}^{m-1} \mathbf{c}_i \alpha_i T_{i-1}}_{\text{in front of } \mathrm{G}_m} + \underbrace{\mathbf{c}_m \alpha_m T_{m-1}}_{\mathrm{G}_m \text{ itself}} + \underbrace{\sum_{i=m+1}^{N} \mathbf{c}_i \alpha_i T_{i-1} + \mathbf{c}_{\mathrm{bg}} T_N}_{\text{behind } \mathrm{G}_m}. \tag{19}$$

Differentiating Equation 19 w.r.t. α_m (Wang et al., 2025):

$$\begin{aligned} \frac{\partial \hat{\mathbf{c}}}{\partial \alpha_m} &= T_{m-1}\Bigg[\mathbf{c}_m - \underbrace{\left(\sum_{i=m+1}^{N} \mathbf{c}_i \alpha_i \prod_{j=m+1}^{i-1}(1-\alpha_j) + \mathbf{c}_{\mathrm{bg}} \prod_{j=m+1}^{N}(1-\alpha_j)\right)}_{= \hat{\mathbf{c}}_{m+1:N} \,\in\, [0,1]^3}\Bigg] \\ &= T_{m-1} \cdot \big(\mathbf{c}_m - \hat{\mathbf{c}}_{m+1:N}\big), \end{aligned} \tag{20}$$

where $\hat{\mathbf{c}}_{m+1:N}$ denotes the alpha-composited color of the Gaussians behind G_m over the background.

Since both $\mathbf{c}_m$ and $\hat{\mathbf{c}}_{m+1:N}$ lie in $[0,1]^3$, their per-channel difference is bounded by 1, yielding the bound

$$\left|\frac{\partial \hat{\mathbf{c}}_c}{\partial \alpha_m}\right| \le T_{m-1}, \quad \forall\, c \in \{1,2,3\}, \tag{21}$$

where $T_{m-1} = \prod_{j=1}^{m-1}(1-\alpha_j)$ is the accumulated transmittance of the $m-1$ Gaussians in front of G_m.

Monotonicity of the bound. Each $\alpha_j \in [0,1]$ implies $1-\alpha_j \in [0,1]$, hence $T_k = T_{k-1}\cdot(1-\alpha_k) \le T_{k-1}$ for all $k \ge 1$. The transmittance sequence is therefore monotonically non-increasing:

$$T_0 = 1 \;\ge\; T_1 \;\ge\; T_2 \;\ge\; \cdots \;\ge\; T_N \;\ge\; 0. \tag{22}$$

Consequently, the upper bounds in Equation 21 can only decrease or stay constant as the depth-sorted index m grows.

A.2.3 Gradient Attenuation

The exact gradient $T_{m-1} \cdot (\mathbf{c}_m - \hat{\mathbf{c}}_{m+1:N})$ depends on m through two factors: the transmittance T_{m-1}, which decays monotonically, and the color difference $\mathbf{c}_m - \hat{\mathbf{c}}_{m+1:N}$. The latter varies non-monotonically with m, because advancing m alters both which Gaussian occupies slot m and the rear subsequence $\hat{\mathbf{c}}_{m+1:N}$ being composited.

While the color-difference term may cause the gradient magnitude to fluctuate, the magnitude can never exceed the monotonic envelope T_{m-1}, and this envelope decays monotonically toward zero as front Gaussians accumulate. This attenuation is inescapable: T_{m-1} depends only on the Gaussians in front of position m, so the gradient bound lies entirely outside the control of rear Gaussians.

As m increases, rear Gaussians therefore receive a progressively weaker 2D positional gradient signal, making them increasingly unable to surpass the densification threshold τ—the direct mechanism of blending-induced densification failure. □

B MECHANISM EVIDENCE

This appendix collects the empirical evidence behind the diagnosis of Section 3, which the main text summarises but does not display. The four subsections follow that section's structure: the magnitude disparity that makes the 2D branch dominant, the component ablation that confirms it, the direction statistics that reveal the missing depth signal, and the single-pixel profile of the blending-induced attenuation.

B.1 GRADIENT-BRANCH MAGNITUDES AND DIRECTIONS

Table 2 collects the branch-wise summary referred to in Section 3.2.1: for each term of Equation 1, the magnitude relative to $\|\mathbf{g}_{\text{all}}\|$ and the direction relative to the viewing ray. The three rows restate the magnitude disparity measured in Figure 2 (Section 3.1) and the direction statistics of Figure 3 (Section 3.2.1). Read together, they explain why the 3D position gradient $\mathbf{g}_{\text{all}}$ is only approximately orthogonal to the viewing ray: $\mathbf{g}_{\text{2d}}$ is strictly orthogonal and dominates the magnitude, $\mathbf{g}_{\text{sh}}$ shares the orthogonality but is orders of magnitude smaller, and $\mathbf{g}_{\text{cov2d}}$ is both small and directionally diffuse. Neither minor branch is therefore effective for depth optimization.

Table 2: **Magnitude and direction of the three gradient branches.** For each term of Equation 1: its magnitude relative to $\|\mathbf{g}_{\text{all}}\|$, and its direction relative to the viewing ray.

	Branch	Magnitude	Direction relative to the viewing ray
$\mathbf{g}_{\text{all}}$	$\mathbf{g}_{\text{2d}}$	$\approx \|\mathbf{g}_{\text{all}}\|$	Orthogonal in theory
	$\mathbf{g}_{\text{sh}}$	$\ll \|\mathbf{g}_{\text{all}}\|$	Orthogonal in practice
	$\mathbf{g}_{\text{cov2d}}$	$\ll \|\mathbf{g}_{\text{all}}\|$	Diffuse, no preferred direction

B.2 COMPONENT ABLATION ON THE 2D POSITION GRADIENT

Table 3 reports the component ablation referred to in Section 3.1: each row enables a subset of the three gradient items of Equation 1, and the three datasets agree on the same ordering. Keeping the 2D position branch reproduces standard 3DGS quality, dropping it costs roughly 3 dB PSNR and 0.1 in SSIM, and dropping either minor branch alone changes little.

Table 3: **Component ablation on the 2D position gradient.** Each row enables the subset of the three gradient items of Equation 1 marked by a check mark; all other settings follow standard 3DGS.

Gradient Items			Mip-NeRF360			Tanks &Temples			Deep Blending		
Cov.	2D	SH	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
✓	✓	✓	**27.52**	**0.816**	**0.215**	**23.73**	**0.853**	**0.169**	**29.80**	**0.907**	**0.238**
	✓	✓	**27.52**	**0.816**	**0.215**	**23.77**	**0.853**	**0.169**	**29.85**	**0.907**	**0.238**
✓		✓	24.27	0.661	0.386	21.59	0.729	0.333	24.92	0.825	0.376
✓	✓		**27.53**	**0.816**	**0.215**	**23.73**	**0.853**	**0.169**	**29.78**	**0.907**	**0.239**
✓			24.18	0.658	0.388	21.54	0.724	0.336	24.88	0.826	0.376
	✓		**27.54**	**0.816**	**0.215**	**23.84**	**0.853**	**0.170**	**29.80**	**0.907**	**0.238**
		✓	25.23	0.706	0.340	22.45	0.771	0.285	28.12	0.879	0.301

B.3 MULTI-VIEW DEPTH CONSTRAINTS: GEOMETRY AND STATISTICS

Figure 9 illustrates the geometric argument of Section 3.2.1: with viewpoints clustered in a narrow angular range, each view constrains a primitive only within the plane orthogonal to its own ray, so the depth direction stays free; the same number of viewpoints spread across diverse angles intersect at the surface and pin the primitive in place.

Figure 10 then measures the consequence over a full training run. Unlike the single-view histograms of Figure 3, which take each primitive's own viewing ray or the optical axis as reference, this figure re-expresses the same deficiency in the global coordinate system: the reference direction is the primitive's position vector $\boldsymbol{x}$, and the quantity recorded is its cosine similarity with the position gradient $\nabla_{\boldsymbol{x}}\mathcal{L}$. The left panel resolves this against distance from the origin, coloured by training iteration; the right panel aggregates distance-ordered equal-frequency bins into per-bin medians with 95% confidence intervals.

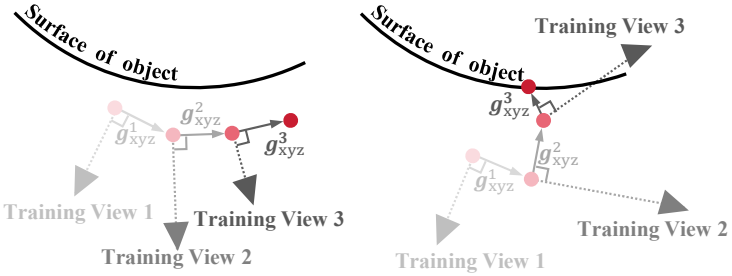

Figure 9: **Why angular diversity, not view count, resolves depth.** Under the orthogonality of Equation 2, one view constrains a primitive only within the plane orthogonal to its ray, leaving the ray's own direction free. **Left:** viewpoints clustered within a narrow angular range constrain nearly the same plane, so the depth direction stays unconstrained however many such views are added. **Right:** the *same number* of viewpoints spread across diverse angles intersect at the object surface and pin the primitive to its location. Depth is therefore paid for in parallax, not in views.

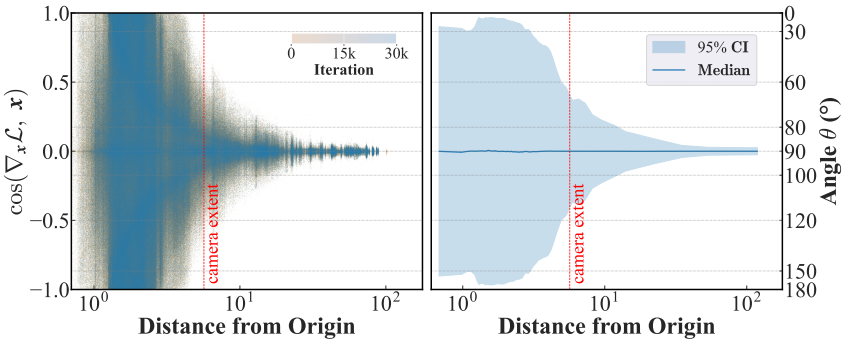

Figure 10: **Multi-view statistics: depth signal degrades with distance.** Cosine similarity between each primitive's position vector $\boldsymbol{x}$ and its position gradient $\nabla_{\boldsymbol{x}}\mathcal{L}$, recorded in the global coordinate system over the whole training run. **Left:** per-primitive values against distance from the origin, coloured by training iteration. **Right:** per-bin medians with 95% confidence intervals over distance-ordered equal-frequency bins. The right axis gives the equivalent angle $\theta = \arccos(\cos)$, and the red vertical dashed line marks the camera-trajectory extent.

B.4 Single-Pixel Profile of the Blending-Induced Attenuation

Figure 11 gives the single-pixel profile referred to in Section 3.2.2: for the primitives that composite one central pixel at a single training iteration, it plots the 2D position gradient magnitude, the camera-space depth and the transmittance T against the rendering order i. The three panels show the gradient attenuation, the disjoint depth intervals of the contributing primitives, and the transmittance decay that produces the attenuation. The corresponding measurement over all pixels of the whole training process is Figure 4 in Section 3.2.2, which isolates the transmittance as its cause.

C Pseudocode and Hyper-parameters

C.1 Notation

Let $\mathcal{G} = \{g_i\}_{i=1}^{N}$ denote the set of Gaussian primitives. Each primitive g_i is parameterized by its position $\boldsymbol{\mu}_i \in \mathbb{R}^3$, anisotropic scale $\mathbf{s}_i \in \mathbb{R}^3$, rotation quaternion $\mathbf{q}_i \in \mathbb{R}^4$ (with corresponding rotation matrix $\mathbf{R}(\mathbf{q}_i)$), opacity $o_i \in \mathbb{R}$, and spherical-harmonic color coefficients $\mathbf{c}_i$. τ_{split} is the standard 3DGS densification gradient threshold. Table 4 summarizes the exploration hyper-parameters.

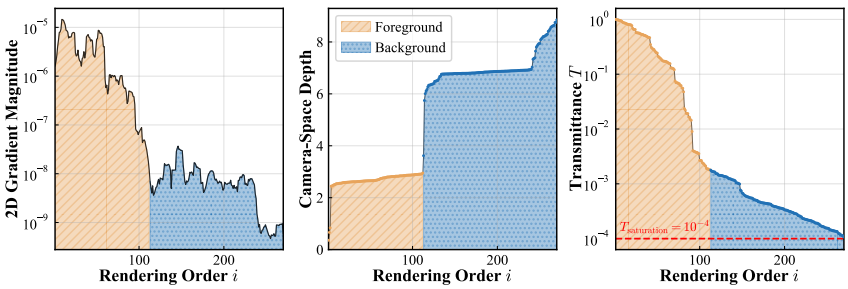

Figure 11: **Single-pixel profile: attenuated gradients behind disjoint structures.** The 2D position gradient magnitude, camera-space depth, and transmittance T of the primitives compositing one pixel, plotted against the rendering order i at a single training iteration. **Left:** Foreground primitives ($i \leq 112$) keep strong gradient signals, while background primitives ($i > 112$) are attenuated by the transmittance accumulated in front of them. **Middle:** the two groups occupy distinct depth intervals, so primitives jointly responsible for one pixel's colour can belong to spatially separate structures. **Right:** the transmittance falls steeply across the foreground yet compositing continues, so the later primitives are optimized with the residual signal only.

Table 4: Exploration hyper-parameters.

Symbol	Meaning	Default
N_{seed}	seeds injected per seeding step	20
I_{seed}	seeding interval (iterations)	100
T_{seed}	stop seeding after this iteration	15000
N_{split}	large Gaussians split per step	20
I_{split}	splitting interval (iterations)	1
T_{split}	stop splitting after this iteration	15000
N	children produced per split parent	2
$t_{\text{densify}}, T_{\text{densify}}$	densification start / end iteration	500 / 15000

Each splitting step is small, but because it fires every iteration the cumulative intervention is not: at the defaults above roughly 2.9×10^5 splits are issued over training.

C.2 Algorithm Pseudocode

Algorithm 1 3DGS Training with Explicit Exploration (ExploreGS)

Require: Training views; SfM-initialized primitive set $\mathcal{G}$

for $t = 1, \ldots, T_{\max}$ **do**
  sample a training camera, rasterize, and compute loss $\mathcal{L}$
  backpropagate $\mathcal{L}$ ▷ standard gradient-based optimization
  if $t < T_{\text{densify}}$ **then**
    gradient densification: clone / split by τ_{split}; prune low-opacity and oversized primitives
  end if
  optimizer step
  ExploreStep($\mathcal{G}$, t) ▷ added operators; Alg. 2
end for
return $\mathcal{G}$

C.3 Oversized-Gaussian Pruning

We retain the oversized-Gaussian criterion of HoGS (Liu et al., 2025). HoGS discard a primitive whose largest world-space scale exceeds a threshold, and we change only the threshold. A fixed fraction of the scene extent ($0.1\times$ in standard 3DGS) is hostile to far-field exploration: a seed carries a scale proportional to its distance from the origin (Algorithm 3), so the more distant the

Algorithm 2 EXPLORESTEP($\mathcal{G}, t$): scheduling of the exploration operators

if $t_{\text{densify}} < t < T_{\text{seed}} \ \wedge \ t \bmod I_{\text{seed}} = 0$ **then**
  RANDOMSEEDING($\mathcal{G}$) ▷ targets the Far-Side Blur Trap
end if
if $t_{\text{densify}} < t < T_{\text{split}} \ \wedge \ t \bmod I_{\text{split}} = 0$ **then**
  RANDOMSPLITTING($\mathcal{G}$) ▷ targets the Near-Side Blur Trap
end if

Algorithm 3 RANDOMSEEDING($\mathcal{G}$): coverage-driven far-side exploration

$\mathbf{b}_{\min}, \mathbf{b}_{\max} \leftarrow \min_i \boldsymbol{\mu}_i, \ \max_i \boldsymbol{\mu}_i$ ▷ per-axis bounding box
$\bar{s} \leftarrow \frac{1}{3N} \sum_{i,k} s_{i,k} \,/\, \|\boldsymbol{\mu}_i\|_2$ ▷ mean scale-to-distance ratio
for $j = 1, \ldots, N_{\text{seed}}$ **do**
  $\boldsymbol{\mu}_j \sim \mathcal{U}(\mathbf{b}_{\min}, \ \mathbf{b}_{\max})$ ▷ uniform inside bounding box
  $\mathbf{c}_j \leftarrow$ random SH color coefficients
  $\mathbf{s}_j \leftarrow \bar{s} \, \|\boldsymbol{\mu}_j\|_2 \cdot \mathbf{1}$ ▷ depth-proportional isotropic scale
  $o_j \leftarrow \gamma + \beta \, \|\boldsymbol{\mu}_j\|_2 \,/\, \max_j \|\boldsymbol{\mu}_j\|_2$ ▷ depth-modulated opacity, e.g. γ=0.2, β=0.5
  $\mathbf{q}_j \leftarrow$ random unit quaternion
  $\mathcal{G} \leftarrow \mathcal{G} \cup \{g_j\}$
end for

Algorithm 4 RANDOMSPLITTING($\mathcal{G}$): scale-biased near-side exploration

for each $g_i \in \mathcal{G}$ **do**
  $w_i \leftarrow \frac{1}{3} \sum_k s_{i,k}$ ▷ mean scale
end for
$p_i \leftarrow w_i \,/\, \sum_j w_j$ ▷ probability proportional to mean scale
$\mathcal{I} \leftarrow \text{Multinomial}(\mathbf{p}, \ N_{\text{split}})$ ▷ without replacement
for $i \in \mathcal{I}$ **do**
  for $j = 1, \ldots, N$ **do**
    $\boldsymbol{\delta} \sim \mathcal{N}(\mathbf{0}, \ \text{diag}(\mathbf{s}_i^2))$ ▷ anisotropic perturbation
    $\boldsymbol{\mu}_{ij} \leftarrow \boldsymbol{\mu}_i + \mathbf{R}(\mathbf{q}_i) \, \boldsymbol{\delta}$
    $\mathbf{s}_{ij} \leftarrow \mathbf{s}_i \,/\, (0.8 \, N)$
    $(\mathbf{c}_{ij}, \ o_{ij}, \ \mathbf{q}_{ij}) \leftarrow (\mathbf{c}_i, \ o_i, \ \mathbf{q}_i)$
    $\mathcal{G} \leftarrow \mathcal{G} \cup \{g_{ij}\}$
  end for
  $\mathcal{G} \leftarrow \mathcal{G} \setminus \{g_i\}$ ▷ remove parent; net change $+N_{\text{split}}(N-1)$
end for

region just explored, the more reliably that rule deletes what was placed there. We use $10\times$ the scene extent in place of $0.1\times$. More generally, what the criterion requires is a threshold that grows with the primitive's distance from the world origin; a linear growth suffices, and it makes the survival test distance-independent, which is what the distance-proportional seeding scale assumes in the first place. Relaxing the threshold is necessary for *Random Seeding* but not sufficient by itself: Appendix D.3.2 separates the two and shows that the far-side blur survives the change.

D EXPERIMENTAL RESULTS

This appendix reports the fuller evidence behind Section 5: repeated runs over multiple seeds, the complete DL3DV scene set rather than the 16-scene subset used in the main text, per-scene paired comparisons, wall-clock timing, per-dataset primitive allocation, and the additional ablations and qualitative comparisons that the main text summarises but does not display. The split ablation of Appendix D.3.4 also serves as a causal test of the Near-Side analysis of Section 3.2.2.

D.1 Main Results

Table 5 repeats the five-dataset comparison over seeds 1–3, reporting the mean and standard deviation of the per-seed dataset averages. The run-to-run spread is small relative to the improvements contributed by the two operators. Across the five datasets and three metrics, the combined configuration is the best or tied-best among our variants in every cell but one—LPIPS on Deep Blending, where it reaches 0.249 against 0.237 for *Seed Exploration*.

Table 5: **Multi-seed results on five datasets.** *Seed Exp.*, *Split Exp.* and *Seed & Split Exp.* abbreviate the three operator configurations. For each seed we average a metric over the scenes of a dataset, then report the mean and standard deviation of those three per-seed averages. *The same primitive budget as *Seed & Split Exploration*, so the two rows are not independent evidence about efficiency.

Dataset	Metric	3DGS	Seed Exp.	Split Exp.	Seed & Split Exp.	GS-MCMC*
Mip-NeRF 360	PSNR	27.52 ±0.02	27.70 ±0.01	27.95 ±0.02	27.96 ±0.02	28.04 ±0.03
	SSIM	0.816 ±0.000	0.818 ±0.000	0.829 ±0.000	0.829 ±0.000	0.831 ±0.000
	LPIPS	0.215 ±0.000	0.214 ±0.000	0.192 ±0.001	0.192 ±0.002	0.191 ±0.003
Tanks & Temples	PSNR	23.74 ±0.10	24.30 ±0.08	24.30 ±0.07	24.37 ±0.07	24.28 ±0.05
	SSIM	0.853 ±0.001	0.860 ±0.001	0.868 ±0.001	0.871 ±0.001	0.865 ±0.000
	LPIPS	0.169 ±0.001	0.163 ±0.001	0.141 ±0.001	0.138 ±0.001	0.149 ±0.000
Deep Blending	PSNR	29.76 ±0.08	29.78 ±0.10	30.01 ±0.03	30.03 ±0.07	29.42 ±0.03
	SSIM	0.907 ±0.000	0.906 ±0.002	0.907 ±0.001	0.907 ±0.001	0.895 ±0.001
	LPIPS	0.238 ±0.000	0.237 ±0.000	0.248 ±0.001	0.249 ±0.001	0.266 ±0.001
OMMO	PSNR	30.50 ±0.03	30.85 ±0.03	31.29 ±0.01	31.27 ±0.02	30.42 ±0.04
	SSIM	0.920 ±0.000	0.924 ±0.000	0.928 ±0.000	0.930 ±0.000	0.921 ±0.000
	LPIPS	0.142 ±0.000	0.131 ±0.001	0.121 ±0.000	0.119 ±0.000	0.138 ±0.001
DL3DV	PSNR	27.16 ±0.02	27.75 ±0.01	28.45 ±0.01	28.46 ±0.01	28.13 ±0.01
	SSIM	0.861 ±0.000	0.868 ±0.000	0.888 ±0.000	0.889 ±0.000	0.886 ±0.000
	LPIPS	0.167 ±0.000	0.155 ±0.000	0.126 ±0.000	0.125 ±0.000	0.132 ±0.000

The main text evaluates a 16-scene subset of DL3DV, which is small enough that dataset means could hide scene-level regressions. Table 6 therefore repeats the comparison over all 135 usable DL3DV scenes and additionally records wall-clock training time, which the main text does not.

Table 6: **Full-scale results on DL3DV.** Mean over all 135 usable scenes, seed 0, 30k iterations; the best value in each column is **bolded**, and the last two columns report model size and wall-clock training time. *The same primitive budget as ours. ‡HoGS at its native 50k iterations; every other method uses 30k.

Method	PSNR (dB) ↑	SSIM ↑	LPIPS ↓	#G (M) ↓	Time (min) ↓
3DGS	29.291	0.9066	0.1324	1.043	9.04
+ Random Seeding	29.725	0.9106	0.1272	**1.021**	10.05
+ Random Splitting	30.333	0.9205	0.1090	1.649	13.35
+ Both **(Ours)**	**30.335**	**0.9206**	**0.1089**	1.648	13.42
AbsGS	29.327	0.9096	0.1220	1.235	**8.54**
3DGS-MCMC*	29.921	0.9179	0.1190	1.648	17.52
HoGS‡	29.981	0.9153	0.1183	1.496	14.86

Table 7 converts the same runs into a per-scene, paired view: for each ordered pair of methods it gives the fraction of the 135 scenes on which the first beats the second on *all three* metrics simultaneously. This is the form of evidence a dataset mean cannot supply, since a method can lead on average while losing on individual scenes, and it is the basis for the generalisation claim in the main text.

Table 8 lists the individual per-scene results behind the aggregates above, so that any scene can be checked directly.

Table 7: **Head-to-head wins on all three metrics.** Each cell is the percentage of the 135 scenes on which the row method beats the column method on PSNR, SSIM and LPIPS *simultaneously*, so a value near 50 would indicate two comparable methods and a value near 0 or 100 a systematic ordering. The better side of each pair is **bolded**.

	Wins over (%)						
Method	3DGS	Seed	Split	Ours	AbsGS	MCMC	HoGS
3DGS	—	0.7	0.0	0.0	5.2	3.0	3.0
+ Random Seeding	**93.3**	—	0.0	0.0	**23.0**	11.9	3.7
+ Random Splitting	**100.0**	**97.0**	—	30.4	**97.0**	**76.3**	**86.7**
+ Both **(Ours)**	**100.0**	**98.5**	**34.1**	—	**96.3**	**77.8**	**89.6**
AbsGS	**55.6**	17.8	*0.0*	*0.0*	—	5.9	7.4
3DGS-MCMC	**79.3**	**62.2**	*1.5*	*1.5*	**59.3**	—	**33.3**
HoGS	**92.6**	**77.0**	*2.2*	*1.5*	**65.2**	30.4	—

Table 8: **Per-scene metrics on DL3DV.** PSNR (dB, 3 d.p.) and SSIM and LPIPS (4 d.p.) for each of the 135 evaluated scenes; `Idx` is the scene index, i.e. `DL3DV_benchmark_Idx` (the excluded indices are listed once in the text, not repeated here). The last row gives the means.

	3DGS			Ours			AbsGS			MCMC			HoGS		
Idx	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
0	26.307	0.8474	0.2483	26.890	0.8629	0.2190	26.309	0.8502	0.2327	25.811	0.8500	0.2689	26.888	0.8617	0.2297
1	22.143	0.6904	0.2860	22.946	0.7361	0.2289	22.279	0.7150	0.2409	22.559	0.7311	0.2368	22.350	0.7040	0.2689
3	32.068	0.9520	0.0691	33.130	0.9599	0.0544	32.591	0.9591	0.0522	32.602	0.9596	0.0568	32.528	0.9550	0.0614
4	25.291	0.8050	0.1706	26.267	0.8391	0.1308	25.458	0.8148	0.1569	26.031	0.8358	0.1388	26.183	0.8324	0.1379
5	30.269	0.9479	0.0625	30.770	0.9520	0.0545	30.316	0.9488	0.0572	30.346	0.9495	0.0624	30.210	0.9471	0.0614
6	23.902	0.7486	0.2255	24.483	0.7786	0.1952	24.118	0.7659	0.2031	24.364	0.7772	0.1996	24.529	0.7785	0.1971
7	23.930	0.7464	0.2490	25.022	0.8031	0.1798	24.481	0.7815	0.2022	24.877	0.7987	0.1886	24.405	0.7623	0.2293
8	23.822	0.8146	0.1606	24.451	0.8427	0.1292	24.051	0.8263	0.1444	24.351	0.8419	0.1312	24.260	0.8268	0.1498
9	27.925	0.8913	0.1595	28.871	0.9078	0.1234	27.777	0.8919	0.1489	28.682	0.9060	0.1409	28.503	0.9068	0.1280
10	29.671	0.9255	0.0610	30.169	0.9341	0.0521	29.658	0.9270	0.0573	29.968	0.9336	0.0526	29.938	0.9273	0.0576
11	24.559	0.7728	0.2435	25.760	0.8197	0.2010	24.585	0.7751	0.2338	24.968	0.8057	0.2265	25.654	0.8157	0.2079
12	26.660	0.8939	0.0918	27.298	0.9071	0.0787	26.655	0.8960	0.0864	27.127	0.9030	0.0827	26.884	0.8948	0.0895
13	29.860	0.9353	0.0527	30.562	0.9454	0.0415	29.925	0.9373	0.0490	30.138	0.9407	0.0455	30.031	0.9386	0.0471
14	26.807	0.9018	0.1161	29.474	0.9363	0.0749	26.712	0.9013	0.1127	28.821	0.9331	0.0798	28.694	0.9295	0.0828
15	30.474	0.9425	0.0555	31.218	0.9516	0.0446	30.427	0.9444	0.0513	30.676	0.9476	0.0520	30.917	0.9468	0.0498
16	28.177	0.9206	0.0703	29.175	0.9349	0.0554	28.232	0.9238	0.0657	28.826	0.9320	0.0600	29.165	0.9357	0.0562
18	23.821	0.7781	0.2146	24.672	0.7992	0.1883	23.893	0.7838	0.2030	24.329	0.7939	0.1993	24.421	0.7924	0.1971
19	29.215	0.9190	0.1061	30.244	0.9327	0.0847	28.778	0.9164	0.1014	29.590	0.9284	0.0909	29.711	0.9223	0.0964
21	24.999	0.7948	0.2253	26.215	0.8272	0.1785	25.230	0.8052	0.2027	25.992	0.8244	0.1962	26.204	0.8226	0.1863
22	33.429	0.9594	0.1102	34.202	0.9647	0.1003	33.017	0.9583	0.1087	33.577	0.9639	0.1081	33.696	0.9620	0.1049
23	27.478	0.9355	0.1035	29.193	0.9464	0.0869	27.205	0.9351	0.0969	28.729	0.9429	0.1045	28.035	0.9402	0.0964
24	31.655	0.9442	0.0916	32.858	0.9557	0.0703	32.000	0.9484	0.0805	32.019	0.9531	0.0771	32.782	0.9536	0.0733
25	29.741	0.9349	0.0955	29.963	0.9375	0.0898	29.372	0.9312	0.0993	29.821	0.9365	0.0939	29.263	0.9322	0.0972
26	28.704	0.8944	0.1902	29.486	0.9098	0.1574	28.563	0.9003	0.1675	29.279	0.9098	0.1598	29.022	0.8991	0.1763
27	32.747	0.9527	0.0629	33.658	0.9574	0.0535	32.951	0.9573	0.0531	33.307	0.9594	0.0536	32.883	0.9540	0.0588
28	30.641	0.9227	0.1089	32.290	0.9441	0.0676	30.547	0.9242	0.0972	32.032	0.9440	0.0734	32.176	0.9432	0.0689
29	25.324	0.8569	0.1498	25.852	0.8709	0.1289	25.324	0.8583	0.1439	25.878	0.8725	0.1313	25.839	0.8689	0.1312
30	33.998	0.9710	0.0565	35.643	0.9786	0.0391	34.466	0.9734	0.0510	34.963	0.9765	0.0473	35.365	0.9775	0.0412
31	33.051	0.9496	0.0757	33.932	0.9574	0.0581	33.058	0.9494	0.0733	33.369	0.9515	0.0736	33.588	0.9538	0.0651
32	25.486	0.8517	0.1660	26.268	0.8636	0.1508	25.159	0.8475	0.1706	25.639	0.8522	0.1839	25.894	0.8546	0.1656
33	31.359	0.9503	0.0564	31.810	0.9549	0.0486	31.157	0.9500	0.0558	31.484	0.9532	0.0531	31.490	0.9525	0.0520
34	29.265	0.9194	0.1410	31.368	0.9451	0.0850	29.353	0.9242	0.1175	31.140	0.9434	0.0915	31.332	0.9460	0.0800
35	25.348	0.8082	0.1827	25.814	0.8245	0.1585	25.401	0.8175	0.1643	25.750	0.8266	0.1630	25.804	0.8237	0.1624
36	28.890	0.9265	0.0880	30.354	0.9461	0.0597	29.472	0.9366	0.0719	30.317	0.9466	0.0605	30.365	0.9470	0.0596
37	26.908	0.8558	0.2356	28.045	0.8846	0.1740	27.352	0.8714	0.2058	27.567	0.8808	0.1968	27.973	0.8751	0.1978
38	31.434	0.9371	0.1928	32.447	0.9445	0.1769	31.514	0.9383	0.1869	31.965	0.9429	0.1866	31.838	0.9389	0.1890

(continued on the next page)

Table 8 (continued)

	3DGS			Ours			AbsGS			MCMC			HoGS		
Idx	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
39	31.488	0.9491	0.1586	31.912	0.9513	0.1526	31.355	0.9492	0.1526	31.485	0.9498	0.1614	31.669	0.9506	0.1537
40	34.384	0.9726	0.0681	35.101	0.9753	0.0580	34.212	0.9727	0.0641	34.052	0.9702	0.0798	35.057	0.9739	0.0643
41	33.459	0.9574	0.1724	33.695	0.9588	0.1619	33.165	0.9567	0.1675	32.959	0.9574	0.1732	33.857	0.9586	0.1660
42	27.069	0.9014	0.1254	27.489	0.9117	0.1112	27.147	0.9060	0.1172	27.420	0.9137	0.1096	27.186	0.9061	0.1191
43	29.890	0.9037	0.2073	30.667	0.9177	0.1755	29.966	0.9096	0.1861	30.325	0.9159	0.1828	30.353	0.9122	0.1871
44	31.688	0.9392	0.1907	32.008	0.9447	0.1756	31.362	0.9380	0.1876	31.397	0.9406	0.1916	31.927	0.9429	0.1802
45	35.148	0.9729	0.0487	36.113	0.9781	0.0350	34.905	0.9732	0.0448	35.455	0.9725	0.0514	35.290	0.9741	0.0427
46	30.770	0.9437	0.1567	31.170	0.9466	0.1481	30.518	0.9412	0.1555	30.733	0.9439	0.1570	30.669	0.9426	0.1555
47	29.103	0.9006	0.1810	29.804	0.9103	0.1616	29.185	0.9043	0.1692	28.401	0.8806	0.2329	29.698	0.9066	0.1749
48	27.456	0.9157	0.1587	27.895	0.9218	0.1442	27.425	0.9156	0.1552	27.814	0.9220	0.1511	27.642	0.9186	0.1542
49	33.279	0.9651	0.0799	34.413	0.9692	0.0733	33.599	0.9662	0.0770	33.930	0.9673	0.0798	34.317	0.9689	0.0737
50	30.324	0.9363	0.1012	31.374	0.9499	0.0721	30.732	0.9450	0.0744	31.125	0.9503	0.0723	30.790	0.9429	0.0856
51	27.898	0.9242	0.0952	28.797	0.9357	0.0737	27.995	0.9263	0.0889	28.708	0.9348	0.0777	28.629	0.9332	0.0779
52	29.053	0.9353	0.0796	29.870	0.9436	0.0662	28.724	0.9279	0.0789	29.773	0.9438	0.0692	28.996	0.9322	0.0802
53	28.127	0.9113	0.1998	29.465	0.9236	0.1743	28.288	0.9138	0.1862	29.278	0.9213	0.1873	29.409	0.9219	0.1868
54	32.458	0.9270	0.2063	33.297	0.9399	0.1789	32.764	0.9378	0.1818	32.941	0.9406	0.1811	32.841	0.9338	0.1911
55	30.835	0.9020	0.1994	31.600	0.9148	0.1669	30.926	0.9067	0.1785	31.325	0.9142	0.1691	31.286	0.9046	0.1944
56	30.525	0.9454	0.1228	30.987	0.9500	0.1098	30.405	0.9451	0.1193	30.736	0.9467	0.1245	30.776	0.9472	0.1196
57	29.310	0.9346	0.1292	29.799	0.9400	0.1142	29.221	0.9341	0.1263	29.339	0.9356	0.1300	29.474	0.9362	0.1242
58	28.217	0.9044	0.1475	28.774	0.9095	0.1395	27.590	0.8956	0.1591	28.670	0.8977	0.1675	28.792	0.9075	0.1450
59	26.282	0.8716	0.1573	27.434	0.8924	0.1304	26.525	0.8742	0.1503	27.309	0.8929	0.1331	27.348	0.8908	0.1325
60	28.606	0.9199	0.1019	29.423	0.9297	0.0864	28.377	0.9194	0.0999	29.220	0.9285	0.0938	29.331	0.9290	0.0880
62	33.034	0.9626	0.0876	33.563	0.9658	0.0767	32.965	0.9627	0.0845	32.959	0.9640	0.0843	33.380	0.9636	0.0849
63	29.157	0.8599	0.1447	30.221	0.8787	0.1190	28.765	0.8564	0.1384	29.659	0.8771	0.1220	29.901	0.8749	0.1246
64	33.224	0.9637	0.0496	33.732	0.9657	0.0449	33.240	0.9672	0.0390	33.290	0.9636	0.0520	32.390	0.9558	0.0592
65	24.108	0.8204	0.2395	24.467	0.8330	0.2154	23.146	0.8054	0.2407	24.084	0.8227	0.2376	24.293	0.8227	0.2355
66	27.831	0.8941	0.1291	28.352	0.9041	0.1111	27.873	0.8969	0.1211	28.046	0.9028	0.1186	28.312	0.9025	0.1145
67	27.638	0.8840	0.1211	28.159	0.8960	0.1036	27.615	0.8862	0.1150	27.948	0.8941	0.1103	27.896	0.8892	0.1115
68	26.832	0.8894	0.1314	27.281	0.8989	0.1147	26.875	0.8923	0.1235	27.004	0.8976	0.1241	27.166	0.8973	0.1187
69	25.269	0.8299	0.1868	26.461	0.8757	0.1290	25.197	0.8435	0.1613	26.374	0.8759	0.1294	26.308	0.8642	0.1486
70	25.137	0.8317	0.1991	25.616	0.8461	0.1799	24.208	0.8121	0.2087	25.218	0.8443	0.1717	25.274	0.8375	0.1924
71	31.336	0.9473	0.0876	33.757	0.9610	0.0647	31.203	0.9496	0.0798	32.896	0.9592	0.0695	33.086	0.9576	0.0722
72	30.812	0.9496	0.0975	31.763	0.9566	0.0845	31.118	0.9521	0.0914	31.365	0.9541	0.0942	31.399	0.9537	0.0896
73	28.196	0.9226	0.1026	31.031	0.9504	0.0714	28.795	0.9308	0.0912	30.331	0.9464	0.0806	31.061	0.9488	0.0743
74	25.128	0.8222	0.1580	26.186	0.8526	0.1201	25.200	0.8261	0.1514	25.945	0.8501	0.1300	26.234	0.8550	0.1203
75	35.088	0.9617	0.0801	36.353	0.9683	0.0652	35.170	0.9641	0.0720	35.292	0.9638	0.0782	35.702	0.9665	0.0685
76	33.504	0.9675	0.0481	34.468	0.9729	0.0391	33.542	0.9686	0.0441	33.897	0.9709	0.0471	34.309	0.9718	0.0396
77	31.699	0.9528	0.0800	32.703	0.9605	0.0687	31.967	0.9551	0.0751	32.330	0.9580	0.0781	32.396	0.9571	0.0722
78	26.843	0.8654	0.1445	27.772	0.8765	0.1290	27.023	0.8697	0.1294	27.681	0.8759	0.1311	27.384	0.8740	0.1318
79	29.163	0.8929	0.1442	29.421	0.8998	0.1313	29.078	0.8942	0.1341	29.075	0.8982	0.1354	28.906	0.8909	0.1436
80	31.281	0.9518	0.1120	32.162	0.9595	0.0825	31.580	0.9551	0.0933	31.266	0.9520	0.1228	32.005	0.9570	0.0978
81	30.395	0.9193	0.1276	30.980	0.9272	0.1072	30.169	0.9180	0.1225	30.664	0.9258	0.1069	30.444	0.9201	0.1230
82	28.923	0.8950	0.1111	29.904	0.9190	0.0789	29.176	0.9068	0.0935	29.469	0.9170	0.0819	29.324	0.8992	0.1012
83	24.277	0.7759	0.1998	24.867	0.8017	0.1695	24.404	0.7883	0.1790	24.549	0.7967	0.1753	24.623	0.7854	0.1888
84	33.589	0.9599	0.0890	34.526	0.9659	0.0749	33.806	0.9618	0.0824	33.710	0.9616	0.0944	34.329	0.9640	0.0805
85	26.383	0.8338	0.1883	26.923	0.8523	0.1600	26.023	0.8342	0.1819	26.720	0.8479	0.1686	26.989	0.8462	0.1698
86	28.533	0.9204	0.0918	30.189	0.9376	0.0702	28.336	0.9208	0.0878	29.515	0.9349	0.0752	29.477	0.9297	0.0789
87	22.886	0.7718	0.1864	23.308	0.7871	0.1728	22.090	0.7505	0.1926	23.306	0.7895	0.1730	22.938	0.7781	0.1766
88	30.474	0.9418	0.0824	31.717	0.9569	0.0543	30.792	0.9480	0.0680	31.423	0.9561	0.0547	30.721	0.9423	0.0774
89	28.015	0.8993	0.1086	28.822	0.9090	0.0987	27.571	0.8921	0.1067	28.640	0.9104	0.0938	28.097	0.9007	0.1030
90	26.059	0.8830	0.1498	26.438	0.8926	0.1334	25.787	0.8797	0.1463	26.113	0.8884	0.1456	25.973	0.8847	0.1453
91	23.962	0.8169	0.1762	24.602	0.8390	0.1518	23.938	0.8207	0.1677	24.666	0.8407	0.1559	24.097	0.8298	0.1593
92	25.167	0.8769	0.1489	25.689	0.8897	0.1294	24.988	0.8772	0.1447	25.470	0.8876	0.1410	25.319	0.8831	0.1389
93	27.938	0.8422	0.2575	29.113	0.8749	0.2013	28.007	0.8591	0.2231	28.459	0.8493	0.2508	29.004	0.8572	0.2385
94	30.359	0.9450	0.0685	32.486	0.9620	0.0468	31.037	0.9535	0.0562	32.191	0.9614	0.0506	32.194	0.9609	0.0489
96	28.085	0.9266	0.1198	28.986	0.9402	0.0887	28.156	0.9317	0.1015	28.648	0.9322	0.1144	28.738	0.9342	0.1075
97	31.459	0.9429	0.1142	34.600	0.9638	0.0775	31.946	0.9523	0.0977	33.993	0.9627	0.0834	34.137	0.9587	0.0885

(continued on the next page)

Table 8 (continued)

	3DGS			Ours			AbsGS			MCMC			HoGS		
Idx	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS	PSNR	SSIM	LPIPS
98	30.051	0.9543	0.0691	32.720	0.9675	0.0504	31.809	0.9646	0.0518	32.842	0.9700	0.0470	32.679	0.9673	0.0487
99	28.514	0.9016	0.1540	29.510	0.9189	0.1192	28.506	0.9064	0.1409	29.241	0.9168	0.1306	29.181	0.9102	0.1361
100	32.339	0.9290	0.2051	33.269	0.9346	0.1898	32.569	0.9294	0.1965	32.794	0.9331	0.2018	33.197	0.9338	0.1913
101	28.048	0.9250	0.0892	29.222	0.9445	0.0615	28.383	0.9340	0.0708	29.056	0.9446	0.0612	28.717	0.9361	0.0696
102	33.186	0.9562	0.0893	34.425	0.9666	0.0626	33.363	0.9623	0.0695	33.206	0.9644	0.0674	34.053	0.9643	0.0674
103	30.716	0.9323	0.1562	31.786	0.9421	0.1312	30.910	0.9353	0.1434	31.444	0.9384	0.1471	31.376	0.9363	0.1455
104	31.209	0.9301	0.1361	33.673	0.9565	0.0639	32.040	0.9388	0.1171	33.342	0.9551	0.0708	33.428	0.9538	0.0693
105	31.348	0.9542	0.0880	36.740	0.9733	0.0532	32.702	0.9627	0.0701	35.295	0.9661	0.0780	36.064	0.9691	0.0662
106	27.611	0.8840	0.1561	28.482	0.9060	0.1206	27.594	0.8945	0.1339	28.141	0.8964	0.1424	27.979	0.8922	0.1433
107	31.997	0.9545	0.0861	33.454	0.9634	0.0642	32.695	0.9597	0.0702	33.105	0.9635	0.0697	32.379	0.9566	0.0768
108	31.482	0.9262	0.1759	32.920	0.9339	0.1431	31.318	0.9272	0.1591	32.037	0.9306	0.1625	32.373	0.9306	0.1576
109	27.538	0.8827	0.1739	29.573	0.9188	0.1144	27.775	0.8869	0.1638	29.051	0.9174	0.1185	29.060	0.9149	0.1166
110	33.475	0.9568	0.1016	34.135	0.9614	0.0853	33.475	0.9577	0.0941	33.460	0.9594	0.0924	33.794	0.9587	0.0980
111	32.349	0.9562	0.0462	33.749	0.9664	0.0336	32.981	0.9620	0.0382	33.224	0.9642	0.0383	33.656	0.9659	0.0337
112	29.362	0.9405	0.1103	30.876	0.9552	0.0832	30.059	0.9473	0.0953	30.359	0.9510	0.1009	30.452	0.9494	0.0966
113	32.307	0.9558	0.0801	32.927	0.9611	0.0665	32.330	0.9575	0.0743	32.411	0.9599	0.0718	32.759	0.9583	0.0720
114	29.995	0.9094	0.1102	31.644	0.9399	0.0676	30.897	0.9313	0.0772	31.903	0.9448	0.0651	31.391	0.9343	0.0778
115	32.033	0.9447	0.1385	32.922	0.9509	0.1231	31.981	0.9462	0.1311	31.824	0.9417	0.1500	32.653	0.9484	0.1301
116	37.891	0.9826	0.0271	39.081	0.9850	0.0202	37.939	0.9829	0.0254	37.867	0.9814	0.0348	38.954	0.9843	0.0232
117	33.413	0.9629	0.0672	34.270	0.9681	0.0541	33.530	0.9642	0.0620	33.931	0.9670	0.0595	34.115	0.9660	0.0588
118	35.370	0.9659	0.0608	36.223	0.9733	0.0426	35.751	0.9713	0.0467	36.089	0.9732	0.0454	36.032	0.9720	0.0457
119	26.914	0.8876	0.1678	27.464	0.8959	0.1513	26.822	0.8874	0.1634	27.082	0.8950	0.1603	27.316	0.8955	0.1575
120	25.429	0.8307	0.1748	25.746	0.8391	0.1604	25.283	0.8315	0.1673	25.721	0.8406	0.1640	25.581	0.8330	0.1668
121	32.025	0.9301	0.1821	33.298	0.9383	0.1490	31.170	0.9289	0.1658	31.928	0.9303	0.1751	32.426	0.9330	0.1661
122	28.416	0.9019	0.1218	29.083	0.9106	0.1077	28.518	0.9039	0.1163	28.779	0.9070	0.1201	29.167	0.9111	0.1069
123	26.932	0.9148	0.1068	27.956	0.9294	0.0861	26.927	0.9171	0.0987	28.368	0.9356	0.0913	26.793	0.9096	0.1083
124	31.216	0.9235	0.1563	31.809	0.9318	0.1358	31.255	0.9265	0.1434	31.640	0.9311	0.1439	31.522	0.9281	0.1421
125	30.998	0.9481	0.0870	33.216	0.9640	0.0590	31.313	0.9535	0.0747	32.203	0.9590	0.0705	32.781	0.9592	0.0688
126	21.247	0.7496	0.3280	21.588	0.7577	0.3064	20.756	0.7360	0.3267	21.468	0.7577	0.3335	21.624	0.7580	0.3152
127	34.289	0.9739	0.0630	35.689	0.9787	0.0487	34.350	0.9747	0.0591	35.038	0.9754	0.0739	34.526	0.9740	0.0605
128	27.975	0.9106	0.1212	28.984	0.9305	0.0892	28.316	0.9203	0.1014	28.842	0.9313	0.0900	28.407	0.9183	0.1083
129	32.507	0.9551	0.0748	33.344	0.9627	0.0589	32.451	0.9606	0.0542	32.858	0.9637	0.0555	32.363	0.9568	0.0650
130	26.960	0.9257	0.1210	28.470	0.9352	0.1047	27.016	0.9263	0.1146	27.932	0.9331	0.1142	27.112	0.9272	0.1168
131	27.679	0.9037	0.1192	28.585	0.9182	0.0932	27.491	0.9024	0.1148	28.235	0.9172	0.0945	28.209	0.9146	0.0965
132	30.815	0.9352	0.1313	31.408	0.9413	0.1141	30.743	0.9354	0.1271	30.951	0.9349	0.1379	31.198	0.9384	0.1257
133	36.048	0.9756	0.0566	38.118	0.9785	0.0552	32.200	0.9420	0.0874	37.214	0.9776	0.0568	36.550	0.9753	0.0582
134	37.408	0.9796	0.0407	37.830	0.9812	0.0318	37.150	0.9802	0.0356	37.079	0.9801	0.0380	37.555	0.9810	0.0331
135	29.593	0.9430	0.1147	30.679	0.9512	0.0979	29.585	0.9448	0.1062	30.265	0.9485	0.1104	30.346	0.9475	0.1062
136	27.085	0.9257	0.1387	28.102	0.9355	0.1138	27.252	0.9284	0.1271	28.124	0.9345	0.1292	27.706	0.9322	0.1253
137	26.314	0.8840	0.1777	30.447	0.9371	0.1069	27.250	0.9063	0.1284	30.252	0.9410	0.1008	30.926	0.9383	0.1215
138	26.970	0.8586	0.2259	27.330	0.8693	0.2059	26.834	0.8588	0.2187	27.289	0.8703	0.2149	27.361	0.8655	0.2155
139	24.667	0.8781	0.2154	25.938	0.8991	0.1836	25.255	0.8866	0.2019	25.989	0.8981	0.2044	25.690	0.8932	0.2004
Mean	29.291	0.9066	0.1324	30.335	0.9206	0.1089	29.327	0.9096	0.1220	29.921	0.9179	0.1190	29.981	0.9153	0.1183

D.2 Model Complexity

Table 9 reports the number of Gaussian primitives retained by each method on each dataset, as referred to in Section 5. Two patterns are worth noting. *Seed Exploration* slightly reduces the primitive count while improving PSNR, so its gain is not bought with capacity. *Split Exploration* reallocates rather than adds: the count falls on the simpler Deep Blending scenes and grows on the geometrically complex unbounded ones, where newly split primitives survive pruning. The GS-MCMC row is identical to *Seed & Split Exploration* by construction, since that baseline is assigned the same primitive budget, so the two rows should not be read as an independent efficiency comparison.

Table 9: **Gaussian primitive allocation (millions) per dataset.** Every entry is the final count after standard pruning, so the column for a dataset should be read together with the fidelity of the same method on that dataset. *GS-MCMC is assigned the same primitive budget as *Seed & Split Exploration*, so its counts are identical by construction.

Method	Mip-360	T & T	DB	OMMO	DL3DV
3DGS	2.72	1.57	2.48	1.78	1.14
AbsGS	3.13	1.31	1.93	1.73	1.05
HoGS (50K)	4.40	2.16	2.43	2.01	1.61
GS-MCMC*	2.53	2.11	**0.79**	1.77	2.11
Seed Exp.	2.62	1.51	2.33	1.68	1.11
Split Exp.	2.58	2.13	**0.86**	1.77	2.13
Seed & Split Exp.	2.53	2.11	**0.79**	1.77	2.11

D.3 Ablations

D.3.1 Depth Supervision versus Random Seeding

Figure 12 gives the qualitative comparison referred to in Section 5: pseudo-depth supervision partially mitigates the Far-Side Blur, but its effective range is limited by the inaccuracy of the pseudo-depth cues, whereas *Random Seeding* probes uncharted depth intervals directly.

Table 10 reports the corresponding quantitative comparison. Pseudo-depth supervision is a genuine depth constraint—it improves every dataset over the plain 3DGS baseline—yet it falls short of even the sparse *Random Seeding* operator, and it does not compound with exploration: adding it on top of *Seed & Split Exploration* leaves PSNR essentially unchanged on Tanks & Temples and costs accuracy on OMMO and DL3DV. The reason is the one that motivates the far-side analysis: pseudo-depth estimates become least reliable precisely at the distances where the Far-Side Blur Trap forms, so the constraint supplied there is the weakest, whereas *Random Seeding* reaches those depth intervals without relying on the estimate at all.

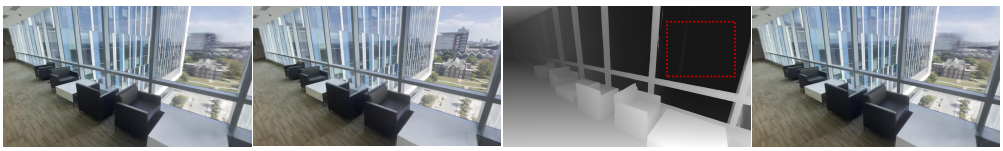

Figure 12: **Depth supervision versus random seeding. Left:** standard 3DGS leaves distant regions blurred. **Second:** *Random Seeding* recovers them. **Third:** the corresponding depth map. **Right:** pseudo-depth supervision helps at moderate distances but leaves the farthest regions blurred, as its estimates are least reliable exactly there.

Table 10: **Depth supervision versus random seeding.** A check mark enables the depth regularizer $\mathcal{L}_{\mathrm{D}}$, which integrates pseudo-depth maps into the standard 3DGS loss; the two halves of the table therefore differ only in whether that prior is active.

$\mathcal{L}_{\mathrm{D}}$	Method	Tanks & Temples			OMMO			DL3DV Benchmark		
		PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
✗	3DGS	23.74	0.853	0.169	30.50	0.920	0.142	27.16	0.861	0.167
	Seed Exp.	24.30	0.860	0.163	30.85	0.924	0.131	27.75	0.868	0.155
	Seed & Split Exp.	24.37	**0.871**	**0.138**	**31.27**	**0.930**	**0.119**	**28.46**	**0.889**	**0.125**
✓	3DGS	24.06	0.856	0.166	30.59	0.923	0.136	27.33	0.864	0.162
	Seed Exp.	24.19	0.859	0.162	30.81	0.925	0.132	27.76	0.865	0.155
	Seed & Split Exp.	**24.38**	0.869	0.140	31.06	0.928	0.123	28.39	0.888	0.127

D.3.2 Relaxed Pruning versus Random Seeding

Relaxing the pruning threshold of Appendix C.3 is necessary for *Random Seeding* but not sufficient by itself, and Figure 13 separates the two. Raising the threshold from $0.1\times$ to $10\times$ the scene extent

keeps alive the large primitives that the standard rule deletes, yet the distant region stays blurred. **Relaxing the threshold is thus *necessary but not sufficient* for escaping the Far-Side Blur Trap.** The threshold exists for the seed's sake: a seed carries a scale proportional to its distance from the origin (Algorithm 3), so the standard distance-independent rule deletes precisely the far-field seeds that explore the depths nothing has reached. What the far field lacks is not permission to hold large primitives but primitives to refine. What the relaxation leaves in place are primitives that were already there: large in scale, and equally without the depth-directed signal that would refine them, which is absent by construction (Section 3.2.1); the region therefore remains a set of soft outlines rather than resolved shapes even though it is no longer empty. Pruning can only remove what is there, never put anything there, and the same holds for relaxing it. Removing the obstacle therefore clears the way for the operator that does place primitives at those depths, and *Random Seeding* is what resolves the region.



Figure 13: **Keeping far Gaussians alive is not the same as reconstructing them.** Left to right: standard 3DGS pruning ($0.1\times$ scene extent), relaxed threshold ($10\times$) with seeding disabled, and the relaxed threshold with *Random Seeding*, on the same unbounded scene. The dashed box marks a distant region: the relaxed threshold keeps large far-field primitives alive, yet the region stays blurred, enough that the outline of the distant building cannot be made out. So the far-side failure is not an artifact of pruning, only *Random Seeding* reconstructs the outline with defined structure.

D.3.3 Positional Perturbation along the Viewing Ray

We adapt the positional perturbation mechanism (Kheradmand et al., 2024) from 3DGS-MCMC to the standard 3DGS pipeline, constraining the perturbation direction strictly along the viewing ray. This is the crudest possible depth-directed intervention—no new loss term is introduced—yet it already improves fidelity on unbounded scenes, as Figure 14 shows. It therefore serves as a lower bound on what explicit depth-directed exploration is worth: *Random Seeding* reaches the same goal without inheriting the relocation and noise machinery of the MCMC formulation.

D.3.4 Split Hyper-parameter Sweeps

Table 11 sweeps the splitting interval at a fixed `split.num` $= 20$ and Table 12 the number split per step at a fixed interval of 1, both on Tanks & Temples under a single seed. Each sweep intervenes on one term of the densification argument of Section 3.2.2, and together they account for the gain of *Random Splitting* from two directions.

Source 1: the averaging horizon. The densification schedule is untouched. Adaptive density control still fires at the same iterations, as many times as in 3DGS, so the interval changes not how often densification is decided but the statistic each decision reads. That statistic, $\|\partial L/\partial \mathbf{p}_{2D}\|$, is accumulated since the last merge of new primitives and zeroed by it, so its window is the spacing between merges: 100 iterations under adaptive density control alone, I once splitting runs every I steps, each contributing one randomly sampled view (Appendix C). Shortening the interval shortens the average, down to a single view. This is the quantity that Section 3.2.2 names as the cause of the Near-Side Blur Trap: a primitive hidden in some views and foreground in others is held below τ_{split} because the two are averaged together, not because its signal is missing. The sweep tests that reading rather than restating it: LPIPS falls by 16% as the window shrinks, while PSNR, SSIM and the primitive count change only moderately. The mechanism accounts for it directly: the screening metric becomes the instantaneous magnitude of one view, so the gradient-gated densification it thresholds fires where the primitive is in the foreground, and the occluded views that previously vetoed it no longer do.

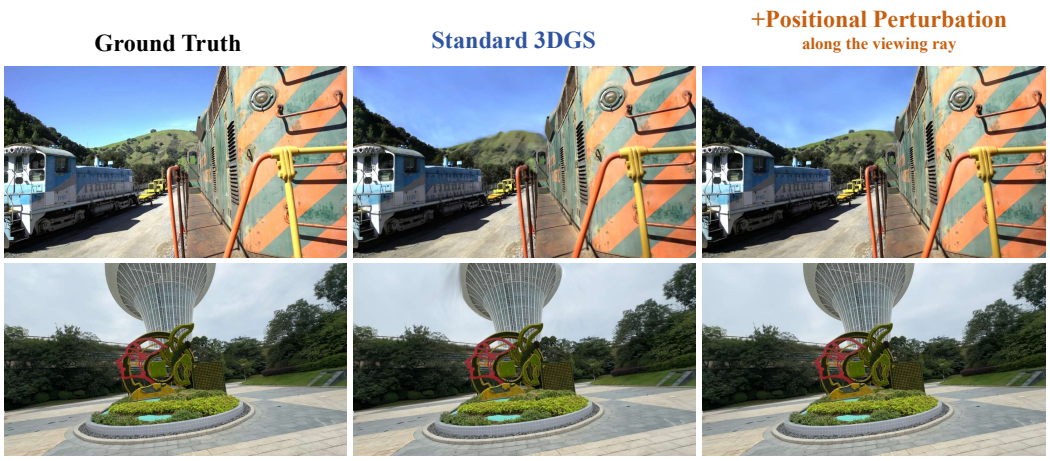

Figure 14: **Positional perturbation along the viewing ray.** Ground truth, standard 3DGS, and 3DGS with the positional perturbation of 3DGS-MCMC (Kheradmand et al., 2024) constrained to the viewing ray, shown on two unbounded scenes. No new loss term is introduced: only the perturbation direction is restricted to the ray.

Source 2: the split horizon. At the shortest interval the window already spans a single iteration, so Table 12 varies only the operator's own selection: it picks by scale rather than by gradient, and so reaches primitives that the densification criterion of Section 3.2.2 can hold below τ_{split} indefinitely. The two sources are the same failure seen from opposite sides: Source 1 restores the criterion by removing the cross-view average, Source 2 bypasses it, refining the primitives the criterion never selects. The table shows they complement rather than duplicate one another. It enable the operator raises PSNR and SSIM, raising its intensity keeps improving LPIPS at a growing primitive and time cost. So `split.num` sets an operating point on the accuracy–perception–cost trade-off, with the default at its accuracy peak.

Table 11: **Split interval ablation.** Tanks & Temples (train, truck, 2 scenes), seed 0; `split.num` = 20, 30k iterations. Mean over the two scenes.

Interval	PSNR (dB) ↑	SSIM ↑	LPIPS ↓	#G (M) ↓	Time (min) ↓
1	24.309	0.8683	0.1395	2.16	13.70
5	24.315	0.8719	0.1410	2.89	14.29
10	24.275	0.8696	0.1474	2.67	13.38
20	24.241	0.8667	0.1532	2.27	12.57
40	24.213	0.8649	0.1561	2.18	12.23
100	24.165	0.8606	0.1662	1.62	11.21

Table 12: **Split number ablation.** Tanks & Temples (train, truck, 2 scenes), seed 0; split interval 1, 30k iterations. Mean over the two scenes.

`split.num`	PSNR (dB) ↑	SSIM ↑	LPIPS ↓	#G (M) ↓	Time (min) ↓
0	24.102	0.8536	0.1474	1.91	13.09
20	24.309	0.8683	0.1395	2.16	13.70
50	24.244	0.8679	0.1374	2.45	14.08
100	24.153	0.8678	0.1358	2.98	17.34
200	24.058	0.8655	0.1345	4.02	16.76

D.3.5 Region-wise Effect of Each Exploration Mode

Table 13 breaks the DL3DV results down by region. The deficit sits in the far field: under standard 3DGS the farthest 5% of the depth range scores 29.10 dB against 29.45 dB for the rest. Both oper-

Table 13: **DL3DV (135 scenes): region-wise fidelity and cost.** PSNR/SSIM/LPIPS computed on the whole image, the *far* region (the farthest 5%). Cost columns give the mean final Gaussian count (millions) and mean training time (minutes).

	PSNR ↑			SSIM ↑			LPIPS ↓			Cost ↓	
Method	Overall	Far	Near	Overall	Far	Near	Overall	Far	Near	#G (M)	T (min)
3DGS	29.29	29.10	29.45	0.907	0.893	0.909	0.132	0.019	0.115	1.043	9.04
+ Random Seeding	29.73	30.17	29.73	0.911	0.913	0.911	0.127	0.015	0.113	1.021	10.05
+ Random Splitting	30.33	31.15	30.28	0.921	0.927	0.920	0.109	0.013	0.098	1.649	13.35
+ Both(**Ours**)	30.34	31.26	30.28	0.921	0.927	0.920	0.109	0.012	0.098	1.648	13.42



Figure 15: **Under-explored regions under three methods. Left:** standard 3DGS leaves distant, sparsely initialized regions blurred. **Middle:** *Seed & Split Exploration* keeps the 2D-gradient densification of 3DGS and adds seeding, so both sparse near-field and ultra-distant regions are populated. **Right:** GS-MCMC neither densifies the sparse transparent Gaussians by relocation nor reaches the ultra-distant regions.

ators improve that region, **though not in the same way**. *Random Seeding* gains 1.07 dB using 2% fewer primitives than the baseline; *Random Splitting* gains more, 2.05 dB, at 58% more primitives.

Splitting cannot put a primitive anywhere new. What the Far-Side Blur Trap is missing is exactly this: primitives at depths that nothing occupies. The position gradient has no component along the viewing ray (Equation 2), so far-side depth is fixed at initialization, and a split child lands within about one parent scale of its parent. Splitting makes a populated depth interval denser; it cannot populate one that is empty. Appendix D.3.2 makes the same point about pruning.

Denser depth intervals are enough to lift the regional averages. Doubling N_{split} doubles the primitive count while PSNR falls (Table 12). They are not enough to remove the blur, and the far-side row of Figure 7 shows this most clearly: with splitting alone the distant terrain is still smoothed, and it looks right only once seeding is in the model. A residual confined to a small part of the mask barely moves an average taken over the whole mask, which is why the two rows separate so little.

Seeding also keeps helping after splitting has acted. On all three far-side metrics the combination is the best of the four configurations, 31.26 dB and 0.012 LPIPS against 31.15 and 0.013 for splitting alone, and it does so on slightly fewer primitives. The two gains are sub-additive: 2.16 dB together where disjoint causes would give 3.12 dB.

D.4 Qualitative Comparisons

D.4.1 Under-Explored Regions: Where GS-MCMC Falls Short

Figure 15 gives the qualitative comparison referred to in Section 5: Standard 3DGS, *Seed & Split Exploration* and GS-MCMC are shown on the same near-field and far-field regions, illustrating that GS-MCMC's relocation and position noise remain anchored to locations the model has already reached.

D.4.2 The Blur Trap Is Inherited by 4DGS

Because 4D Gaussian Splatting (4DGS; Wu et al., 2024) shares the optimization pipeline of 3DGS, the mechanism analysed in Section 3 should apply to it unchanged. We therefore treat Neu3D (Li et al., 2022) as a *discriminative test* rather than a benchmark extension: its scenes span a comparatively bounded depth range, so the Near-Side Blur Trap should dominate and the Far-Side one should be largely absent. The resulting prediction about which of the two operators can help is specific, and falsifiable. As illustrated in Figure 16, regions classically susceptible to occlusion remain severely blurred in the 4DGS baseline despite extensive multi-view coverage, such as distant buildings visible through windows, text on indoor bottles, and steak textures dynamically obscured by flames. By the analysis of Section 3.2.2, these are precisely the primitives whose periodically accumulated 2D position gradient should be diluted below the splitting threshold.

The measurements match that prediction (Table 14, all rows re-trained on the same data): *Random Splitting*, applied sparsely ($N_{\text{split}} = 5$), already reduces LPIPS by 8.4% (vgg) and 18.9% (alex) over the 4DGS baseline. Enabling *Random Seeding* on top adds a comparable PSNR gain ($+0.265$ dB, against $+0.245$ dB for splitting alone) but moves LPIPS by less than 1%, which is what a bounded depth range implies: the far-side remedy has little perceptual headroom to recover. Two consequences follow. First, an intervention this sparse already determines the perceptual outcome, indicating that the bottleneck is the gradient-gated densification criterion rather than representational capacity. Second, because the same failure recurs, and yields to the same remedy, in a pipeline whose temporal model differs from 3DGS, the deficiency tracks the optimization rather than the 3D setting. Neu3D provides the narrowest possible depth range, and the two operators behave there exactly as the diagnosis predicts: a far-side remedy that has little perceptual headroom to recover, and a near-side remedy that determines the outcome almost on its own. We therefore read the 4DGS result not as a fourth benchmark but as evidence that the Blur Trap is a deficit of the 3DGS optimizer itself—inherited unchanged along with that optimizer—rather than of any particular 3D scene or representation.



Figure 16: **Random splitting in 4DGS. Top:** the 4DGS baseline leaves occluded regions underdensified, with buildings behind windows blurred despite multi-view coverage and flames blurring the food textures behind them. **Bottom:** a highly sparse *Random Splitting* ($N_{\text{split}} = 5$) refines those stagnated primitives across both static and dynamic occluders.

D.4.3 Per-Method Comparisons on Far and Near Regions

Figures 17 and 18 give the per-method qualitative comparisons referred to in Section 5, showing the Far-Side and Near-Side Blur Trap regions across all compared methods on the same scenes used for the quantitative evaluation.

Table 14: **4DGS on the Neu3D Dataset.** Occlusion-induced Near-Side blur remains the optimization bottleneck in Neu3D (Li et al., 2022), so *Random Splitting* delivers the perceptual gains while *Random Seeding* contributes mainly to PSNR. Neu3D scenes span a comparatively bounded depth range: consistently, splitting reduces LPIPS by 8.4% (vgg) and 18.9% (alex) over the 4DGS baseline, whereas additionally enabling seeding moves LPIPS by less than 1%. All rows are re-trained on the same Neu3D data under a common protocol.

Method	PSNR ↑	SSIM ↑	MS-SSIM ↑	D-SSIM ↓	LPIPS-vgg ↓	LPIPS-alex ↓
4DGS	30.575	0.9314	0.9659	0.0171	0.1507	0.0602
Split Exp.	30.820	0.9369	0.9689	0.0156	0.1380	0.0488
Seed & Split Exp.	**31.085**	**0.9379**	**0.9702**	**0.0149**	**0.1377**	**0.0483**



Figure 17: **Far-side regions across all compared methods.** Same scenes as the quantitative evaluation; every method is shown on the identical crop.

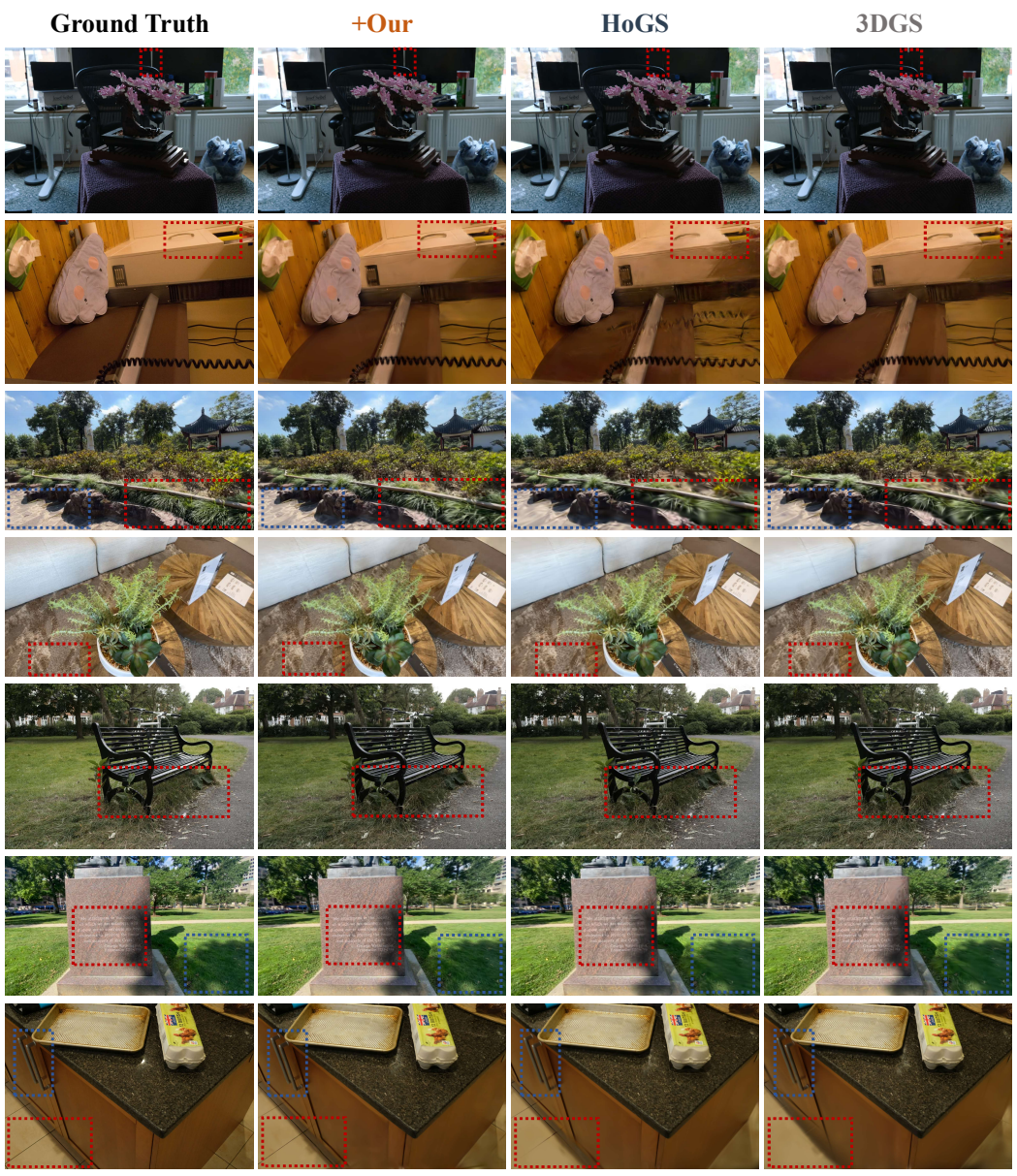

Figure 18: **Near-side regions across all compared methods.** Same scenes as the quantitative evaluation; every method is shown on the identical crop.