# Unified Loss-Aware Density Control for 3D Gaussian Splatting

**Tae-Young Kim** **Jiwoo Chung** **Jae-Pil Heo**[†]
Sungkyunkwan University
{vanmeruso, jiwoo.jg, jaepilheo}@gmail.com

## Abstract

The efficiency and fidelity of 3D Gaussian Splatting critically depend on adaptive density control, which grows and suppresses Gaussian primitives throughout optimization. However, the standard pipeline still relies on heuristic view-space gradients, scale thresholds, and global opacity resets, rather than explicitly assessing the loss-reducing effect of each split, clone, or opacity update. This mismatch can allocate primitives to already-sufficient regions, miss under-reconstructed structures, and destabilize training through blanket opacity changes. In this work, we recast adaptive density control as a collection of local loss-reduction decisions. We derive efficient Taylor-based per-Gaussian surrogates that predict the objective change caused by three primitive-level operations: splitting, cloning, and opacity decay. This yields a unified loss-aware framework with three components. First, we extend second-order splitting to a normalized joint position-scale space, enabling children to adapt not only their locations but also their anisotropic extents. Second, we formulate duplicate cloning as a surrogate minimization problem, selecting only Gaussians whose duplicated contribution is predicted to improve reconstruction and assigning each child a principled opacity. Third, we replace global opacity reset with selective soft opacity decay that weakens harmful primitives while preserving useful ones. Across standard 3DGS benchmarks, our method produces more targeted primitive allocation and achieves improved rendering quality with comparable or smaller Gaussian budgets compared to baselines.

## 1 Introduction

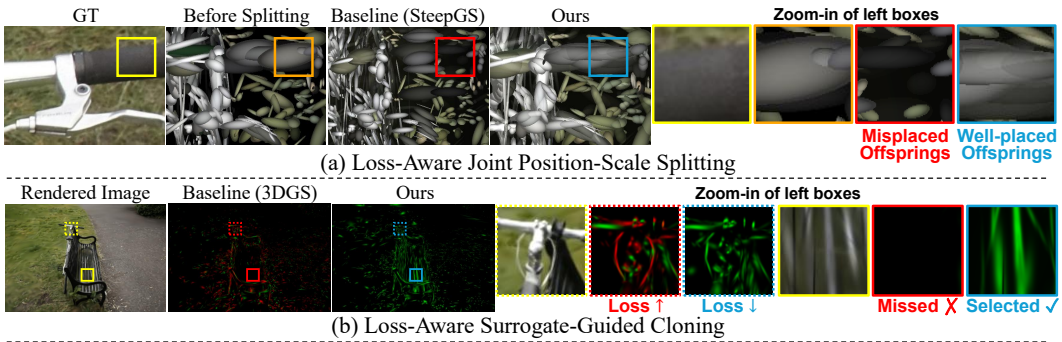
3D Gaussian Splatting (3DGS) [12] has emerged as an effective scene representation for high-quality, real-time novel view synthesis. Subsequent studies have improved the reconstruction quality and rendering efficiency of 3DGS through advances in primitive representations [10, 21], volumetric rendering [24, 31], anti-aliasing [38, 35], and level-of-detail (LOD) strategies for large-scale scenes [13].

Beyond these advances, reconstruction quality and representation complexity of 3DGS are also influenced by how Gaussian primitives are allocated and refined during training. Adaptive Density Control (ADC) is a key component of this process, progressively refining the scene representation through splitting, cloning, and pruning Gaussians. Despite its effectiveness, ADC still depends heavily on hand-crafted heuristic thresholds and positional gradients. These heuristic strategies can produce redundant Gaussian primitives even in regions where densification is unnecessary [30, 42, 19, 32], which increases memory usage and slows rendering as they accumulate.

To address this problem, prior works have revisited ADC from various perspectives [37, 11, 42, 14, 23, 39]. More recently, SteepGS [32] mitigates redundant primitive generation by predicting whether splitting each Gaussian locally reduces the reconstruction loss. Despite these advances, two

[†]Corresponding author

40th Conference on Neural Information Processing Systems (NeurIPS 2026).




(a) Loss-Aware Joint Position-Scale Splitting

(b) Loss-Aware Surrogate-Guided Cloning

(c) Loss-Aware Surrogate-Guided Opacity Decay

| Method | Splitting | Cloning | Opacity Update |
|---|---|---|---|
| 3DGS | Heuristic | Heuristic | Global Reset |
| SteepGS | Loss-Aware, Position-only | Heuristic | Global Reset |
| Ours | **Loss-Aware, Position-Scale** | **Loss-Aware** | **Loss-Aware** |

(d) Method Component Comparison

Figure 1: **Intuition of our loss-aware adaptive density control (ADC).** We visualize the effect of each density-control operation from the same intermediate state at the 3,000$^{\text{th}}$ training iteration. (a) Splitting: we compare the position-only split of [32] with our position-scale split, visualizing splitting Gaussians as ellipsoids. While the position-only split produces misplaced offspring around the thin bicycle handle, our position-scale split places offspring more appropriately and forms the handle structure more faithfully. (b) Cloning: we compare standard 3DGS and ours by individually cloning candidate Gaussians and measuring the resulting loss change; green/red indicate candidates whose duplication decreases/increases the loss. Our cloning selects mostly loss-reducing candidates and recovers missed structures, whereas heuristic cloning introduces loss-increasing candidates. (c) Opacity decay: we compare the opacity reset of standard 3DGS with our selective opacity decay. Opacity reset makes all Gaussian primitives nearly transparent, leading to unstable training. In contrast, our method weakens only Gaussians predicted to hinder optimization. (d) Component comparison: standard 3DGS relies on heuristics. SteepGS improves splitting with a position-only loss-aware rule but keeps cloning and opacity update heuristic. Our method applies loss-aware decisions to position-scale splitting, cloning, and opacity decay.

important aspects remain underexplored. In particular, existing loss-based splitting formulations mainly focus on position parameters, while the influence of other Gaussian parameters remains underexplored. Moreover, this loss-aware perspective has not yet been systematically extended to other density-control operations, such as cloning and opacity-based pruning, where conventional heuristic rules are still commonly used.

Building on these observations, we propose a loss-aware density control framework for efficient and faithful Gaussian representation. We first revisit splitting. While moving offspring centers refines local geometry, thin or anisotropic structures often require adapting offspring spatial extents in addition to positions. Thus, we extend splitting from a position-only formulation to a joint position-scale space, allowing each split to account for Gaussian extent. As shown in Fig. 1 (a), position-only splitting can leave gaps around thin structures such as a bicycle handle, whereas our scale-aware splitting jointly adjusts offspring positions and scales to form a more faithful local representation.

We further extend the loss-aware principle to cloning. Conventional cloning relies on view-space gradient and scale thresholds [12], without explicitly estimating whether a duplicate will reduce the photometric objective. Our surrogate-guided cloning instead predicts the loss change induced by each duplicate and clones only primitives whose duplicate contributions are expected to reduce the loss. As shown in Fig. 1 (b), standard 3DGS selects both green and red candidates in the zoom-in boxes. This mixed selection reveals that its heuristic cloning rule is not directly aligned with the loss effect of duplication. The baseline also misses important candidates around under-reconstructed structures such as the wooden frame, leaving these regions inadequately refined. By contrast, our method selects mostly green candidates and places clones in under-reconstructed regions, allowing the proposed surrogate to guide cloning toward primitives that are more likely to improve the objective.

A similar objective-agnostic issue appears in opacity-based pruning. Standard 3DGS periodically applies a global opacity reset, reducing the opacity of all Gaussians regardless of whether their current

contributions help or harm the objective. To address this, our loss-aware opacity decay predicts which Gaussians currently increase the loss and weakens only those primitives, while preserving useful Gaussians. As shown in Fig. 1 (c), the global opacity reset makes all Gaussian primitives nearly transparent and introduces abrupt changes that can destabilize training [30], whereas our opacity decay avoids unnecessary global perturbations and yields a more stable training trajectory.

However, directly evaluating the loss change of each candidate operation is computationally expensive, as it requires applying the operation, re-rendering affected views, and recomputing the loss. We therefore derive efficient Taylor-based per-Gaussian surrogates that approximate the objective change induced by splitting, cloning, and opacity decay [18, 34, 27, 28]. These surrogates provide a unified criterion for deciding whether each density-control operation is locally expected to reduce the loss.

Concretely, as shown in Fig. 1 (d), our framework consists of three loss-aware components: position-scale splitting, surrogate-guided cloning, and surrogate-guided opacity decay. The first adapts both offspring positions and spatial extents, the second selects loss-reducing duplicates and assigns adequate child opacity, and the third weakens Gaussians predicted to increase the objective. Together, these components form a unified loss-aware ADC strategy that allocates primitives more effectively and improves rendering quality with fewer Gaussians on standard 3DGS benchmarks.

## 2 Related works

### 2.1 Novel view synthesis

Novel View Synthesis (NVS) [1] aims to render images of a 3D scene from previously unseen viewpoints using given multi-view observations. Neural Radiance Fields (NeRF) [26] have emerged as a major milestone in NVS by representing scenes as continuous volumetric fields. However, their reliance on dense sampling and implicit representations often leads to high rendering costs. In contrast, 3D Gaussian Splatting (3DGS) [12] represents a scene explicitly as a set of anisotropic 3D Gaussians and combines this representation with differentiable, visibility-aware rasterization, enabling both efficient rendering and high-quality novel view synthesis.

Recent studies have further improved 3DGS through alternative primitive representations [10, 21], exact volumetric rendering formulations [31, 24], anti-aliasing techniques [38, 35], and hierarchical level-of-detail (LOD) strategies [13]. As a result of these advances, 3DGS has been actively studied across a wide range of tasks and application domains [20, 36, 17, 25, 7].

### 2.2 Adaptive Density Control in 3DGS

Adaptive density control (ADC) determines how Gaussian primitives are cloned, split, and pruned during 3DGS training, and therefore strongly affects both reconstruction quality and primitive efficiency. Since the original 3DGS relies on heuristic gradient and scale thresholds, recent works have improved the split and clone operations in ADC by refining densification signals [37, 41, 11, 42], introducing stochastic sampling-based updates [14, 15], or revising the densification procedure itself [6, 30, 5, 23].

The most relevant work to ours is SteepGS [32], which makes Gaussian splitting loss-aware using second-order information. However, its formulation is limited to position-only splitting, while other ADC operations such as cloning and opacity reset also remain heuristic. We extend this loss-aware perspective to the broader ADC pipeline through joint position-scale splitting and local surrogate-based cloning and opacity reduction.

## 3 Preliminary

### 3.1 3D Gaussian Splatting

3D Gaussian Splatting (3DGS) represents a scene as a collection of $n$ learnable anisotropic Gaussians,

$$\theta = \{\theta^{(i)}\}_{i=1}^{n}, \qquad \theta^{(i)} = \left(p^{(i)}, \Sigma^{(i)}, o^{(i)}, c^{(i)}\right), \tag{1}$$

where $p^{(i)} \in \mathbb{R}^3$, $\Sigma^{(i)} \in \mathbb{S}_+^{3\times3}$, $o^{(i)}$, and $c^{(i)}$ denote the center position, covariance, opacity, and appearance attributes of the $i^{\text{th}}$ Gaussian, respectively. Following the standard 3DGS parameterization,

each covariance matrix is expressed as $\Sigma^{(i)} = R^{(i)} S^{(i)} S^{(i)\top} R^{(i)\top}$, where $R^{(i)} \in SO(3)$ is a rotation matrix and $S^{(i)} = \mathrm{diag}(s^{(i)})$ is a diagonal scaling matrix with $s^{(i)} \in \mathbb{R}^3_+$. Under a camera projection $\varphi : \mathbb{R}^3 \to \mathbb{R}^2$, these Gaussians are projected on the image plane and composited in front-to-back depth order to render the color $C(x)$ at pixel $x \in \mathbb{R}^2$.

$$\alpha^{(i)}(x) = o^{(i)} \exp\left( -\frac{1}{2} \big(x - \varphi(p^{(i)})\big)^\top \left(\Sigma_\varphi^{(i)}\right)^{-1} \big(x - \varphi(p^{(i)})\big) \right). \tag{2}$$

$$T^{(i)}(x) = \prod_{j=1}^{i-1} \big(1 - \alpha^{(j)}(x)\big), \qquad C(x) = \sum_{i=1}^{n} c^{(i)}\, \alpha^{(i)}(x) T^{(i)}(x), \tag{3}$$

where $T^{(i)}(x)$ is the accumulated transmittance before the $i^{\text{th}}$ Gaussian, $\alpha^{(i)}(x)$ is its image-space opacity contribution, and $\Sigma_\varphi^{(i)}$ is its projected covariance on the image plane. Then, Gaussian parameters are optimized by minimizing the following photometric loss:

$$\mathcal{L}(\theta) = (1 - \lambda_{\text{D-SSIM}})\, \mathcal{L}_1(\theta) + \lambda_{\text{D-SSIM}}\, \mathcal{L}_{\text{D-SSIM}}(\theta), \tag{4}$$

where $\mathcal{L}_1$, $\mathcal{L}_{\text{D-SSIM}}$, and $\lambda_{\text{D-SSIM}}$ denote the pixel-wise L1 loss, SSIM loss [33], and control hyper-parameter between the two losses, respectively.

### 3.2 Adaptive Density Control

Along with standard gradient-based optimization, 3DGS [12] periodically employs Adaptive Density Control (ADC) to dynamically refine the set of Gaussians during training. ADC determines which Gaussians should be cloned, split, or pruned, and therefore has a direct effect on both reconstruction quality and primitive efficiency. In standard 3DGS, ADC first identifies Gaussians with large average view-space positional gradients as densification candidates. Among these candidates, Gaussians with small scales are cloned to improve local coverage, while Gaussians with large scales are split into smaller offspring. In addition, Gaussians with low opacity are pruned, and opacity values are periodically reset as a heuristic stabilization mechanism.

However, these ADC rules do not explicitly assess whether a candidate operation would reduce the training objective. Candidate Gaussians are selected using a hand-crafted threshold on the view-space gradient magnitude, and the choice between cloning and splitting is determined using a scale threshold. As a result, Gaussians may be densified even when they could be refined solely through ordinary gradient-based updates, thereby increasing the number of Gaussians unnecessarily.

### 3.3 SteepGS

SteepGS [32] introduces a loss-aware splitting strategy to address the limitations of heuristic density control in 3DGS. It observes that Gaussians can become stationary even in insufficiently reconstructed regions, limiting further loss reduction through gradient-based updates. SteepGS addresses this limitation by selectively splitting such Gaussians and displacing their offspring. To guide this process, it constructs a per-Gaussian splitting matrix that captures the second-order loss change induced by splitting. Using this matrix, it splits only Gaussians for which a local loss decrease is predicted. A detailed derivation of the splitting matrix is provided in Appendix A.3.

However, its loss-aware splitting considers only Gaussian position, while cloning and opacity reset still rely on heuristic rules. We address these limitations by extending loss-aware splitting to the joint position-scale space and introducing surrogate-guided cloning and opacity decay.

## 4 Methodology

### 4.1 Overview

Our goal is to replace the heuristic adaptive density control (ADC) decisions with loss-aware criteria for splitting, cloning, and opacity decay. As illustrated in Fig. 2, we accumulate per-Gaussian surrogate statistics during training and use them at each density-control interval to decide which operation should be applied to each Gaussian. Our framework consists of three components. First, in Sec. 4.2, we extend loss-aware splitting from a position-only formulation to a normalized joint

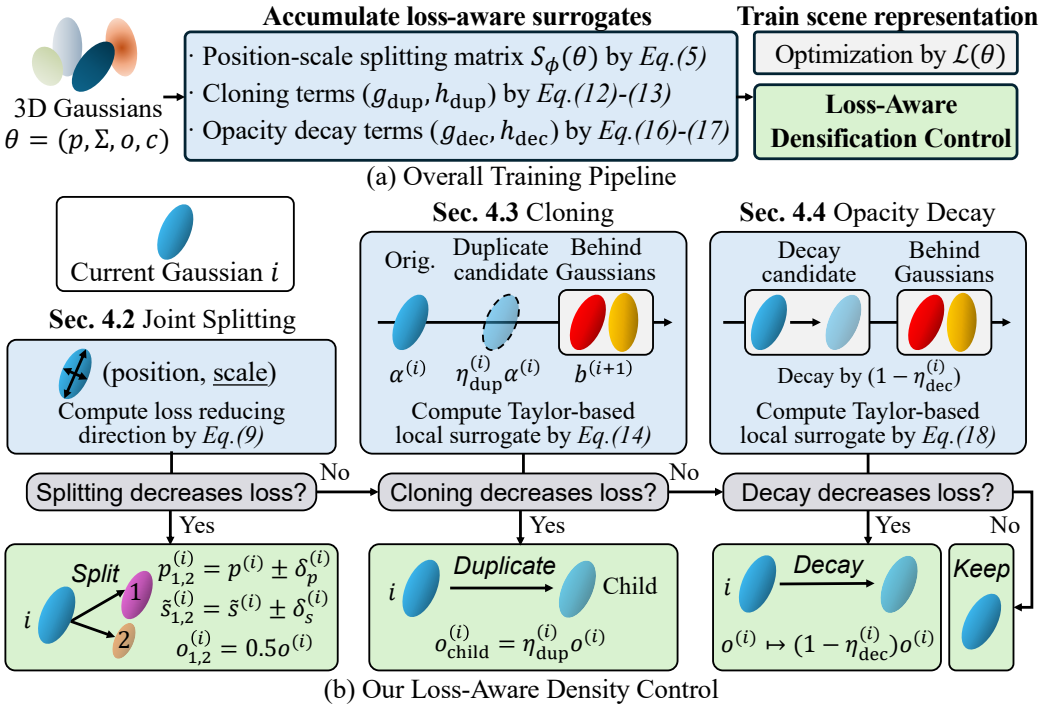



Figure 2: **Overview of our framework.** (a) During training, we accumulate per-Gaussian surrogate statistics, including the joint splitting matrix and surrogate terms for cloning and opacity decay. At each densification interval, these surrogates determine whether each Gaussian is split, cloned, or assigned opacity decay. (b) For each Gaussian, we sequentially evaluate splitting, cloning, and opacity decay using accumulated loss-aware surrogate statistics. Joint splitting computes a loss-reducing direction in the joint position-scale space and initializes two children with corresponding position and log-scale perturbations. Cloning minimizes a local Taylor-based surrogate with respect to the duplicate opacity and appends a duplicate child when it is predicted to decrease the loss. Opacity decay similarly estimates a loss-reducing opacity decay strength and selectively decreases the opacity of Gaussians whose current contributions are predicted to be harmful.

position-scale space, allowing offspring to adapt both their locations and spatial extents. Second, in Sec. 4.3, we introduce surrogate-guided cloning, which appends a duplicate child only when its contribution is predicted to decrease the loss and assigns its opacity from the same local surrogate. Third, in Sec. 4.4, we replace global opacity reset with surrogate-guided opacity decay, which weakens only Gaussians whose current contributions are predicted to increase the objective.

### 4.2 Loss-Aware Joint Position-Scale Splitting

Our baseline SteepGS [32] interprets Gaussian splitting from a loss-aware perspective and identifies directions that can reduce the reconstruction loss when each Gaussian is split. Specifically, SteepGS defines the split variable as the Gaussian position and therefore focuses only on offspring displacement in the position space. However, the rendering contribution of a Gaussian depends not only on its position but also on its scale, which determines its spatial extent. Therefore, effective splitting requires jointly adapting the positions and scales of the offspring to better capture local scene structure.

To formalize this idea, we build on the loss-aware splitting matrix of SteepGS [32] and generalize its split variable from the position space to the joint position-scale space. For each Gaussian $i$, we define the splitting matrix $\mathcal{S}^{(i)}_\phi$, which measures the local effect of splitting Gaussian $i$ through the curvature of its image-space contribution $\alpha_i(x)$ with respect to a local split variable $\phi^{(i)}$:

$$\phi^{(i)} = \begin{bmatrix} p^{(i)} \\ \tilde{s}^{(i)} \end{bmatrix}, \quad \tilde{s}^{(i)} = \log s^{(i)}, \qquad \mathcal{S}^{(i)}_\phi(\theta) = \mathbb{E}_{(\varphi,x)}\left[ \frac{\partial \mathcal{L}_{\varphi,x}}{\partial \alpha^{(i)}(x)} \nabla^2_{\phi^{(i)}} \alpha^{(i)}(x) \right]. \tag{5}$$

where $\tilde{s}^{(i)}$ denotes the log-scale vector associated with the Gaussian scale $s^{(i)}$, $\mathcal{L}_{\varphi,x}$ denotes the per-pixel photometric loss for camera $\varphi$ and pixel $x$, and the expectation is taken over sampled training views and pixels. Since $\phi^{(i)}$ contains both position and log-scale variables, the resulting position-scale splitting matrix $\mathcal{S}_{\phi}^{(i)}(\theta)$ lies in $\mathbb{R}^{6\times 6}$. A detailed derivation of the original SteepGS splitting matrix is provided in Appendix A.3.

However, directly applying eigendecomposition to this raw matrix is inappropriate, since the position and log-scale blocks have different units and numerical scales, making their eigenvectors not directly comparable. We therefore evaluate split directions in a normalized joint coordinate system as follows:

$$L^{(i)} = R^{(i)}\mathrm{diag}(s^{(i)}), \qquad J^{(i)} = \begin{bmatrix} L^{(i)} & 0 \\ 0 & \frac{1}{\sqrt{2}}\mathbf{I} \end{bmatrix}, \qquad \tilde{\mathcal{S}}_{\phi}^{(i)} = J^{(i)\top}\bar{\mathcal{S}}_{\phi}^{(i)}J^{(i)}. \tag{6}$$

where $J^{(i)}$ maps a normalized joint direction to a perturbation in the position-log-scale space, $L^{(i)}$ is the covariance factor of the $i^{\text{th}}$ Gaussian satisfying $L^{(i)}L^{(i)\top} = \Sigma^{(i)}$, and $\mathbf{I}$ is the $3\times 3$ identity matrix. This transformation enables split directions to be compared in a normalized joint position-scale space. The detailed derivation of Eq. 6 is provided in Appendix A.4. In this normalized space, we determine the split direction by computing the minimum eigenpair of $\tilde{\mathcal{S}}_{\phi}^{(i)}$:

$$(\lambda_{\min}^{(i)}, v_{\min}^{(i)}) = \mathrm{eigmin}\left(\tilde{\mathcal{S}}_{\phi}^{(i)}\right), \tag{7}$$

where $\lambda_{\min}^{(i)}$ is the smallest eigenvalue and $v_{\min}^{(i)}$ is the corresponding normalized joint direction. Since $v_{\min}^{(i)}$ gives the direction of smallest predicted second-order loss change, a negative $\lambda_{\min}^{(i)}$ indicates a local loss-decreasing split direction. This is especially useful for nearly stationary Gaussians, where gradient-based updates make little progress and the negative-curvature direction moves the offspring away from the stationary region [32]. We therefore split Gaussian $i$ only when it is nearly stationary, has unresolved view-space variation, and exhibits sufficiently negative curvature:

$$\mathcal{I}_{\text{split}} = \left\{ i \;\middle|\; \left(\|\bar{g}_{\text{pos}}^{(i)}\|_2 \le \tau_{\text{pos}}\right) \wedge \left(\|\bar{g}_{\text{scale}}^{(i)}\|_2 \le \tau_{\text{scale}}\right) \wedge \left(\lambda_{\min}^{(i)} < \tau_{\text{split}}\right) \right\}. \tag{8}$$

where $\mathcal{I}_{\text{split}}$ denotes the set of Gaussians selected for splitting, and $\bar{g}_{\text{pos}}^{(i)}$ and $\bar{g}_{\text{scale}}^{(i)}$ are the position and log-scale gradients of Gaussian $i$, respectively. The thresholds $\tau_{\text{pos}}$ and $\tau_{\text{scale}}$ are used to test whether the Gaussian is nearly stationary with respect to position and scale updates. The negative threshold $\tau_{\text{split}}$ prevents splits caused by noisy curvature arising from the approximation process. Once a Gaussian satisfies these criteria, its offspring are initialized along the joint split direction:

$$p_{1,2}^{(i)} = p^{(i)} \pm \delta_p^{(i)}, \quad \tilde{s}_{1,2}^{(i)} = \tilde{s}^{(i)} \pm \delta_{\tilde{s}}^{(i)}, \quad o_{1,2}^{(i)} = \frac{1}{2}o^{(i)}, \quad \begin{bmatrix} \delta_p^{(i)} \\ \delta_{\tilde{s}}^{(i)} \end{bmatrix} = \delta\phi^{(i)} = J^{(i)}v_{\min}^{(i)}, \tag{9}$$

where $\delta_p^{(i)}$ and $\delta_{\tilde{s}}^{(i)}$ are the position and log-scale displacements. The two children inherit the parent rotation and appearance. By defining the split direction in the joint position-scale space, our splitting rule estimates not only loss-aware position displacements $\delta p^{(i)}$, but also loss-aware scale refinements $\delta\tilde{s}^{(i)}$, enabling the offspring to better adapt to the local reconstruction geometry.

### 4.3 Loss-Aware Surrogate-Guided Cloning

We next introduce local surrogate-based rules for cloning and opacity decay. Although the two operations are different, they both modify the opacity contribution of a Gaussian. Specifically, cloning adds a new contribution by appending an exact duplicate at the same location, whereas opacity decay removes part of the current contribution. This shared structure allows both operations to be modeled with a quadratic approximation around zero operation strength. For Gaussian $i$, we define the cloning operation $\mathcal{C}(\theta;\eta_{\text{dup}}^{(i)})$ following the standard 3DGS, except that the duplicate child's opacity is explicitly parameterized by a relative opacity variable $\eta_{\text{dup}}^{(i)}$:

$$\mathcal{C}(\theta;\eta_{\text{dup}}^{(i)}) = \left(p^{(i)}, \Sigma^{(i)}, \eta_{\text{dup}}^{(i)}\, o^{(i)}, c^{(i)}\right), \qquad \eta_{\text{dup}}^{(i)} \in [0,1], \tag{10}$$

where $\mathcal{C}(\theta;\eta_{\text{dup}}^{(i)})$ denotes the duplicate child candidate with relative opacity $\eta_{\text{dup}}^{(i)}$. When $\eta_{\text{dup}}^{(i)} = 1$, it coincides with standard 3DGS cloning. To derive a local surrogate, we analyze its image-space

effect. Following Eq. 3, we define the image-space effect of Gaussian $i$ at pixel $x$ as $\Delta^{(i)}(x)$, and the per-unit duplicate contribution as $\Delta^{(i)}_{dup}(x)$:

$$\Delta^{(i)}(x) = T^{(i)}(x)\,\alpha^{(i)}(x)\,\big(c^{(i)} - b^{(i+1)}(x)\big), \qquad \Delta^{(i)}_{\mathrm{dup}}(x) = \big(1-\alpha^{(i)}(x)\big)\Delta^{(i)}(x), \tag{11}$$

where $b^{(i+1)}(x)$ denotes the background-inclusive color composited from all Gaussians behind Gaussian $i$. Specifically, $\Delta^{(i)}(x)$ measures the image-space effect induced by Gaussian $i$, and $\Delta^{(i)}_{\mathrm{dup}}(x)$ denotes the image-space change caused by adding a duplicate of Gaussian $i$ with unit opacity weight. Thus, assigning opacity $\eta^{(i)}_{\mathrm{dup}}$ to the child induces the perturbation $\eta^{(i)}_{\mathrm{dup}}\Delta^{(i)}_{\mathrm{dup}}(x)$. Using this perturbation, we define a Taylor-based local surrogate for cloning:

$$\mathcal{L}\Big(\theta \cup \Big\{\mathcal{C}(\theta;\eta^{(i)}_{\mathrm{dup}})\Big\}\Big) = \mathcal{L}(\theta) + \eta^{(i)}_{\mathrm{dup}}\, g^{(i)}_{\mathrm{dup}} + \frac{1}{2}(\eta^{(i)}_{\mathrm{dup}})^2 h^{(i)}_{\mathrm{dup}} + O((\eta^{(i)}_{\mathrm{dup}})^3) \tag{12}$$

$$g^{(i)}_{\mathrm{dup}} = \sum_{x,c} \frac{\partial \mathcal{L}}{\partial I_c(x)} \Delta^{(i)}_{\mathrm{dup},c}(x), \qquad h^{(i)}_{\mathrm{dup}} = \sum_{x,c} \omega_c(x) \left(\Delta^{(i)}_{\mathrm{dup},c}(x)\right)^2, \tag{13}$$

where the summation is over all image pixels $x$ and color channels $c$, with nonzero terms only for pixels affected by Gaussian $i$. $I_c(x)$ denotes the rendered color value at pixel $x$ in channel $c$, $\Delta^{(i)}_{\mathrm{dup},c}(x)$ is the corresponding channel of $\Delta^{(i)}_{\mathrm{dup}}(x)$, $\omega_c(x)$ denotes the proxy curvature weight described in Appendix A.5, and $O((\eta^{(i)}_{\mathrm{dup}})^3)$ denotes the omitted higher-order remainder.

This surrogate estimates the loss variation induced by assigning a small relative opacity to the duplicate child. Based on this local approximation, we determine whether a duplicate can reduce the current loss by minimizing the quadratic surrogate with respect to the clone strength. This gives the predicted strength $\eta^{(i)}_{\mathrm{dup}} = -g^{(i)}_{\mathrm{dup}}/h^{(i)}_{\mathrm{dup}}$. We append the duplicate child with strength $\eta^{\mathrm{dup}}$ only when the local surrogate predicts a valid loss decrease within the small-perturbation regime:

$$\begin{aligned} \mathcal{I}_{\mathrm{clone}} = \Big\{ i \;\Big|\; (i \notin \mathcal{I}_{\mathrm{split}}) \wedge \Big(g^{(i)}_{\mathrm{dup}} < 0\Big) \wedge \Big(\tau_{\mathrm{clone}} < \eta^{(i)}_{\mathrm{dup}} \le 10^{-2}\Big)\Big\}, \\ \theta \leftarrow \theta \cup \Big\{ \big(p^{(i)}, \Sigma^{(i)}, \eta^{(i)}_{\mathrm{dup}}\, o^{(i)}, c^{(i)}\big) \;\Big|\; i \in \mathcal{I}_{\mathrm{clone}} \Big\}. \end{aligned} \tag{14}$$

where $\mathcal{I}_{\mathrm{clone}}$ denotes the set of Gaussians selected for duplicate cloning. The threshold $\tau_{\mathrm{clone}}$ filters out negligible clone strengths, while the upper bound restricts the duplicate operation to a small perturbation regime in which the Taylor-based surrogate remains reliable. This selective cloning expands only Gaussians whose duplicate contribution is predicted to reduce the loss, helping avoid redundant Gaussian generation.

### 4.4 Loss-Aware Surrogate-Guided Opacity Decay

As the counterpart to cloning, opacity decay decreases part of the current opacity contribution of a Gaussian. Instead of applying a global opacity reset, we selectively reduce the opacity of Gaussians whose current contributions are predicted to hinder optimization. For Gaussian $i$, we define the opacity decay operation $\mathcal{P}(\theta;\eta^{(i)}_{\mathrm{dec}})$ by reducing only its opacity while preserving the other parameters:

$$\mathcal{P}(\theta;\eta^{(i)}_{\mathrm{dec}}) = \theta\big[o^{(i)} \mapsto (1-\eta^{(i)}_{\mathrm{dec}})o^{(i)}\big], \qquad \eta^{(i)}_{\mathrm{dec}} \in [0,1]. \tag{15}$$

where $\eta^{(i)}_{\mathrm{dec}}$ denotes the fraction by which the opacity is reduced. To derive a local surrogate, we use the image-space effect $\Delta^{(i)}(x)$ defined in Eq. 11. Unlike cloning, opacity decay does not introduce a new Gaussian or an additional image-space term, but simply reduces the existing contribution of the selected Gaussian. Therefore, applying opacity decay with strength $\eta^{(i)}_{\mathrm{dec}}$ to Gaussian $i$ induces the image-space perturbation $-\eta^{(i)}_{\mathrm{dec}}\Delta^{(i)}(x)$ at pixel $x$. Using this perturbation, we define a Taylor-based local surrogate for opacity decay:

$$\mathcal{L}\Big(\mathcal{P}(\theta;\eta^{(i)}_{\mathrm{dec}})\Big) = \mathcal{L}(\theta) - \eta^{(i)}_{\mathrm{dec}}\, g^{(i)}_{\mathrm{dec}} + \frac{1}{2}{\eta^{(i)}_{\mathrm{dec}}}^2 h^{(i)}_{\mathrm{dec}} + O((\eta^{(i)}_{\mathrm{dec}})^3) \tag{16}$$

$$g^{(i)}_{\mathrm{dec}} = \sum_{x,c} \frac{\partial \mathcal{L}}{\partial I_c(x)} \Delta^{(i)}_{c}(x), \qquad h^{(i)}_{\mathrm{dec}} = \sum_{x,c} \omega_c(x) \left(\Delta^{(i)}_{c}(x)\right)^2, \tag{17}$$

Table 1: Quantitative comparison on standard 3DGS benchmarks. Num denotes average Gaussians (M); * follows the 3DGS+residual-splitting setting of [23] to isolate the effect of the proposed method.

| Dataset | Mip-NeRF360 | | | | Tanks&Temples | | | | Deep Blending | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | PSNR↑ | SSIM↑ | LPIPS↓ | Num | PSNR↑ | SSIM↑ | LPIPS↓ | Num | PSNR↑ | SSIM↑ | LPIPS↓ | Num |
| 3DGS [12] | 27.35 | 0.814 | 0.217 | 3.4 | 23.39 | 0.841 | 0.183 | 1.8 | 29.52 | 0.903 | 0.242 | 3.0 |
| AbsGS [37] | $27.47_{\pm 0.04}$ | $0.819_{\pm 0.00}$ | $0.191_{\pm 0.00}$ | 3.1 | $23.67_{\pm 0.06}$ | $0.852_{\pm 0.00}$ | $0.163_{\pm 0.00}$ | 1.3 | $29.62_{\pm 0.04}$ | $0.901_{\pm 0.00}$ | $0.236_{\pm 0.00}$ | 1.9 |
| PixelGS [41] | $27.58_{\pm 0.01}$ | $0.824_{\pm 0.00}$ | $0.190_{\pm 0.00}$ | 5.5 | $23.78_{\pm 0.04}$ | $0.857_{\pm 0.00}$ | $0.150_{\pm 0.00}$ | 4.5 | $28.93_{\pm 0.03}$ | $0.897_{\pm 0.00}$ | $0.248_{\pm 0.00}$ | 4.6 |
| DC4GS [11] | $27.59_{\pm 0.02}$ | $0.826_{\pm 0.00}$ | $0.189_{\pm 0.00}$ | 2.6 | $23.94_{\pm 0.04}$ | $0.857_{\pm 0.00}$ | $0.161_{\pm 0.00}$ | 1.1 | $29.65_{\pm 0.03}$ | $0.906_{\pm 0.00}$ | $0.235_{\pm 0.00}$ | 1.5 |
| GDAGS [42] | $27.50_{\pm 0.01}$ | $0.822_{\pm 0.00}$ | $0.192_{\pm 0.00}$ | 3.0 | $23.66_{\pm 0.06}$ | $0.852_{\pm 0.00}$ | $0.167_{\pm 0.00}$ | 1.4 | $29.61_{\pm 0.05}$ | $0.904_{\pm 0.00}$ | $0.236_{\pm 0.00}$ | 2.0 |
| ResGS* [23] | $27.65_{\pm 0.02}$ | $0.817_{\pm 0.00}$ | $0.218_{\pm 0.00}$ | 1.9 | $24.02_{\pm 0.03}$ | $0.851_{\pm 0.00}$ | $0.178_{\pm 0.00}$ | 1.0 | $29.54_{\pm 0.17}$ | $0.903_{\pm 0.00}$ | $0.239_{\pm 0.00}$ | 1.2 |
| SteepGS [32] | $27.05_{\pm 0.02}$ | $0.795_{\pm 0.00}$ | $0.248_{\pm 0.00}$ | 2.1 | $23.48_{\pm 0.04}$ | $0.839_{\pm 0.00}$ | $0.193_{\pm 0.00}$ | 1.3 | $29.59_{\pm 0.05}$ | $0.904_{\pm 0.00}$ | $0.251_{\pm 0.00}$ | 1.6 |
| Ours | $27.72_{\pm 0.02}$ | $0.818_{\pm 0.00}$ | $0.219_{\pm 0.00}$ | 1.9 | $24.20_{\pm 0.02}$ | $0.853_{\pm 0.00}$ | $0.174_{\pm 0.00}$ | 1.0 | $29.79_{\pm 0.09}$ | $0.906_{\pm 0.00}$ | $0.245_{\pm 0.00}$ | 1.3 |

where $\Delta_c^{(i)}(x)$ denotes the $c^{\text{th}}$ channel of $\Delta^{(i)}(x)$. To determine whether reducing the opacity of Gaussian primitive $i$ can locally reduce the current loss, we follow the same local minimization as in cloning. This gives the predicted decay strength $\eta_{\text{dec}}^{(i)} = g_{\text{dec}}^{(i)}/h_{\text{dec}}^{(i)}$. We then apply opacity decay only when this strength is positive and remains within a small perturbation range:

$$\begin{aligned}\mathcal{I}_{\text{decay}} = \Big\{ i \;\Big|\; (i \notin \mathcal{I}_{\text{split}}) \wedge (i \notin \mathcal{I}_{\text{clone}}) \wedge \Big(g_{\text{dec}}^{(i)} > 0\Big) \wedge \Big(\tau_{\text{decay}} < \eta_{\text{dec}}^{(i)} \le 10^{-2}\Big)\Big\}, \\ \theta \leftarrow \theta\big[o^{(i)} \mapsto (1-\eta_{\text{dec}}^{(i)})o^{(i)} \;\big|\; i \in \mathcal{I}_{\text{decay}}\big].\end{aligned} \tag{18}$$

where $\mathcal{I}_{\text{decay}}$ denotes the set of Gaussians selected for opacity decay. The threshold $\tau_{\text{decay}}$ filters out negligible opacity-decay strengths, while the upper bound restricts the opacity update to a sufficiently local change where the Taylor-based surrogate is expected to remain accurate. This selective formulation reduces opacity only for Gaussians whose current contributions are predicted to increase the loss. In contrast to the global opacity reset used in 3DGS, it provides a more targeted update and avoids abrupt discontinuities in the training trajectory.

# 5 Experiments

## 5.1 Implementation Details

**Benchmarks.** We evaluate our method on three widely used real-world benchmarks for 3D Gaussian Splatting, namely Mip-NeRF360 [3], Tanks&Temples [16], and Deep Blending [8]. Following the standard 3DGS evaluation protocol, we use all scenes from Mip-NeRF360, the Train and Truck scenes from Tanks&Temples, and the DrJohnson and Playroom scenes from Deep Blending. We use the image resolution settings commonly adopted in prior 3DGS works. Specifically, outdoor scenes in Mip-NeRF360 are downsampled by a factor of 4, indoor scenes by a factor of 2, and the remaining datasets are evaluated at their original resolution.

**Training configuration.** Our training setup follows the original 3DGS configuration [12] and uses the same hyperparameters across all scenes and benchmarks, with detailed settings provided in Appendix A.1. All experiments are conducted on a single NVIDIA A6000 GPU. For evaluation, we report PSNR, SSIM [33], LPIPS [40], and the average number of Gaussians. To ensure a fair comparison, we reproduce all baselines using their official implementations under the same software environment and evaluation protocol. Each experiment is repeated five times, and we report the mean and standard deviation across runs, with the standard deviation rounded to two decimal places.

## 5.2 Experimental Results

**Quantitative results.** In Tab. 1, our method achieves the best PSNR across all datasets while using substantially fewer Gaussians than standard 3DGS. Heuristic ADC can introduce redundant primitives in regions that do not necessarily require densification, whereas our loss-aware surrogates apply splitting, cloning, and opacity decay only when they are predicted to reduce the objective. Although DC4GS [11] achieves marginally better SSIM and LPIPS in some cases, our method uses fewer Gaussians and obtains higher PSNR across all datasets, indicating a stronger reconstruction-compactness trade-off. Compared with SteepGS [32], the closest loss-aware density-control baseline, our method consistently improves reconstruction quality with a smaller number of Gaussians. These results validate the effectiveness of our loss-aware density control.

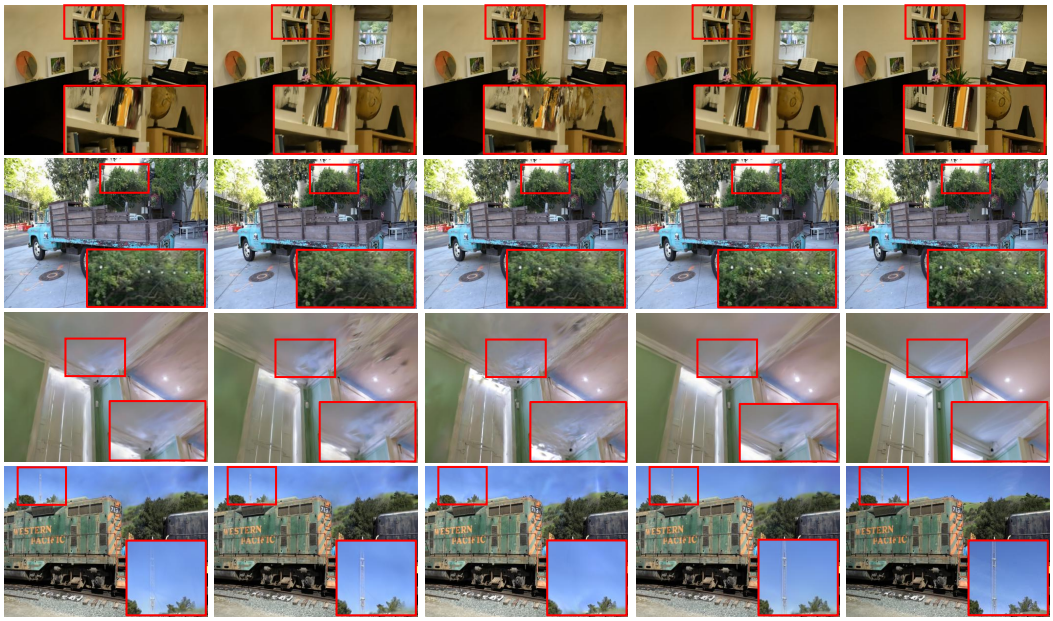



Figure 3: Qualitative comparison on standard 3DGS benchmarks. Red boxes highlight zoom-in regions where our method better preserves fine details and object boundaries.

Table 2: Ablation study of the proposed components on Mip-NeRF360 [3].

| Method | PSNR↑ | SSIM↑ | LPIPS↓ | Num | Method | PSNR↑ | SSIM↑ | LPIPS↓ | Num |
|---|---|---|---|---|---|---|---|---|---|
| (a): Baseline [32] | 27.05 | 0.795 | 0.248 | 2.1 | (c): (b) + Sec 4.3 | 27.37 | 0.813 | 0.229 | **1.6** |
| (b): (a) + Sec 4.2 | 27.10 | 0.810 | 0.230 | 2.0 | (d): (c) + Sec 4.4 | **27.72** | **0.818** | **0.219** | 1.9 |

**Qualitative results.** In Fig. 3, our method preserves fine structures and object boundaries more faithfully than the baselines, even with a smaller Gaussian budget. Our loss-aware density control allocates primitives to visually important regions rather than simply increasing the number of Gaussians. For example, in the second row, the red highlighted region around the tree and white lamps shows that our method reconstructs all white lamps clearly. However, the baselines tend to miss or blur some of the white lamps. Similar improvements are observed in the red boxed regions of the last row, where our method successfully reconstructs the thin spire above the train, whereas the baselines produce blurry structures and partially fail to recover the detail. These results show that our method efficiently allocates Gaussian primitives to regions that require additional capacity.

### 5.3 Analysis

**Ablation study.** Tab. 2 evaluates the contribution of each component on Mip-NeRF360 [3]. Starting from the SteepGS baseline (a), adding joint position-scale splitting (b), surrogate-guided cloning (c), and surrogate-guided opacity decay (d) progressively improves all reconstruction metrics. In particular, the full model achieves the best PSNR, SSIM, and LPIPS among all variants. These results show that all key components play complementary roles in improving loss-aware density control.

To further isolate the contribution of joint position-scale splitting, we compare it with the position-only splitting of SteepGS [32], while keeping our surrogate-guided cloning, opacity decay, and all other training settings unchanged. As shown in Tab. 3, joint position-scale splitting consistently improves PSNR, SSIM, and LPIPS across all three benchmarks while using comparable numbers of Gaussians. These results demonstrate that jointly adapting offspring positions and scales improves reconstruction quality compared with position-only splitting.

**Validation of the loss-reducing densification.** To assess whether densification reduces the objective, we compute $\Delta\mathcal{L} = \mathcal{L}_{\text{after}} - \mathcal{L}_{\text{before}}$ at each densification step, where negative values indicate loss

Table 3: Direct comparison of position-only and joint position-scale splitting.

| Dataset | Mip-NeRF360 | | | | Tanks&Temples | | | | Deep Blending | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | PSNR↑ | SSIM↑ | LPIPS↓ | Num | PSNR↑ | SSIM↑ | LPIPS↓ | Num | PSNR↑ | SSIM↑ | LPIPS↓ | Num |
| Position-only [32] | 27.36 | 0.798 | 0.239 | 2.0 | 23.94 | 0.847 | 0.178 | 1.1 | 29.47 | 0.896 | 0.255 | 1.1 |
| Joint position-scale (Ours) | **27.72** | **0.818** | **0.219** | 1.9 | **24.20** | **0.853** | **0.174** | 1.0 | **29.79** | **0.906** | **0.245** | 1.3 |

Table 4: Comparison of training cost and rendering speed (FPS) on a single NVIDIA A6000 GPU.

| Method | Training time (min)↓ | Peak memory (GB)↓ | FPS↑ |
|---|---|---|---|
| 3DGS [12] | **32.5** | **9.91** | 157 |
| Ours | 47.5 | 11.32 | **232** |

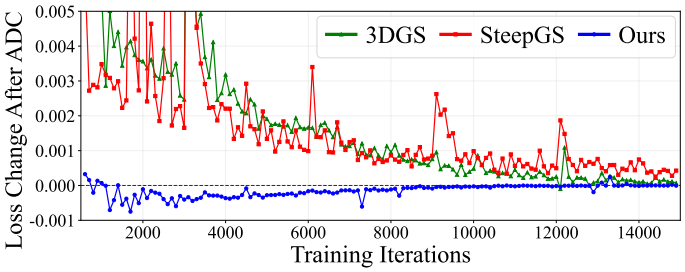



Figure 4: Loss change comparison after density-control.

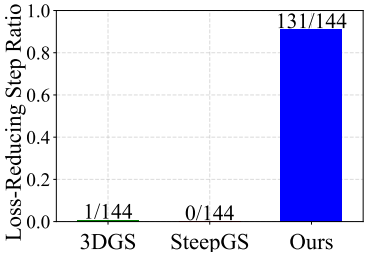



Figure 5: Loss-reducing step ratio.

reduction. Each marker in Fig. 4 denotes one densification step, and Fig. 5 reports the fraction of steps with $\Delta\mathcal{L} < 0$. We select a single viewpoint from the Counter scene of Mip-NeRF360 [3] and keep it fixed across all densification steps when measuring $\Delta\mathcal{L}$. As shown in Figs. 4 and 5, our method yields negative loss changes for most densification steps, showing that the proposed operations update Gaussians toward loss-reducing directions. In contrast, 3DGS does not consider the loss effect of densification, while SteepGS accounts for it only in splitting; thus, neither consistently reduces the loss across all density-control operations. These results show that our loss-aware densification more effectively reduces the actual training loss than conventional ADC.

**Computational cost.** Tab. 4 compares our method with 3DGS [12] in terms of training time, peak GPU memory usage during training, and rendering speed on a single NVIDIA A6000 GPU. Accumulating the joint position-scale splitting matrix and surrogate curvature statistics within the rasterizer increases training time and peak GPU memory usage. However, this overhead is limited to training; inference uses standard Gaussian rasterization without surrogate computation. Our more compact Gaussian representation enables faster rendering in this comparison, illustrating a trade-off between training cost and rendering efficiency.

## 6 Conclusion

We introduce a unified loss-aware density control framework for 3D Gaussian Splatting. Instead of relying on heuristic gradient, scale, and opacity reset rules, our method predicts the local loss effect of splitting, cloning, and opacity decay using efficient per-Gaussian surrogates. By combining joint position-scale splitting, surrogate-guided cloning, and opacity decay, the proposed method allocates primitives more effectively and suppresses harmful contributions more stably. Experiments across the benchmarks show that ours improves the quality-compactness trade-off with fewer Gaussians.

## 7 Acknowledgements

This work was supported in part by MSIT/IITP (No. RS-2022-II220680, RS-2020-II201821, RS-2019-II190421, RS-2024-00459618, RS-2024-00360227, RS-2026-25549037, RS-2025-25442569), MSIT/NIPA (No. RS-2026-25621569), MSIT/NRF (No. RS-2024-00357729), and KNPA/KIPoT (No. RS-2025-25393280).

## References


[1] Shai Avidan and Amnon Shashua. Novel view synthesis in tensor space. In *Proceedings of IEEE computer society conference on computer vision and pattern recognition*, pages 1034–1040. IEEE, 1997.

[2] Jonathan T Barron. A general and adaptive robust loss function. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4331–4339, 2019.

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

[4] Pierre Charbonnier, Laure Blanc-Féraud, Gilles Aubert, and Michel Barlaud. Deterministic edge-preserving regularization in computed imaging. *IEEE Transactions on image processing*, 6(2):298–311, 1997.

[5] Zhemeng Dong, Junjun Jiang, Youyu Chen, Jiaxin Zhang, Kui Jiang, and Xianming Liu. Reframing gaussian splatting densification with complexity-density consistency of primitives. *Advances in Neural Information Processing Systems*, 38:127410–127433, 2025.

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

[7] Xinyu Gao, Ziyi Yang, Bingchen Gong, Xiaoguang Han, Sipeng Yang, and Xiaogang Jin. Towards realistic example-based modeling via 3d gaussian stitching. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 26597–26607, 2025.

[8] 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.

[9] Lukas Höllein, Aljaž Božič, Michael Zollhöfer, and Matthias Nießner. 3dgs-lm: Faster gaussian-splatting optimization with levenberg-marquardt. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 26740–26750, 2025.

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

[11] Moonsoo Jeong, Dongbeen Kim, Minseong Kim, and Sungkil Lee. DC4GS: Directional Consistency-Driven Adaptive Density Control for 3D Gaussian Splatting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

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

[13] Bernhard Kerbl, Andreas Meuleman, Georgios Kopanas, Michael Wimmer, Alexandre Lanvin, and George Drettakis. A hierarchical 3d gaussian representation for real-time rendering of very large datasets. *ACM Transactions On Graphics (TOG)*, 43(4):1–15, 2024.

[14] 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.

[15] Hyunjin Kim, Haebeom Jung, and Jaesik Park. Metropolis-hastings sampling for 3d gaussian reconstruction. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[16] 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.

[17] Eunseo Koh, Sangeek Hyun, MinKyu Lee, Jiwoo Chung, Kangmin Seo, and Jae-Pil Heo. Diffusion feature field for text-based 3d editing with gaussian splatting. In *Advances in Neural Information Processing Systems*, volume 38, pages 159–178, 2025.

[18] Yann LeCun, John Denker, and Sara Solla. Optimal brain damage. *Advances in neural information processing systems*, 2, 1989.

[19] Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3d gaussian representation for radiance field. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21719–21728, 2024.

[20] Zhihao Liang, Qi Zhang, Ying Feng, Ying Shan, and Kui Jia. Gs-ir: 3d gaussian splatting for inverse rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21644–21653, 2024.

[21] Rong Liu, Dylan Sun, Meida Chen, Yue Wang, and Andrew Feng. Deformable beta splatting. In *Proceedings of the special interest group on computer graphics and interactive techniques conference conference papers*, pages 1–11, 2025.

[22] Linjie Lyu, Ayush Tewari, Jianchun Chen, Thomas Leimkühler, and Christian Theobalt. Faster 3D gaussian splatting convergence via structure-aware densification. In *ACM SIGGRAPH 2026 Conference Papers*, 2026. doi: 10.1145/3799902.3811212.

[23] Yanzhe Lyu, Kai Cheng, Xin Kang, and Xuejin Chen. Resgs: Residual densification of 3d gaussian for efficient detail recovery. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 28093–28102, 2025.

[24] Alexander Mai, Peter Hedman, George Kopanas, Dor Verbin, David Futschik, Qiangeng Xu, Falko Kuester, Jonathan T Barron, and Yinda Zhang. Ever: Exact volumetric ellipsoid rendering for real-time view synthesis. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4930–4939. IEEE, 2025.

[25] Hidenobu Matsuki, Riku Murai, Paul HJ Kelly, and Andrew J Davison. Gaussian splatting slam. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18039–18048, 2024.

[26] 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.

[27] Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. Pruning convolutional neural networks for resource efficient inference. In *International Conference on Learning Representations*, 2017.

[28] Pavlo Molchanov, Arun Mallya, Stephen Tyree, Iuri Frosio, and Jan Kautz. Importance estimation for neural network pruning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11264–11272, 2019.

[29] Hamza Pehlivan, Andrea Boscolo Camiletto, Lin Geng Foo, Marc Habermann, and Christian Theobalt. Matrix-free second-order optimization of gaussian splats with residual sampling. In *2026 International Conference on 3D Vision (3DV)*, pages 18–28. IEEE, 2026.

[30] Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. Revising densification in gaussian splatting. In *European Conference on Computer Vision*, pages 347–362. Springer, 2024.

[31] Chinmay Talegaonkar, Yash Belhe, Ravi Ramamoorthi, and Nicholas Antipa. Volumetrically consistent 3d gaussian rasterization. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10953–10963. IEEE, 2025.

[32] Peihao Wang, Yuehao Wang, Dilin Wang, Sreyas Mohan, Zhiwen Fan, Lemeng Wu, Ruisi Cai, Yu-Ying Yeh, Zhangyang Wang, Qiang Liu, et al. Steepest descent density control for compact 3d gaussian splatting. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 26663–26672, 2025.

[33] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[34] Lemeng Wu, Bo Liu, Peter Stone, and Qiang Liu. Firefly neural architecture descent: a general approach for growing neural networks. *Advances in neural information processing systems*, 33:22373–22383, 2020.

[35] Zhiwen Yan, Weng Fei Low, Yu Chen, and Gim Hee Lee. Multi-scale 3d gaussian splatting for anti-aliased rendering. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 20923–20931. IEEE, 2024.

[36] Mingqiao Ye, Martin Danelljan, Fisher Yu, and Lei Ke. Gaussian grouping: Segment and edit anything in 3d scenes. In *European conference on computer vision*, pages 162–179. Springer, 2024.

[37] 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*, pages 1053–1061, 2024.

[38] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-splatting: Alias-free 3d gaussian splatting. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19447–19456. IEEE, 2024.

[39] Jiahui Zhang, Fangneng Zhan, Muyu Xu, Shijian Lu, and Eric Xing. Fregs: 3d gaussian splatting with progressive frequency regularization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21424–21433, 2024.

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

[41] 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*, pages 326–342. Springer, 2024.

[42] 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 *International Conference on Learning Representations*, 2026.

# Appendix

## A More implementation detail

### A.1 Experiment Setting and Hyperparameters

We adopt the same training configuration as the original 3DGS, including learning rates, the optimization schedule, and the total number of training iterations. For density control, we follow the default 3DGS protocol: densification starts from iteration 500, is performed every 100 iterations, and is applied until iteration 15,000. The opacity threshold for pruning is also kept the same as in the original 3DGS.

All hyperparameters are fixed for every experiment across all scenes and benchmarks. For the reproduced baselines, we follow the official configurations provided by their released codebases. However, since some official implementations enable exposure compensation while others do not, we disable exposure compensation for all reproduced baselines and our method to ensure a fair comparison. For the 3DGS results in the main Tab. 1, we use the officially released pretrained models provided by 3DGS.

For all experiments, we empirically set the hyperparameters used in our method to $\tau_{\mathrm{pos}} = \tau_{\mathrm{scale}} = 10^{-4}$, which serve as thresholds for determining whether a Gaussian primitive is at a stationary point, and $\tau_{\mathrm{split}} = -10^{-8}$, $\tau_{\mathrm{clone}} = 10^{-6}$, and $\tau_{\mathrm{decay}} = 10^{-4}$, which are the main hyperparameters that significantly affect loss-aware densification. We also report a sensitivity analysis for the main hyperparameters in the densification process of our method, $\tau_{\mathrm{split}}$, $\tau_{\mathrm{clone}}$, and $\tau_{\mathrm{decay}}$, as shown in Tables 5, 6, and 7.

### A.2 Pseudo code

Algorithm 1 summarizes our loss-aware density control. Instead of relying on the heuristic ADC rules, we make per-Gaussian decisions using local loss surrogates for three operations: split, clone, and opacity decay. During training, we accumulate the joint position–scale splitting matrix together with the quadratic surrogate terms for cloning and opacity decay. At each density-control interval, these statistics are averaged and used to select split candidates, append beneficial duplicate children, and suppress harmful opacity contributions.

The three operations are applied sequentially, with splitting taking priority over cloning and opacity decay. This ordering follows the strength of the underlying criteria. The split decision is directly tied to the second-order splitting matrix and its minimum-eigenvalue direction, giving a principled descent direction for escaping stationary regions. In contrast, cloning and opacity decay are then applied to the remaining Gaussians, as they correspond to small opacity-contribution perturbations estimated with proxy-curvature quadratic surrogates. We therefore prioritize splitting and use the other two operations as residual actions for Gaussians that are not selected for geometry-aware refinement. Since all statistics and decisions are defined independently for each Gaussian, the procedure is naturally parallelizable. As in SteepGS [32], the statistics used by our method are computed within the rasterizer.

**Operation priority.** To examine the effect of the priority order, we evaluate all six permutations of splitting (S), cloning (C), and opacity decay (D). We use Bicycle from Mip-NeRF360, Truck from Tanks&Temples, and Dr. Johnson from Deep Blending, changing only the priority order while keeping all other settings unchanged. Tab. 8 reports the average results over these three scenes. The adopted Split→Clone→Decay order achieves the highest average PSNR and tied-best SSIM and LPIPS, providing empirical support for this ordering.

**Adaptive arbitration.** We also evaluate an adaptive loss-scale arbitration strategy that selects, for each Gaussian, the applicable operation with the largest predicted loss reduction. Under the same evaluation setting, our fixed-priority strategy achieves higher PSNR and SSIM and lower LPIPS than this variant (Tab. 9). A possible explanation is the different parameterizations and approximation scales of the operation-specific surrogates. Splitting is evaluated in the normalized joint position-scale space, whereas cloning and opacity decay use one-dimensional Taylor surrogates over opacity contributions. Thus, although each surrogate estimates the local loss effect of its corresponding operation, their raw values are not necessarily calibrated for direct cross-operation comparison.

Table 5: Experiment on the splitting hyperparameter. We set $\tau_{\text{split}} = -10^{-8}$ for default configuration.

| Dataset | Mip-NeRF360 | | | | Tanks&Temples | | | | Deep Blending | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | PSNR↑ | SSIM↑ | LPIPS↓ | Num | PSNR↑ | SSIM↑ | LPIPS↓ | Num | PSNR↑ | SSIM↑ | LPIPS↓ | Num |
| Ours ($\tau_{\text{split}} = -10^{-7}$) | 27.68 | 0.815 | 0.225 | 1.6 | 24.11 | 0.852 | 0.176 | 0.8 | 29.61 | 0.905 | 0.245 | 1.2 |
| Ours ($\tau_{\text{split}} = -10^{-8}$) | 27.72 | 0.818 | 0.219 | 1.9 | 24.20 | 0.853 | 0.174 | 1.0 | 29.79 | 0.906 | 0.245 | 1.3 |
| Ours ($\tau_{\text{split}} = -10^{-9}$) | 27.72 | 0.818 | 0.219 | 2.0 | 24.26 | 0.853 | 0.174 | 1.1 | 29.72 | 0.906 | 0.245 | 1.3 |

Table 6: Experiment on the clone hyperparameter. We set $\tau_{\text{clone}} = 10^{-6}$ for default configuration.

| Dataset | Mip-NeRF360 | | | | Tanks&Temples | | | | Deep Blending | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | PSNR↑ | SSIM↑ | LPIPS↓ | Num | PSNR↑ | SSIM↑ | LPIPS↓ | Num | PSNR↑ | SSIM↑ | LPIPS↓ | Num |
| Ours ($\tau_{\text{clone}} = 10^{-5}$) | 27.70 | 0.818 | 0.218 | 1.9 | 24.21 | 0.854 | 0.174 | 1.0 | 29.79 | 0.905 | 0.245 | 1.3 |
| Ours ($\tau_{\text{clone}} = 10^{-6}$) | 27.72 | 0.818 | 0.219 | 1.9 | 24.20 | 0.853 | 0.174 | 1.0 | 29.79 | 0.906 | 0.245 | 1.3 |
| Ours ($\tau_{\text{clone}} = 10^{-7}$) | 27.71 | 0.818 | 0.219 | 1.9 | 24.10 | 0.853 | 0.174 | 1.0 | 29.80 | 0.906 | 0.245 | 1.4 |

Calibrating these estimates for adaptive operation selection remains a promising direction for future work.

### A.3 Splitting Matrix of SteepGS

We review the splitting matrix of SteepGS [32] using the notation introduced in Sec. 4.2. Let $\Phi = \{\phi^{(i)}\}_{i=1}^{n}$ denote the collection of local split variables. In the original SteepGS implementation, this variable is chosen as the Gaussian center, i.e., $\phi^{(i)} = p^{(i)}$. More generally, the same derivation can be applied to an extended local variable, such as the joint position-scale variable used in Sec. 4.2, as long as the projected Gaussian contribution is twice differentiable with respect to that variable.

For a camera–pixel pair $(\varphi, x)$, we view the per-pixel objective as a function of the projected Gaussian contributions:

$$\mathcal{L}(\Phi) = \mathbb{E}_{(\varphi,x)}\left[\mathcal{L}_{\varphi,x}\big(\alpha^{(1)}(x;\phi^{(1)}),\dots,\alpha^{(n)}(x;\phi^{(n)})\big)\right]. \tag{19}$$

Here, $\alpha^{(i)}(x;\phi^{(i)})$ denotes the image-space contribution of Gaussian $i$, corresponding to $\sigma_{\varphi}(x;\theta^{(i)})$ in SteepGS [32]. The alpha-compositing operation, color attributes, and the comparison with the ground-truth pixel are absorbed into $\mathcal{L}_{\varphi,x}$. This abstraction allows us to analyze the local effect of splitting through the scalar contribution of each Gaussian. For the Taylor analysis below, we assume that the resulting objective is differentiable in the chosen local split variables.

We consider splitting Gaussian $i$ into $m_i$ offspring. The local split variable of the $j$-th offspring is denoted by $\phi_j^{(i)}$, and its opacity weight is denoted by $w_j^{(i)} \geq 0$. To preserve the local opacity mass, the weights satisfy

$$\sum_{j=1}^{m_i} w_j^{(i)} = 1. \tag{20}$$

Under this constraint, the parent contribution is locally replaced by a weighted sum of offspring contributions:

$$\alpha^{(i)}(x;\phi^{(i)}) \quad \longrightarrow \quad \sum_{j=1}^{m_i} w_j^{(i)}\alpha^{(i)}(x;\phi_j^{(i)}). \tag{21}$$

Therefore, the weight $w_j^{(i)}$ is absorbed into the opacity of the $j$-th offspring after splitting. Following SteepGS, we decompose each offspring displacement into a shared mean shift and an individual zero-mean splitting offset:

$$\phi_j^{(i)} = \phi^{(i)} + \varepsilon\big(\mu^{(i)} + \delta_j^{(i)}\big), \qquad \sum_{j=1}^{m_i} w_j^{(i)}\delta_j^{(i)} = 0. \tag{22}$$

The scalar $\varepsilon > 0$ controls the perturbation size. The vector $\mu^{(i)}$ is the weighted mean displacement of the offspring group, while $\delta_j^{(i)}$ describes the relative displacement of each offspring around this mean.

Table 7: Experiment on the decay hyperparameter. We set $\tau_{\text{decay}} = 10^{-4}$ for default configuration.

| Dataset | Mip-NeRF360 | | | | Tanks&Temples | | | | Deep Blending | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | PSNR↑ | SSIM↑ | LPIPS↓ | Num | PSNR↑ | SSIM↑ | LPIPS↓ | Num | PSNR↑ | SSIM↑ | LPIPS↓ | Num |
| Ours ($\tau_{\text{decay}} = 10^{-3}$) | 27.71 | 0.818 | 0.219 | 1.6 | 24.00 | 0.850 | 0.184 | 0.8 | 29.51 | 0.904 | 0.250 | 1.0 |
| Ours ($\tau_{\text{decay}} = 10^{-4}$) | 27.72 | 0.818 | 0.219 | 1.9 | 24.20 | 0.853 | 0.174 | 1.0 | 29.79 | 0.906 | 0.245 | 1.3 |
| Ours ($\tau_{\text{decay}} = 10^{-5}$) | 27.74 | 0.818 | 0.218 | 2.0 | 24.20 | 0.854 | 0.170 | 1.2 | 29.61 | 0.904 | 0.245 | 1.5 |

**Algorithm 1** Loss-Aware Density Control

**Input:** Initial Gaussians $\theta = \{\theta^{(i)}\}_{i=1}^{n}$; loss function $\mathcal{L}$; total iterations $T$; ADC interval $T_{\text{dc}}$.

**for** each training step $t = 1, \cdots, T$ **do**
  Sample a training view and update $\theta$ by standard gradient descent.
  Accumulate per-Gaussian loss-aware statistics for splitting, cloning, and opacity decay.
  **if** $t \bmod T_{\text{dc}} = 0$ **then**
  **Split.**
    Compute the normalized joint splitting matrix $\tilde{\mathcal{S}}_{\phi}^{(i)}$.
    Compute its minimum eigenpair for each Gaussian.
    Select split candidates $\mathcal{I}_{\text{split}}$ using Eq. (8).
    Split each $i \in \mathcal{I}_{\text{split}}$ along the joint position–scale direction using Eq. (9).
  **Clone.**
    Compute the duplicate strength $\eta_{\text{dup}}^{(i)} = -g_{\text{dup}}^{(i)}/h_{\text{dup}}^{(i)}$.
    Select clone candidates $\mathcal{I}_{\text{clone}}$ using Eq. (14).
    Append duplicate children $\mathcal{C}(\theta; \eta_{\text{dup}}^{(i)})$ for all $i \in \mathcal{I}_{\text{clone}}$.
  **Opacity decay.**
    Compute the opacity decay strength $\eta_{\text{dec}}^{(i)} = g_{\text{dec}}^{(i)}/h_{\text{dec}}^{(i)}$.
    Select opacity decay candidates $\mathcal{I}_{\text{decay}}$ using Eq. (18).
    Apply $\mathcal{P}(\theta; \eta_{\text{dec}}^{(i)})$ for all $i \in \mathcal{I}_{\text{decay}}$.
    Remove low-opacity Gaussians and reset all density-control accumulators.
  **end if**
**end for**
**return** Optimized Gaussians $\theta$.

We first consider the effect of splitting a single Gaussian $i$, while all other Gaussians are fixed. Define the first-order term

$$g_{\phi}^{(i)} = \mathbb{E}_{(\varphi,x)}\left[\frac{\partial \mathcal{L}_{\varphi,x}}{\partial \alpha^{(i)}(x)} \nabla_{\phi^{(i)}} \alpha^{(i)}(x; \phi^{(i)})\right]. \tag{23}$$

By the chain rule, the Hessian of the loss with respect to $\phi^{(i)}$ can be decomposed as

$$\nabla^2_{\phi^{(i)}} \mathcal{L}(\Phi) = \mathcal{T}_{\phi}^{(i)}(\theta) + \mathcal{S}_{\phi}^{(i)}(\theta), \tag{24}$$

where

$$\mathcal{T}_{\phi}^{(i)}(\theta) = \mathbb{E}_{(\varphi,x)}\left[\frac{\partial^2 \mathcal{L}_{\varphi,x}}{\partial \alpha^{(i)}(x)^2} \nabla_{\phi^{(i)}} \alpha^{(i)}(x; \phi^{(i)}) \nabla_{\phi^{(i)}} \alpha^{(i)}(x; \phi^{(i)})^{\top}\right], \tag{25}$$

and

$$\mathcal{S}_{\phi}^{(i)}(\theta) = \mathbb{E}_{(\varphi,x)}\left[\frac{\partial \mathcal{L}_{\varphi,x}}{\partial \alpha^{(i)}(x)} \nabla^2_{\phi^{(i)}} \alpha^{(i)}(x; \phi^{(i)})\right]. \tag{26}$$

The matrix $\mathcal{T}_{\phi}^{(i)}$ comes from the curvature of the loss with respect to the Gaussian contribution, whereas $\mathcal{S}_{\phi}^{(i)}$ comes from the curvature of the Gaussian contribution itself. Although both terms appear in the Hessian of the ordinary parameter update, they play different roles under splitting.

Table 8: Effect of operation priority, averaged over Bicycle, Truck, and Dr. Johnson. S, C, and D denote splitting, cloning, and opacity decay, respectively.

| Priority order | PSNR↑ | SSIM↑ | LPIPS↓ | Num |
|---|---|---|---|---|
| C→D→S | 26.92 | 0.854 | 0.201 | 1.9 |
| C→S→D | 26.84 | 0.854 | 0.201 | 1.9 |
| D→C→S | 26.69 | 0.853 | 0.204 | 1.9 |
| D→S→C | 26.86 | 0.854 | 0.200 | 2.1 |
| S→D→C | 26.76 | 0.853 | 0.201 | 2.1 |
| S→C→D (Ours) | **26.93** | **0.854** | **0.200** | 2.0 |

Table 9: Comparison of adaptive arbitration and fixed priority on the same three scenes as in Tab. 8.

| Arbitration strategy | PSNR↑ | SSIM↑ | LPIPS↓ | Num |
|---|---|---|---|---|
| Adaptive loss-scale arbitration | 26.87 | 0.853 | 0.203 | 2.0 |
| Fixed S→C→D (Ours) | **26.93** | **0.854** | **0.200** | 2.0 |

Using a Taylor expansion around $\varepsilon = 0$, the loss after splitting Gaussian $i$ becomes

$$\begin{aligned}\mathcal{L}_i(\varepsilon) = \mathcal{L}(\Phi) &+ \varepsilon g_\phi^{(i)\top} \sum_{j=1}^{m_i} w_j^{(i)} \big(\mu^{(i)} + \delta_j^{(i)}\big) \\ &+ \frac{\varepsilon^2}{2} \left( \sum_{j=1}^{m_i} w_j^{(i)} \big(\mu^{(i)} + \delta_j^{(i)}\big) \right)^\top \mathcal{T}_\phi^{(i)} \left( \sum_{j=1}^{m_i} w_j^{(i)} \big(\mu^{(i)} + \delta_j^{(i)}\big) \right) \\ &+ \frac{\varepsilon^2}{2} \sum_{j=1}^{m_i} w_j^{(i)} \big(\mu^{(i)} + \delta_j^{(i)}\big)^\top \mathcal{S}_\phi^{(i)} \big(\mu^{(i)} + \delta_j^{(i)}\big) + O(\varepsilon^3). \end{aligned} \tag{27}$$

By the zero-mean constraint $\sum_j w_j^{(i)} \delta_j^{(i)} = 0$, the first-order term reduces to $g_\phi^{(i)\top} \mu^{(i)}$, and the $\mathcal{T}_\phi^{(i)}$ term depends only on the mean shift $\mu^{(i)}$. In contrast, the $\mathcal{S}_\phi^{(i)}$ term separates as

$$\sum_{j=1}^{m_i} w_j^{(i)} \big(\mu^{(i)} + \delta_j^{(i)}\big)^\top \mathcal{S}_\phi^{(i)} \big(\mu^{(i)} + \delta_j^{(i)}\big) = \mu^{(i)\top} \mathcal{S}_\phi^{(i)} \mu^{(i)} + \sum_{j=1}^{m_i} w_j^{(i)} \delta_j^{(i)\top} \mathcal{S}_\phi^{(i)} \delta_j^{(i)}. \tag{28}$$

Therefore,

$$\mathcal{L}_i(\varepsilon) = \mathcal{L}(\Phi) + \varepsilon g_\phi^{(i)\top} \mu^{(i)} + \frac{\varepsilon^2}{2} \mu^{(i)\top} \nabla^2_{\phi^{(i)}} \mathcal{L}(\Phi) \mu^{(i)} + \frac{\varepsilon^2}{2} \sum_{j=1}^{m_i} w_j^{(i)} \delta_j^{(i)\top} \mathcal{S}_\phi^{(i)}(\theta) \delta_j^{(i)} + O(\varepsilon^3). \tag{29}$$

The first three terms describe the effect of moving the weighted average of the offspring group. This behavior is equivalent to moving the original Gaussian and can be handled by ordinary gradient-based optimization. The last term depends only on the zero-mean offsets of the offspring and therefore captures the intrinsic effect of splitting. This motivates the splitting characteristic function

$$\Delta^{(i)}_{\text{split}}\big(\{\delta_j^{(i)}\}_{j=1}^{m_i}, w^{(i)}\big) = \frac{1}{2} \sum_{j=1}^{m_i} w_j^{(i)} \delta_j^{(i)\top} \mathcal{S}_\phi^{(i)}(\theta) \delta_j^{(i)}. \tag{30}$$

For this reason, $\mathcal{S}_\phi^{(i)}(\theta)$ is called the splitting matrix.

Applying the same expansion to all Gaussians and concatenating the mean shifts as

$$\mu = \begin{bmatrix} \mu^{(1)\top} & \cdots & \mu^{(n)\top} \end{bmatrix}^\top,$$

we obtain the SteepGS loss decomposition:

$$\mathcal{L}(\Phi_{\text{child}}, w) = \mathcal{L}(\Phi) + \varepsilon \nabla_\Phi \mathcal{L}(\Phi)^\top \mu + \frac{\varepsilon^2}{2} \mu^\top \nabla^2_\Phi \mathcal{L}(\Phi) \mu + \frac{\varepsilon^2}{2} \sum_{i=1}^{n} \sum_{j=1}^{m_i} w_j^{(i)} \delta_j^{(i)\top} \mathcal{S}_\phi^{(i)}(\theta) \delta_j^{(i)} + O(\varepsilon^3). \tag{31}$$

This decomposition shows that splitting can locally reduce the objective only when the splitting matrix has a negative-curvature direction. In particular, if

$$\lambda_{\min}^{(i)} = \lambda_{\min}\big(\mathcal{S}_{\phi}^{(i)}(\theta)\big) < 0, \tag{32}$$

then choosing the offspring offsets along the minimum-eigenvalue eigenvector yields a negative value of $\Delta_{\text{split}}^{(i)}$. Under the constraint $\|\delta_j^{(i)}\|_2 \le 1$, the SteepGS solution is

$$m_i = 2, \qquad w_1^{(i)} = w_2^{(i)} = \frac{1}{2}, \qquad \delta_1^{(i)\star} = v_{\min}^{(i)}, \qquad \delta_2^{(i)\star} = -v_{\min}^{(i)}, \tag{33}$$

where $v_{\min}^{(i)}$ is the eigenvector associated with $\lambda_{\min}^{(i)}$. Thus, SteepGS splits a Gaussian only when its splitting matrix is not positive semidefinite, places the two offspring in opposite directions along the steepest negative-curvature direction, and assigns half of the parent opacity to each offspring.

### A.4 Normalized Joint Position-Scale Splitting Matrix

**Joint position–scale splitting matrix.** Following the second-order splitting view [32], the local effect of splitting a Gaussian is characterized by the loss-weighted curvature of its projected contribution. We extend the local split variable from the position alone to the joint position–log-scale parameter

$$\phi^{(i)} = \begin{bmatrix} p^{(i)} \\ \tilde{s}^{(i)} \end{bmatrix} \in \mathbb{R}^6, \qquad \tilde{s}^{(i)} = \log s^{(i)}, \tag{34}$$

where $p^{(i)} \in \mathbb{R}^3$ is the Gaussian center and $\tilde{s}^{(i)} \in \mathbb{R}^3$ is the log-scale parameter.

For a camera–pixel pair $(\varphi, x)$, we introduce the screen-space variable

$$z^{(i)}(\phi^{(i)};\varphi) = \begin{bmatrix} u^{(i)}(\phi^{(i)};\varphi) \\ q^{(i)}(\phi^{(i)};\varphi) \end{bmatrix} = \Big[u_x^{(i)}, u_y^{(i)}, q_{xx}^{(i)}, q_{xy}^{(i)}, q_{yy}^{(i)}\Big]^\top \in \mathbb{R}^5. \tag{35}$$

Here, $u^{(i)} = \varphi(p^{(i)}) \in \mathbb{R}^2$ is the projected mean, and $q^{(i)} = (q_{xx}^{(i)}, q_{xy}^{(i)}, q_{yy}^{(i)}) \in \mathbb{R}^3$ parameterizes the conic matrix of the inverse projected covariance,

$$Q(q^{(i)}) = \begin{bmatrix} q_{xx}^{(i)} & q_{xy}^{(i)} \\ q_{xy}^{(i)} & q_{yy}^{(i)} \end{bmatrix}. \tag{36}$$

Let $d^{(i)} = u^{(i)} - x$. The projected Gaussian contribution is written as

$$\alpha^{(i)}(x;\phi^{(i)}) = o^{(i)}\, G^{(i)}(z^{(i)}), \qquad G^{(i)}(z^{(i)}) = \exp\left(-\frac{1}{2} d^{(i)\top} Q(q^{(i)}) d^{(i)}\right), \tag{37}$$

where $o^{(i)}$ is the opacity. Since $o^{(i)}$ is fixed when computing the splitting direction, it is absorbed into the derivatives of $\alpha^{(i)}$.

The screen-space variable $z^{(i)}$ is useful because rasterization naturally provides quantities associated with the projected mean $u^{(i)}$ and conic $q^{(i)}$. For each sampled camera–pixel pair, define

$$g_z^{(i)}(\varphi, x) = \frac{\partial \mathcal{L}_{\varphi,x}}{\partial \alpha^{(i)}(x)} \nabla_{z^{(i)}} \alpha^{(i)}(x;\phi^{(i)}), \qquad \mathcal{H}_z^{(i)}(\varphi, x) = \frac{\partial \mathcal{L}_{\varphi,x}}{\partial \alpha^{(i)}(x)} \nabla_{z^{(i)}}^2 \alpha^{(i)}(x;\phi^{(i)}). \tag{38}$$

These are the loss-weighted gradient and Hessian of the projected Gaussian contribution with respect to the intermediate screen-space variable $z^{(i)}$.

The Jacobian of $z^{(i)}$ with respect to the joint variable $\phi^{(i)}$ is

$$\begin{aligned} J_{up}^{(i)} = \frac{\partial u^{(i)}}{\partial p^{(i)}}, \qquad J_{qp}^{(i)} = \frac{\partial q^{(i)}}{\partial p^{(i)}}, \qquad J_{q\tilde{s}}^{(i)} = \frac{\partial q^{(i)}}{\partial \tilde{s}^{(i)}}, \\ \mathcal{J}_z^{(i)}(\varphi) = \frac{\partial z^{(i)}}{\partial \phi^{(i)}} = \begin{bmatrix} J_{up}^{(i)} & 0 \\ J_{qp}^{(i)} & J_{q\tilde{s}}^{(i)} \end{bmatrix}. \end{aligned} \tag{39}$$

The zero block appears because the projected mean $u^{(i)}$ depends on the Gaussian position but not directly on its scale. Then, applying the second-order chain rule to $\alpha^{(i)}(x;\phi^{(i)}) = \alpha^{(i)}(z^{(i)}(\phi^{(i)};\varphi))$ gives

$$\nabla^2_{\phi^{(i)}}\alpha^{(i)} = \mathcal{J}_z^{(i)\top}\nabla^2_{z^{(i)}}\alpha^{(i)}\mathcal{J}_z^{(i)} + \sum_{m=1}^{5}\frac{\partial\alpha^{(i)}}{\partial z_m^{(i)}}\nabla^2_{\phi^{(i)}}z_m^{(i)}. \tag{40}$$

Multiplying by $\partial\mathcal{L}_{\varphi,x}/\partial\alpha^{(i)}(x)$ and taking the expectation over sampled camera–pixel pairs yields the joint splitting matrix

$$\mathcal{S}_\phi^{(i)}(\theta) = \mathbb{E}_{(\varphi,x)}\left[\mathcal{J}_z^{(i)}(\varphi)^\top\mathcal{H}_z^{(i)}(\varphi,x)\mathcal{J}_z^{(i)}(\varphi) + \sum_{m=1}^{5} g_{z_m}^{(i)}(\varphi,x)\nabla^2_{\phi^{(i)}}z_m^{(i)}(\phi^{(i)};\varphi)\right]. \tag{41}$$

This expression is the direct joint-variable counterpart of the SteepGS splitting matrix. The first term pulls back the screen-space loss-weighted Hessian to the joint position–scale space, while the second term accounts for the nonlinear dependence of the screen-space variables $z^{(i)}$ on $\phi^{(i)}$.

In block form,

$$\mathcal{S}_\phi^{(i)} = \begin{bmatrix}\mathcal{S}_{pp}^{(i)} & \mathcal{S}_{p\tilde{s}}^{(i)}\\ \mathcal{S}_{\tilde{s}p}^{(i)} & \mathcal{S}_{\tilde{s}\tilde{s}}^{(i)}\end{bmatrix}. \tag{42}$$

For a single camera–pixel pair, the corresponding position–position block is

$$\begin{aligned}\mathcal{S}_{pp}^{(i)} =& J_{up}^{(i)\top}\mathcal{H}_{uu}^{(i)}J_{up}^{(i)} + J_{up}^{(i)\top}\mathcal{H}_{uq}^{(i)}J_{qp}^{(i)} + J_{qp}^{(i)\top}\mathcal{H}_{qu}^{(i)}J_{up}^{(i)} + J_{qp}^{(i)\top}\mathcal{H}_{qq}^{(i)}J_{qp}^{(i)}\\ &+\sum_{k=1}^{2} g_{u_k}^{(i)}\nabla^2_{p^{(i)}}u_k^{(i)} + \sum_{r=1}^{3} g_{q_r}^{(i)}\nabla^2_{p^{(i)}}q_r^{(i)}.\end{aligned} \tag{43}$$

The final $\mathcal{S}_{pp}^{(i)}$ is obtained by averaging this quantity over sampled camera–pixel pairs. The two summations are the second-order chain-rule corrections: the first accounts for the curvature of the perspective projection $u^{(i)} = \varphi(p^{(i)})$, and the second accounts for the curvature of the conic parameters with respect to the 3D position.

Similarly, the position–scale block for a single sample is

$$\mathcal{S}_{p\tilde{s}}^{(i)} = J_{up}^{(i)\top}\mathcal{H}_{uq}^{(i)}J_{q\tilde{s}}^{(i)} + J_{qp}^{(i)\top}\mathcal{H}_{qq}^{(i)}J_{q\tilde{s}}^{(i)} + \sum_{r=1}^{3} g_{q_r}^{(i)}\nabla^2_{p^{(i)}\tilde{s}^{(i)}}q_r^{(i)}. \tag{44}$$

In our implementation, we use the more stable first-order pullback form and omit the mixed second-order conic curvature term:

$$\mathcal{S}_{p\tilde{s}}^{(i)} \approx J_{up}^{(i)\top}\mathcal{H}_{uq}^{(i)}J_{q\tilde{s}}^{(i)} + J_{qp}^{(i)\top}\mathcal{H}_{qq}^{(i)}J_{q\tilde{s}}^{(i)}, \qquad \mathcal{S}_{\tilde{s}p}^{(i)} = \mathcal{S}_{p\tilde{s}}^{(i)\top}. \tag{45}$$

The scale–scale block is

$$\mathcal{S}_{\tilde{s}\tilde{s}}^{(i)} = J_{q\tilde{s}}^{(i)\top}\mathcal{H}_{qq}^{(i)}J_{q\tilde{s}}^{(i)} + \sum_{r=1}^{3} g_{q_r}^{(i)}\nabla^2_{\tilde{s}^{(i)}}q_r^{(i)}. \tag{46}$$

The off-diagonal block $\mathcal{S}_{p\tilde{s}}^{(i)}$ distinguishes the joint formulation from a position-only splitting matrix. The term $J_{up}^{(i)\top}\mathcal{H}_{uq}^{(i)}J_{q\tilde{s}}^{(i)}$ captures the interaction between the projected mean change induced by position and the conic change induced by scale. The term $J_{qp}^{(i)\top}\mathcal{H}_{qq}^{(i)}J_{q\tilde{s}}^{(i)}$ captures the interaction between position-induced and scale-induced conic changes. Therefore, the minimum-eigenvalue direction of $\mathcal{S}_\phi^{(i)}$ estimates a splitting direction that jointly accounts for the Gaussian center and its anisotropic spatial extent.

**Normalized joint coordinate.** The matrix $\mathcal{S}_\phi^{(i)}$ is expressed in the raw position–log-scale parameter space. However, the two blocks of $\phi^{(i)} = [p^{(i)\top}, \tilde{s}^{(i)\top}]^\top$ have different physical units and induce different local changes in the Gaussian. Directly computing eigenvectors in this raw coordinate

system can therefore make the joint split direction sensitive to the relative parameterization of position and scale.

To compare position and log-scale perturbations in a common coordinate system, we use a local Gaussian-shape metric. Let

$$\Sigma^{(i)} = L^{(i)} L^{(i)\top}$$

be the covariance of the parent Gaussian, where $L^{(i)} = R^{(i)} \operatorname{diag}(s^{(i)})$ is the covariance factor aligned with the principal scale axes. We measure infinitesimal perturbations of the center and log-scale by

$$d^2_{\mathcal{G}} = \delta p^{(i)\top} \Sigma^{(i)-1} \delta p^{(i)} + 2\|\delta \tilde{s}^{(i)}\|_2^2. \tag{47}$$

The first term is the Mahalanobis norm of the position displacement, so a unit position perturbation corresponds to moving by one local Gaussian standard deviation. The second term is a Fisher-inspired normalization for log-scale perturbations of the underlying normalized Gaussian shape.

To see the origin of the factor 2, consider the one-dimensional normalized Gaussian density

$$q_{\mathcal{N}}(y; \mu, \tilde{s}) = \frac{1}{\sqrt{2\pi}\exp(\tilde{s})} \exp\left(-\frac{1}{2}\left(\frac{y-\mu}{\exp(\tilde{s})}\right)^2\right),$$

and let $z = (y-\mu)/\exp(\tilde{s})$. Then

$$\frac{\partial \log q_{\mathcal{N}}}{\partial \tilde{s}} = z^2 - 1.$$

Since $z \sim \mathcal{N}(0,1)$, $\mathbb{E}[z^2] = 1$ and $\mathbb{E}[z^4] = 3$, giving

$$\mathbb{E}\left[\left(\frac{\partial \log q_{\mathcal{N}}}{\partial \tilde{s}}\right)^2\right] = \mathbb{E}[(z^2-1)^2] = 2. \tag{48}$$

Applying this independently along the three principal scale axes gives the log-scale part of Eq. (47).

We emphasize that this metric is used as a coordinate normalization for the underlying Gaussian shape. The rendered 3DGS opacity contribution is an unnormalized Gaussian kernel, so Eq. (47) should not be interpreted as the exact Fisher metric of the rendered opacity contribution. Instead, it provides a dimensionless local coordinate system in which position and log-scale perturbations are compared on a common scale.

We introduce a normalized joint direction $v^{(i)} = [v_p^{(i)\top}, v_{\tilde{s}}^{(i)\top}]^\top \in \mathbb{R}^6$ and map it to a raw position-log-scale perturbation by

$$\delta\phi^{(i)} = \begin{bmatrix} \delta p^{(i)} \\ \delta \tilde{s}^{(i)} \end{bmatrix} = J^{(i)} v^{(i)} = \begin{bmatrix} L^{(i)} & 0 \\ 0 & \frac{1}{\sqrt{2}}\mathbf{I} \end{bmatrix} \begin{bmatrix} v_p^{(i)} \\ v_{\tilde{s}}^{(i)} \end{bmatrix} = \begin{bmatrix} L^{(i)} v_p^{(i)} \\ \frac{1}{\sqrt{2}} v_{\tilde{s}}^{(i)} \end{bmatrix}. \tag{49}$$

Substituting these into Eq. (47) gives

$$d^2_{\mathcal{G}} = \|v_p^{(i)}\|_2^2 + \|v_{\tilde{s}}^{(i)}\|_2^2 = \|v^{(i)}\|_2^2. \tag{50}$$

Thus, $v^{(i)}$ is a dimensionless coordinate in which one unit of position perturbation and one unit of log-scale perturbation are measured using the same local Gaussian metric.

The raw second-order splitting surrogate for a perturbation $\delta\phi^{(i)}$ is

$$\delta\phi^{(i)\top} \bar{\mathcal{S}}_\phi^{(i)} \delta\phi^{(i)}.$$

Here, $\bar{\mathcal{S}}_\phi^{(i)}$ denotes the symmetrized raw joint splitting matrix used for eigendecomposition. Using $\delta\phi^{(i)} = J^{(i)} v^{(i)}$, this becomes

$$\delta\phi^{(i)\top} \bar{\mathcal{S}}_\phi^{(i)} \delta\phi^{(i)} = v^{(i)\top} \left(J^{(i)\top} \bar{\mathcal{S}}_\phi^{(i)} J^{(i)}\right) v^{(i)}. \tag{51}$$

Therefore, the splitting matrix in the normalized joint coordinate system is

$$\tilde{\mathcal{S}}_\phi^{(i)} = J^{(i)\top} \bar{\mathcal{S}}_\phi^{(i)} J^{(i)}. \tag{52}$$

In practice, we symmetrize $\mathcal{S}_\phi^{(i)}$ before eigendecomposition to remove numerical asymmetry from accumulation. We then compute the split direction as the eigenvector associated with the smallest eigenvalue of $\tilde{\mathcal{S}}_\phi^{(i)}$, and map this normalized direction back to the raw position-log-scale perturbation using Eq. (49).

### A.5 Proxy Curvature for Local Surrogates

This section details the image-space curvature weight used in the clone and opacity decay surrogates. In Sec. 4.3 and Sec. 4.4, both operations are modeled by a one-dimensional quadratic approximation with respect to the operation strength. To define these surrogates, we need a local curvature term that describes how the photometric loss $\mathcal{L}$ changes under an image-space perturbation. However, directly computing the exact curvature of $\mathcal{L}$ is challenging because $\mathcal{L}$ combines an $\mathcal{L}_1$ term and an SSIM [33] term. The $\mathcal{L}_1$ component is non-smooth, while the SSIM component depends on local-window statistics such as means, variances, and covariances, which couple neighboring pixels. Therefore, instead of computing the full Hessian of $\mathcal{L}$, we construct a diagonal image-space curvature proxy that is positive, stable, and inexpensive to accumulate during rasterization.

We first approximate the $\mathcal{L}_1$ curvature using a Charbonnier robust penalty [4, 2]. Let the residual $r_c(x)$ at pixel $x$ and color channel $c$ be

$$r_c(x) = I_c(x) - \hat{I}_c(x), \tag{53}$$

where $I_c(x)$ and $\hat{I}_c(x)$ denote the rendered and target color values, respectively, at pixel $x$ in the $c$-th color channel. The Charbonnier penalty is

$$\rho_\epsilon(r_c(x)) = \sqrt{r_c(x)^2 + \epsilon^2} - \epsilon. \tag{54}$$

where $\epsilon$ is a small smoothing parameter controlling the transition between quadratic behavior near zero residuals and $\mathcal{L}_1$-like behavior for large residuals. We set $\epsilon = 10^{-3}$ in all experiments. We use the curvature induced by this smooth penalty as the Charbonnier curvature weight:

$$\omega_{\mathrm{charb},c}(x) = \rho''_\epsilon(r_c(x)) = \frac{\epsilon^2}{\left(r_c(x)^2 + \epsilon^2\right)^{3/2}}. \tag{55}$$

This gives a smooth diagonal approximation to the $\mathcal{L}_1$ curvature. To also account for the SSIM component, we follow recent 3DGS second-order optimizers that use diagonal approximations to handle its local pixel coupling efficiently [9, 29], and introduce a diagonal SSIM curvature proxy. To approximate the SSIM curvature, we follow the standard 3DGS SSIM setting and use an $11 \times 11$ Gaussian window with standard deviation $1.5$. Let $G$ denote this normalized window, and let $*$ denote convolution over the image plane. For each color channel, we compute target-window statistics:

$$\mu_c(x) = (G * \hat{I}_c)(x), \qquad \nu_c(x) = (G * \hat{I}_c^2)(x) - \mu_c(x)^2, \tag{56}$$

where $\mu_c(x)$ and $\nu_c(x)$ denote the local target mean and variance. We then form the inverse luminance and contrast denominators

$$D^{-1}_{\mu,c}(x) = \frac{1}{2\mu_c(x)^2 + C_1 + \epsilon_{\mathrm{ssim}}}, \qquad D^{-1}_{\nu,c}(x) = \frac{1}{2\nu_c(x) + C_2 + \epsilon_{\mathrm{ssim}}}. \tag{57}$$

Using these local denominators, we approximate the diagonal curvature induced by the SSIM term as

$$\omega_{\mathrm{ssim},c}(x) = 2\,(G^2 * D^{-1}_{\mu,c})(x) + 2\left[(G * D^{-1}_{\nu,c})(x) - (G^2 * D^{-1}_{\nu,c})(x)\right], \tag{58}$$

where $G^2$ denotes the element-wise squared Gaussian window. This expression keeps only the diagonal response of the local-window SSIM approximation and drops cross-pixel Hessian terms, making it efficient to evaluate as separable convolutions.

Finally, we combine the Charbonnier and SSIM curvature proxies according to the same weighting used in $\mathcal{L}$:

$$\omega_c(x) = (1 - \lambda_{\text{D-SSIM}})\omega_{\mathrm{charb},c}(x) + \lambda_{\text{D-SSIM}}\gamma_{\mathrm{ssim}}\omega_{\mathrm{ssim},c}(x). \tag{59}$$

where $\gamma_{\mathrm{ssim}}$ controls the scale of the SSIM curvature proxy. We set $\gamma_{\mathrm{ssim}} = 10^{-4}$ empirically to balance the magnitude of the SSIM proxy with the Charbonnier curvature. This mixed curvature weight is used as the proxy curvature in the clone and opacity decay surrogates. By combining the residual-based Charbonnier term with the diagonal SSIM proxy, it provides a lightweight and effective second-order approximation of the photometric loss $\mathcal{L}$ under local image-space perturbations. In practice, $\omega_c(x)$ is computed once from the current rendered image and target image, and then reused for all Gaussian candidates contributing to the same pixel.

Table 10: PSNR scores for Mip-NeRF360, Tanks&Temples, and Deep Blending scenes.

| | Mip-NeRF360 | | | | | | | | | Tanks&Temples | | Deep Blending | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | bicycle | flowers | garden | stump | treehill | room | counter | kitchen | bonsai | train | truck | drjohnson | playroom |
| 3DGS | 25.19 | 21.51 | 27.31 | 26.58 | 22.31 | 31.38 | 28.93 | 30.84 | 32.05 | 21.79 | 24.99 | 28.98 | 30.07 |
| AbsGS | $25.27_{\pm 0.02}$ | $21.26_{\pm 0.03}$ | $27.57_{\pm 0.02}$ | $26.59_{\pm 0.10}$ | $21.93_{\pm 0.04}$ | $31.60_{\pm 0.12}$ | $29.12_{\pm 0.06}$ | $31.63_{\pm 0.25}$ | $32.26_{\pm 0.16}$ | $21.68_{\pm 0.12}$ | $25.65_{\pm 0.04}$ | $29.16_{\pm 0.04}$ | $30.08_{\pm 0.06}$ |
| PixelGS | $25.30_{\pm 0.03}$ | $21.56_{\pm 0.02}$ | $27.48_{\pm 0.04}$ | $26.82_{\pm 0.04}$ | $22.26_{\pm 0.03}$ | $31.40_{\pm 0.05}$ | $29.16_{\pm 0.02}$ | $31.74_{\pm 0.07}$ | $32.45_{\pm 0.07}$ | $22.02_{\pm 0.07}$ | $25.53_{\pm 0.01}$ | $28.07_{\pm 0.03}$ | $29.79_{\pm 0.04}$ |
| DC4GS | $25.48_{\pm 0.04}$ | $21.55_{\pm 0.02}$ | $27.62_{\pm 0.14}$ | $26.89_{\pm 0.01}$ | $22.13_{\pm 0.02}$ | $31.78_{\pm 0.07}$ | $28.99_{\pm 0.09}$ | $31.54_{\pm 0.20}$ | $32.30_{\pm 0.01}$ | $22.07_{\pm 0.07}$ | $25.82_{\pm 0.02}$ | $29.17_{\pm 0.02}$ | $30.12_{\pm 0.05}$ |
| GDAGS | $25.34_{\pm 0.03}$ | $21.25_{\pm 0.08}$ | $27.58_{\pm 0.03}$ | $26.78_{\pm 0.05}$ | $21.98_{\pm 0.03}$ | $31.54_{\pm 0.06}$ | $29.07_{\pm 0.03}$ | $31.76_{\pm 0.08}$ | $32.20_{\pm 0.10}$ | $21.67_{\pm 0.11}$ | $25.64_{\pm 0.02}$ | $29.22_{\pm 0.04}$ | $30.00_{\pm 0.13}$ |
| ResGS | $25.28_{\pm 0.02}$ | $21.66_{\pm 0.04}$ | $27.59_{\pm 0.01}$ | $26.67_{\pm 0.03}$ | $22.77_{\pm 0.06}$ | $31.85_{\pm 0.08}$ | $29.02_{\pm 0.02}$ | $31.69_{\pm 0.06}$ | $32.34_{\pm 0.04}$ | $22.23_{\pm 0.08}$ | $25.81_{\pm 0.03}$ | $29.21_{\pm 0.20}$ | $29.87_{\pm 0.31}$ |
| SteepGS | $24.81_{\pm 0.01}$ | $20.72_{\pm 0.03}$ | $27.14_{\pm 0.02}$ | $25.87_{\pm 0.05}$ | $22.18_{\pm 0.09}$ | $31.35_{\pm 0.09}$ | $28.73_{\pm 0.02}$ | $30.86_{\pm 0.17}$ | $31.80_{\pm 0.03}$ | $21.74_{\pm 0.09}$ | $25.22_{\pm 0.01}$ | $29.29_{\pm 0.05}$ | $29.99_{\pm 0.06}$ |
| Ours | $25.31_{\pm 0.03}$ | $21.72_{\pm 0.01}$ | $27.59_{\pm 0.07}$ | $26.62_{\pm 0.02}$ | $22.83_{\pm 0.06}$ | $32.07_{\pm 0.08}$ | $29.13_{\pm 0.01}$ | $31.78_{\pm 0.02}$ | $32.46_{\pm 0.03}$ | $22.44_{\pm 0.03}$ | $25.96_{\pm 0.02}$ | $29.44_{\pm 0.06}$ | $30.14_{\pm 0.14}$ |

Table 11: SSIM scores for Mip-NeRF360, Tanks&Temples, and Deep Blending scenes.

| | Mip-NeRF360 | | | | | | | | | Tanks&Temples | | Deep Blending | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | bicycle | flowers | garden | stump | treehill | room | counter | kitchen | bonsai | train | truck | drjohnson | playroom |
| 3DGS | 0.764 | 0.605 | 0.866 | 0.771 | 0.632 | 0.918 | 0.907 | 0.925 | 0.940 | 0.808 | 0.876 | 0.899 | 0.907 |
| AbsGS | $0.782_{\pm 0.00}$ | $0.621_{\pm 0.00}$ | $0.872_{\pm 0.00}$ | $0.777_{\pm 0.00}$ | $0.614_{\pm 0.00}$ | $0.925_{\pm 0.00}$ | $0.912_{\pm 0.00}$ | $0.928_{\pm 0.00}$ | $0.945_{\pm 0.00}$ | $0.817_{\pm 0.00}$ | $0.887_{\pm 0.00}$ | $0.896_{\pm 0.00}$ | $0.906_{\pm 0.00}$ |
| PixelGS | $0.779_{\pm 0.00}$ | $0.636_{\pm 0.00}$ | $0.871_{\pm 0.00}$ | $0.785_{\pm 0.00}$ | $0.634_{\pm 0.00}$ | $0.922_{\pm 0.00}$ | $0.915_{\pm 0.00}$ | $0.931_{\pm 0.00}$ | $0.946_{\pm 0.00}$ | $0.826_{\pm 0.00}$ | $0.887_{\pm 0.00}$ | $0.888_{\pm 0.00}$ | $0.905_{\pm 0.00}$ |
| DC4GS | $0.792_{\pm 0.00}$ | $0.635_{\pm 0.00}$ | $0.874_{\pm 0.00}$ | $0.793_{\pm 0.00}$ | $0.631_{\pm 0.00}$ | $0.925_{\pm 0.00}$ | $0.911_{\pm 0.00}$ | $0.929_{\pm 0.00}$ | $0.945_{\pm 0.00}$ | $0.825_{\pm 0.00}$ | $0.889_{\pm 0.00}$ | $0.902_{\pm 0.00}$ | $0.910_{\pm 0.00}$ |
| GDAGS | $0.786_{\pm 0.00}$ | $0.626_{\pm 0.00}$ | $0.872_{\pm 0.00}$ | $0.785_{\pm 0.00}$ | $0.622_{\pm 0.00}$ | $0.924_{\pm 0.00}$ | $0.911_{\pm 0.00}$ | $0.930_{\pm 0.00}$ | $0.944_{\pm 0.00}$ | $0.818_{\pm 0.00}$ | $0.886_{\pm 0.00}$ | $0.900_{\pm 0.00}$ | $0.907_{\pm 0.00}$ |
| ResGS | $0.767_{\pm 0.00}$ | $0.606_{\pm 0.00}$ | $0.867_{\pm 0.00}$ | $0.774_{\pm 0.00}$ | $0.638_{\pm 0.00}$ | $0.919_{\pm 0.00}$ | $0.908_{\pm 0.00}$ | $0.928_{\pm 0.00}$ | $0.942_{\pm 0.00}$ | $0.816_{\pm 0.00}$ | $0.885_{\pm 0.00}$ | $0.902_{\pm 0.00}$ | $0.905_{\pm 0.00}$ |
| SteepGS | $0.731_{\pm 0.00}$ | $0.547_{\pm 0.00}$ | $0.856_{\pm 0.00}$ | $0.733_{\pm 0.00}$ | $0.612_{\pm 0.00}$ | $0.916_{\pm 0.00}$ | $0.901_{\pm 0.00}$ | $0.923_{\pm 0.00}$ | $0.938_{\pm 0.00}$ | $0.804_{\pm 0.00}$ | $0.876_{\pm 0.00}$ | $0.902_{\pm 0.00}$ | $0.907_{\pm 0.00}$ |
| Ours | $0.770_{\pm 0.00}$ | $0.607_{\pm 0.00}$ | $0.868_{\pm 0.00}$ | $0.776_{\pm 0.00}$ | $0.637_{\pm 0.00}$ | $0.921_{\pm 0.00}$ | $0.909_{\pm 0.00}$ | $0.927_{\pm 0.00}$ | $0.943_{\pm 0.00}$ | $0.818_{\pm 0.00}$ | $0.889_{\pm 0.00}$ | $0.903_{\pm 0.00}$ | $0.908_{\pm 0.00}$ |

# B More Experiment Results

## B.1 All scene results

We report the results for all scenes included in the main table in Tab. 10, 11, and 12. All results are computed based on the averages obtained from five different runs.

## B.2 Additional Ablation Study

We provide additional ablation results on Tanks&Temples and Deep Blending in Tab. 13, using the same component configurations as in Tab. 2.

## B.3 Matched-Budget Comparison

We compare our method with 3DGS [12] and DC4GS [11] under comparable Gaussian budgets, using the same scene setting as in the main experiments. For DC4GS, we adjust the number of densification events to obtain a final Gaussian count comparable to ours, while keeping its other training settings unchanged. As shown in Tab. 15, our method achieves higher PSNR and SSIM than DC4GS across all three benchmarks, whereas DC4GS obtains lower LPIPS.

We additionally compare our method with 3DGS and 3DGS-MCMC [14], with the Gaussian counts of the baselines matched to those of our method. For Mip-NeRF360, we follow the scene selection of 3DGS-MCMC and evaluate all compared methods on the seven publicly available scenes. As shown in Tab. 14, our method achieves the best PSNR, SSIM, and LPIPS on Tanks&Temples and Deep Blending. On the seven-scene Mip-NeRF360 subset, our method achieves comparable PSNR to 3DGS-MCMC, while 3DGS-MCMC obtains higher SSIM and lower LPIPS.

## B.4 Convergence Analysis

We compare the convergence behavior of our method with 3DGS [12] by tracking PSNR, LPIPS, and the number of Gaussians over 30K training iterations. Following the representative-scene evaluation setting of SAD-GS [22], we use Bicycle from Mip-NeRF360, Truck from Tanks&Temples, and Dr. Johnson from Deep Blending. Fig. 6 presents the results for each scene.

Our method initially uses substantially fewer Gaussians with slightly lower reconstruction quality. As training progresses, it surpasses 3DGS in PSNR at the 12.5K checkpoint on Bicycle and the 7.5K checkpoint on Truck and Dr. Johnson. At 20K iterations, our method already exceeds the PSNR achieved by 3DGS at 30K iterations on all three scenes, while using substantially fewer Gaussians. These results show that our loss-aware density control improves mid-to-late-stage PSNR at a given iteration budget while maintaining a more compact Gaussian representation.

Table 12: LPIPS scores for Mip-NeRF360, Tanks&Temples, and Deep Blending scenes.

| | Mip-NeRF360 | | | | | | | | | Tanks&Temples | | Deep Blending | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | bicycle | flowers | garden | stump | treehill | room | counter | kitchen | bonsai | train | truck | drjohnson | playroom |
| 3DGS | 0.212 | 0.336 | 0.108 | 0.216 | 0.324 | 0.220 | 0.201 | 0.128 | 0.205 | 0.214 | 0.151 | 0.245 | 0.240 |
| AbsGS | $0.172_{\pm 0.00}$ | $0.271_{\pm 0.00}$ | $0.099_{\pm 0.00}$ | $0.197_{\pm 0.00}$ | $0.281_{\pm 0.00}$ | $0.200_{\pm 0.00}$ | $0.188_{\pm 0.00}$ | $0.122_{\pm 0.00}$ | $0.190_{\pm 0.00}$ | $0.193_{\pm 0.00}$ | $0.132_{\pm 0.00}$ | $0.241_{\pm 0.00}$ | $0.232_{\pm 0.00}$ |
| PixelGS | $0.180_{\pm 0.00}$ | $0.261_{\pm 0.00}$ | $0.099_{\pm 0.00}$ | $0.188_{\pm 0.00}$ | $0.277_{\pm 0.00}$ | $0.210_{\pm 0.00}$ | $0.183_{\pm 0.00}$ | $0.119_{\pm 0.00}$ | $0.191_{\pm 0.00}$ | $0.179_{\pm 0.00}$ | $0.120_{\pm 0.00}$ | $0.255_{\pm 0.00}$ | $0.241_{\pm 0.00}$ |
| DC4GS | $0.166_{\pm 0.00}$ | $0.269_{\pm 0.00}$ | $0.099_{\pm 0.00}$ | $0.188_{\pm 0.00}$ | $0.272_{\pm 0.00}$ | $0.202_{\pm 0.00}$ | $0.190_{\pm 0.00}$ | $0.122_{\pm 0.00}$ | $0.191_{\pm 0.00}$ | $0.189_{\pm 0.00}$ | $0.132_{\pm 0.00}$ | $0.238_{\pm 0.00}$ | $0.233_{\pm 0.00}$ |
| GDAGS | $0.171_{\pm 0.00}$ | $0.274_{\pm 0.00}$ | $0.099_{\pm 0.00}$ | $0.193_{\pm 0.00}$ | $0.279_{\pm 0.00}$ | $0.204_{\pm 0.00}$ | $0.192_{\pm 0.00}$ | $0.122_{\pm 0.00}$ | $0.192_{\pm 0.00}$ | $0.197_{\pm 0.00}$ | $0.137_{\pm 0.00}$ | $0.239_{\pm 0.00}$ | $0.233_{\pm 0.00}$ |
| ResGS | $0.212_{\pm 0.00}$ | $0.339_{\pm 0.00}$ | $0.110_{\pm 0.00}$ | $0.218_{\pm 0.00}$ | $0.331_{\pm 0.00}$ | $0.219_{\pm 0.00}$ | $0.202_{\pm 0.00}$ | $0.127_{\pm 0.00}$ | $0.205_{\pm 0.00}$ | $0.208_{\pm 0.00}$ | $0.148_{\pm 0.00}$ | $0.237_{\pm 0.00}$ | $0.240_{\pm 0.00}$ |
| SteepGS | $0.264_{\pm 0.00}$ | $0.399_{\pm 0.00}$ | $0.127_{\pm 0.00}$ | $0.274_{\pm 0.00}$ | $0.376_{\pm 0.00}$ | $0.229_{\pm 0.00}$ | $0.216_{\pm 0.00}$ | $0.135_{\pm 0.00}$ | $0.211_{\pm 0.00}$ | $0.226_{\pm 0.00}$ | $0.160_{\pm 0.00}$ | $0.249_{\pm 0.00}$ | $0.254_{\pm 0.00}$ |
| Ours | $0.213_{\pm 0.00}$ | $0.342_{\pm 0.00}$ | $0.112_{\pm 0.00}$ | $0.219_{\pm 0.00}$ | $0.335_{\pm 0.00}$ | $0.218_{\pm 0.00}$ | $0.199_{\pm 0.00}$ | $0.128_{\pm 0.00}$ | $0.202_{\pm 0.00}$ | $0.205_{\pm 0.00}$ | $0.142_{\pm 0.00}$ | $0.245_{\pm 0.00}$ | $0.246_{\pm 0.00}$ |

Table 13: Ablation study of the proposed components on Tanks&Temples and Deep Blending. Variants (a)–(d) follow Tab. 2. Num denotes the average number of Gaussians in millions.

| Dataset | Tanks&Temples | | | | Deep Blending | | | |
|---|---|---|---|---|---|---|---|---|
| Method | PSNR↑ | SSIM↑ | LPIPS↓ | Num | PSNR↑ | SSIM↑ | LPIPS↓ | Num |
| (a): Baseline [32] | 23.48 | 0.839 | 0.193 | 1.26 | 29.59 | 0.904 | 0.251 | 1.57 |
| (b): (a) + Sec. 4.2 | 23.53 | 0.842 | 0.191 | 1.27 | 29.65 | 0.907 | 0.248 | 1.64 |
| (c): (b) + Sec. 4.3 | 23.82 | 0.845 | 0.189 | **0.98** | 29.70 | **0.907** | 0.245 | **1.31** |
| (d): (c) + Sec. 4.4 | **24.20** | **0.853** | **0.174** | 1.04 | **29.79** | 0.906 | **0.245** | 1.32 |

The LPIPS trends are scene-dependent. At 30K iterations, our method achieves lower LPIPS on Truck, whereas 3DGS obtains slightly lower LPIPS on Bicycle and Dr. Johnson.

### B.5 Regional Analysis and Failure Cases

To examine where our method performs well or underperforms, we conduct a patch-level comparison with DC4GS [11]. We divide each rendered image into an $8 \times 10$ grid and compute PSNR, SSIM, and LPIPS for each patch. For each metric, we inspect the top-50 patches showing the largest improvements and the top-50 showing the largest degradations relative to DC4GS across all evaluated datasets, scenes, and test images.

Among the inspected patches, our method effectively reconstructs object structures and object-background boundaries. However, preserving fine-grained details in regions with repeated high-frequency textures, such as roads, grass, gravel, and foliage, remains challenging. In these regions, our reconstructions tend to appear smoother than those of DC4GS. Fig. 7 presents representative examples of these strengths and limitations.

A possible explanation lies in our loss-aware allocation criterion, which evaluates density-control operations according to their predicted local reduction in photometric loss. In texture-rich regions, a few relatively large Gaussians may already approximate the dominant colors and low-frequency appearance, leaving only small predicted gains from further splitting or cloning. Consequently, the method may allocate insufficient additional Gaussians to preserve fine textures. This behavior may partially explain the LPIPS gap with DC4GS, which persists even under comparable Gaussian budgets (Tab. 15). We therefore consider improving perceptual fidelity in high-frequency texture regions within our loss-aware framework an important direction for future work.

## C Dataset licenses

- Mip-NeRF 360 [3]: no license terms provided. Available at `https://jonbarron.info/mipnerf360/`.
- Deep Blending [8]: no license terms provided. Available at `http://visual.cs.ucl.ac.uk/pubs/deepblending/`.
- Tanks & Temples [16]: made available under Creative Commons Attribution-NonCommercial-ShareAlike 3.0 License. `https://www.tanksandtemples.org/license/`.

Table 14: Matched-budget comparison with 3DGS-MCMC. Mip-NeRF360 results are averaged over seven publicly available scenes, unlike the nine-scene results in Tab. 15. Num denotes the average number of Gaussians in millions.

| Dataset | Mip-NeRF360 (7 scenes) | | | | Tanks&Temples | | | | Deep Blending | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | PSNR↑ | SSIM↑ | LPIPS↓ | Num | PSNR↑ | SSIM↑ | LPIPS↓ | Num | PSNR↑ | SSIM↑ | LPIPS↓ | Num |
| 3DGS [12] | 28.90 | 0.863 | 0.196 | 1.8 | 23.62 | 0.842 | 0.186 | 1.0 | 29.48 | 0.901 | 0.254 | 1.3 |
| 3DGS-MCMC [14] | 29.28 | **0.885** | **0.168** | 1.8 | 24.04 | 0.852 | 0.176 | 1.0 | 29.53 | 0.897 | 0.266 | 1.3 |
| Ours | **29.30** | 0.874 | 0.184 | 1.8 | **24.20** | **0.853** | **0.174** | 1.0 | **29.79** | **0.906** | **0.245** | 1.3 |

Table 15: Comparison with DC4GS under comparable Gaussian budgets.

| Dataset | Mip-NeRF360 (9 scenes) | | | | Tanks&Temples | | | | Deep Blending | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | PSNR↑ | SSIM↑ | LPIPS↓ | Num | PSNR↑ | SSIM↑ | LPIPS↓ | Num | PSNR↑ | SSIM↑ | LPIPS↓ | Num |
| 3DGS [12] | 27.32 | 0.804 | 0.233 | 1.9 | 23.62 | 0.842 | 0.186 | 1.0 | 29.48 | 0.901 | 0.254 | 1.3 |
| DC4GS [11] | 27.52 | 0.816 | **0.201** | 1.9 | 23.95 | 0.852 | **0.168** | 1.0 | 29.65 | 0.901 | **0.244** | 1.3 |
| Ours | **27.72** | **0.818** | 0.219 | 1.9 | **24.20** | **0.853** | 0.174 | 1.0 | **29.79** | **0.906** | 0.245 | 1.3 |

# D Broader Impact

Our work improves the efficiency of 3D Gaussian Splatting by replacing heuristic density-control operations with loss-aware, primitive-wise decisions. By reducing redundant Gaussian allocation and avoiding abrupt global opacity reset, the proposed method can lower memory and rendering costs while maintaining reconstruction quality. This may support more practical deployment of real-time 3D scene representations on resource-constrained devices, including mobile platforms and AR/VR systems, and may reduce storage, bandwidth, and computational demands.

However, more efficient 3D reconstruction can also make it easier to capture and distribute detailed representations of real-world environments. Such capability may raise privacy concerns when applied to private spaces, people, or sensitive objects without consent, and could be used in surveillance or synthetic media pipelines.

# E Limitations and Future Work.

While our method provides a loss-aware density control strategy for 3D Gaussian Splatting, it relies on tractable local surrogate approximations to enable efficient per-Gaussian decisions. This design choice suggests several directions for further refinement. First, as described in Sec. A.5, our curvature estimate is an approximation of the full training objective. This approximation may introduce a discrepancy between the estimated curvature and the true objective curvature, which can occasionally lead to suboptimal densification decisions. Second, our surrogate approximates each operation point-wise. While rendering interactions are implicitly reflected in the back-propagated signals, it does not explicitly model cross-Gaussian effects. Third, the additional surrogate computations increase training time and peak GPU memory usage, although they are not required at inference (Tab. 4). Finally, preserving fine details in repeated high-frequency textures remains challenging, which can limit perceptual fidelity (Sec. B.5).

Future directions involve curvature estimation that more faithfully reflects the full training objective, split operations that consider parameters beyond position and scale, interaction-aware surrogate modeling, and improvements in training efficiency and perceptual fidelity within our loss-aware framework.

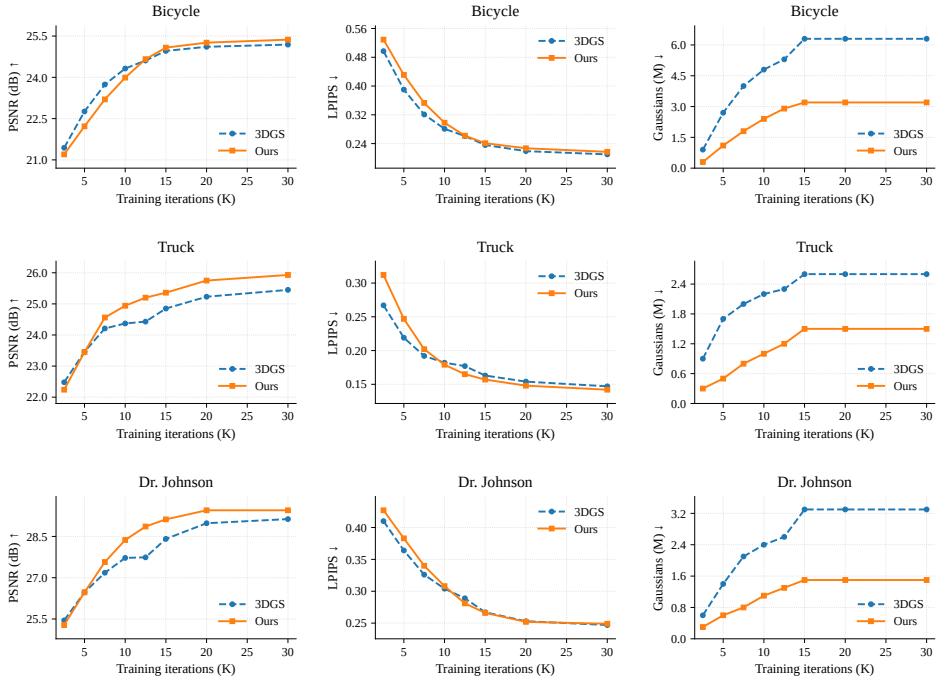



Figure 6: Convergence comparison with 3DGS on Bicycle, Truck, and Dr. Johnson. Columns show PSNR, LPIPS, and the number of Gaussians.

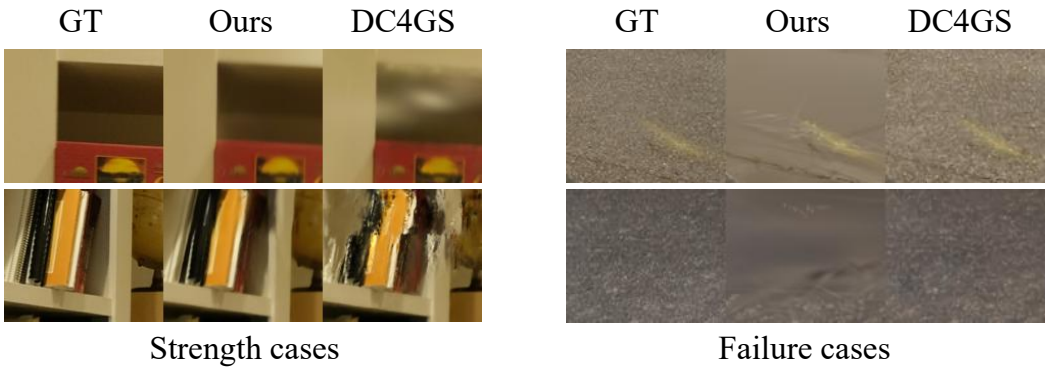



Figure 7: Patch-level comparison with DC4GS. The examples illustrate strengths in reconstructing object structures and object-background boundaries, as well as limitations in preserving fine details in repeated high-frequency textures.

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims in the abstract and introduction accurately reflect the proposed method and are supported by the theoretical derivations and experimental results.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of our method in Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We state the assumptions for our theoretical derivations and provide the corresponding proofs and derivations in the Appendix.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide the implementation details in Sec. 5.1 and Appendix A.1, and include the implementation code in the supplementary material.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We attach the implementation code in supplementary.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We report the experiment details used in our experiments in Sec. 5.1 and Appendix. A.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report the mean and standard deviation of each evaluation metric over five runs in Tab. 1 and Appendix. B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We report the computational resources used in our experiments in Sec. 5.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We provide this in Appendix. D.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: This work does not involve personally identifiable information, private user data, or sensitive datasets, and therefore does not introduce direct privacy-related risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We provide the licenses of the datasets used in our experiments in Appendix C.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: We do not provide new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This work does not include crowdsourcing experiments or studies involving human participants.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This work does not include crowdsourcing experiments or studies involving human participants.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: We used LLMs only for auxiliary purposes, such as polishing English sentences, editing, and LaTeX formatting.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.