CONSTRAINTGS: MODELING PRIMITIVE-LEVEL RECONSTRUCTION CONSTRAINTS FOR SPARSE-VIEW GAUSSIAN SPLATTING

Anonymous authors
Paper under double-blind review

ABSTRACT

3D Gaussian Splatting enables high-quality, real-time novel-view synthesis but is prone to overfitting under sparse-view supervision. Beyond the limited number of training views, effective supervision is unevenly distributed across Gaussian primitives: even under the same observations, individual primitives receive substantially different amounts and directional distributions of rendering support. We introduce ConstraintGS, a framework for modeling and exploiting primitive-level reconstruction constraints under sparse views. We analyze how rendering support and viewing geometry jointly characterize constraint strength and examine its relationship with local photometric response. Building on this analysis, we construct a self-opacity-normalized cross-view descriptor that combines support magnitude and directional coverage. The descriptor allocates opacity attenuation under a prescribed mean budget and per-primitive bounds, assigning stronger suppression to weakly constrained primitives. ConstraintGS integrates into existing Gaussian splatting pipelines without modifying their inference procedure. Experiments on three widely used benchmarks demonstrate improved novel-view synthesis quality across multiple Gaussian splatting methods under sparse-view conditions.

1 INTRODUCTION

3D Gaussian Splatting (3DGS) (Kerbl et al., 2023) enables high-quality novel view synthesis (NVS) with real-time rendering, but its reconstruction quality depends strongly on the available observations. The difficulty of collecting densely sampled images motivates sparse-view reconstruction (Wang et al., 2023; Yang et al., 2023; Truong et al., 2023; Zhang et al., 2024a; Bao et al., 2025). With few training views, different Gaussian configurations can explain the observed images similarly well while producing substantially different novel views, leading to floaters, unstable occlusions, and incomplete structures.

Existing methods mitigate this ambiguity through depth guidance (Li et al., 2024; Zhu et al., 2024), pseudo-view synthesis (Zhu et al., 2024), or diffusion priors (Kong et al., 2025), which provide auxiliary supervision but introduce computation and dependence on estimated or generated targets. Other approaches regularize Gaussian representations through co-regularization (Zhang et al., 2024a) or stochastic dropout (Park et al., 2025; Xu et al., 2025). Despite their effectiveness, a basic question remains insufficiently addressed: *how much effective cross-view constraint does each Gaussian actually receive?*

Training-view count alone cannot answer this question. Within the same scene, foreground attenuation can leave some primitives with little rendering support. Even substantial support may provide limited geometric evidence when observations follow similar viewing directions. Primitive-level constraint therefore depends on both *support magnitude* and *directional coverage*. Our diagnostics reveal uneven rendering support across primitives and a relationship between constraint scores and local photometric responses. These findings motivate stronger regularization of less-supported contributions.

We introduce **ConstraintGS**, a framework that uses these differences to guide opacity attenuation without introducing external depth or geometry priors. The descriptor guides opacity attenuation to regulate each primitive's rendering contribution, while geometric refinement remains governed

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

by the host's original training objective. Specifically, we aggregate effective rendering contributions with viewing-direction projectors and normalize support to a common self-opacity scale to remove its direct dependence on the primitive's own opacity under fixed rendering conditions. A log-determinant score summarizes support magnitude and directional coverage. Scene-relative scores allocate stronger attenuation to weakly constrained primitives, while a prescribed mean budget and a per-primitive cap control its strength. The update follows the host optimizer step without differentiating through the scores or adding a training loss.

ConstraintGS requires no additional learned model and preserves the host's inference procedure. A shared regularization configuration improves multiple Gaussian splatting pipelines across LLFF (Mildenhall et al., 2019), Mip-NeRF 360 (Barron et al., 2022), and Blender (Mildenhall et al., 2021).

Our main contributions are:

- We analyze primitive-level reconstruction constraints through rendering support and directional coverage, and examine their relationship with local photometric response.
- We construct a cross-view constraint descriptor that factors out direct self-opacity dependence under fixed rendering conditions and jointly captures support magnitude and directional coverage.
- We introduce constraint-adaptive opacity attenuation under a fixed mean budget and per-primitive bounds, improving multiple Gaussian splatting methods with a shared regularization configuration.

2 Related Work

Novel View Synthesis. Neural Radiance Fields (NeRF) (Mildenhall et al., 2021) represent scenes as continuous radiance and density fields and synthesize novel views through differentiable volume rendering. Subsequent advances improve reconstruction quality through multiscale modeling and anti-aliasing (Barron et al., 2021). 3D Gaussian Splatting (3DGS) (Kerbl et al., 2023) instead adopts explicit Gaussian primitives and differentiable rasterization, enabling high-quality, real-time rendering. Further developments improve detail recovery, density control, and representation compactness (Ye et al., 2024; Zhang et al., 2024b; Lee et al., 2024). Under sparse observations, however, the flexibility of these representations permits multiple configurations that explain training images but differ substantially at novel viewpoints.

Sparse-View Reconstruction with Auxiliary Priors. One line of work reduces reconstruction ambiguity by introducing additional geometric or appearance cues. Depth-supervised NeRF methods use sparse geometry or pretrained depth predictions to constrain reconstruction (Deng et al., 2022; Roessle et al., 2022; Wang et al., 2023), while DietNeRF (Jain et al., 2021) uses pretrained semantic features. Generalizable methods learn cross-scene reconstruction priors, including pixelNeRF (Yu et al., 2021), IBRNet (Wang et al., 2021), MVSNeRF (Chen et al., 2021), and GeoNeRF (Johari et al., 2022). For sparse-view 3DGS, monocular or binocular depth guidance (Li et al., 2024; Zhu et al., 2024; Han et al., 2024) and diffusion priors (Kong et al., 2025; Wang et al., 2024) likewise supply auxiliary targets. Such cues help resolve ambiguities left by sparse observations, but their effectiveness depends on the reliability of the additional supervision.

Regularizing Sparse-View Representations. A complementary direction constrains reconstruction through the representation or optimization process. FreeNeRF (Yang et al., 2023) regularizes frequency content, while RegNeRF (Niemeyer et al., 2022) constrains geometry and appearance at unobserved viewpoints. Gaussian-based approaches employ coherent parameterization (Paliwal et al., 2024), geometric regularization (Liu et al., 2024), co-regularization (Zhang et al., 2024a), and stochastic dropout (Park et al., 2025; Xu et al., 2025). D^2GS (Song et al., 2026) combines depth-and-density-guided dropout with distance-aware fidelity enhancement, adapting its treatment of near-field and far-field regions.

ConstraintGS evaluates cross-view support at a common self-opacity scale, accounting for both support magnitude and directional coverage. The resulting descriptor guides continuous opacity attenuation under a prescribed mean budget and per-primitive bounds.

3 What Constrains a Gaussian Primitive?

Sparse-view supervision constrains individual Gaussians unevenly, even when they are optimized from the same training images. This variation depends on both the amount of rendering support each Gaussian receives and the viewing directions from which that support is obtained. In this section, we examine these two aspects—support magnitude and directional coverage—and investigate how a descriptor combining them relates to local photometric response. Detailed measurement, perturbation, and aggregation protocols are provided in Appendix A.

3.1 Effective Rendering Support

In 3DGS (Kerbl et al., 2023), the color of a pixel $\mathbf{x}$ in training view t is obtained by front-to-back alpha compositing:

$$\mathbf{C}_t(\mathbf{x}) = \sum_i T_{it}(\mathbf{x}) a_{it}(\mathbf{x}) \boldsymbol{\rho}_{it}, \qquad T_{it}(\mathbf{x}) = \prod_{j<i} \big(1 - a_{jt}(\mathbf{x})\big). \tag{1}$$

Here, a_{it} is projected alpha, T_{it} is the transmittance preceding primitive i, and $\boldsymbol{\rho}_{it}$ is its color for view t; the background term is omitted for brevity. Holding the remaining rendering quantities fixed yields $\partial \mathbf{C}_t / \partial \boldsymbol{\rho}_{it} = T_{it} a_{it} \mathbf{I}$. Thus, a primitive's influence on the image depends on its effective contribution, not simply on whether it lies inside the camera frustum. Repeated foreground attenuation can leave a visible primitive with little photometric support.

We summarize this contribution by a nonnegative per-view support w_{it} derived from $T_{it} a_{it}$ and define total support as $s_i = \sum_t w_{it}$. To examine support beyond differences in the primitives' own opacity, Figure 1 reports a center-sampled diagnostic of unit-opacity support across eight LLFF scenes (Mildenhall et al., 2019). Support varies substantially within every scene: the least-supported 50% of primitives account for only 19.9% of total unit-opacity support in the across-scene median. Thus, the same training views provide markedly unequal support to individual primitives, which a scene-level view count cannot capture.

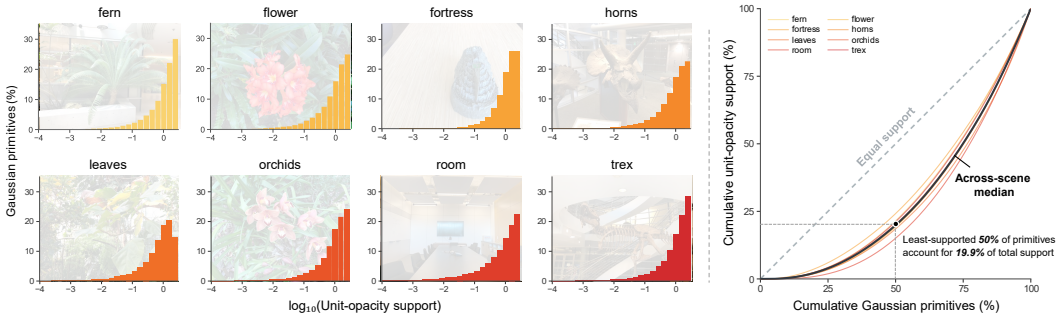

Figure 1: Uneven rendering support across Gaussian primitives on LLFF. Left: Histograms of unit-opacity support across eight scenes in $\log_{10}$ space. Bar heights indicate the percentage of primitives in each bin. Background images identify scenes. Right: Cumulative support with primitives ranked by increasing support within each scene. The dashed diagonal denotes equal support, and the black curve shows the pointwise median across scenes. The least-supported 50% of primitives account for only 19.9% of total unit-opacity support in the across-scene median.

3.2 Directional Coverage of the Support

Total support does not specify which spatial directions the training rays cover. Consider a Gaussian center $\boldsymbol{\mu}_i$ and camera center $\mathbf{o}_t$. Their unit viewing direction and transverse projector are

$$\mathbf{r}_{it} = \frac{\boldsymbol{\mu}_i - \mathbf{o}_t}{\|\boldsymbol{\mu}_i - \mathbf{o}_t\|_2}, \qquad \mathbf{P}_{it} = \mathbf{I} - \mathbf{r}_{it}\mathbf{r}_{it}^\top. \tag{2}$$

For a small displacement $\delta\boldsymbol{\mu}_i$, the first-order change in viewing direction is $\delta \mathbf{r}_{it} = d_{it}^{-1} \mathbf{P}_{it} \delta\boldsymbol{\mu}_i$, where $d_{it} = \|\boldsymbol{\mu}_i - \mathbf{o}_t\|_2$. Displacement along a ray leaves its direction unchanged, whereas trans-

verse displacement changes it. Observations along similar rays therefore provide redundant directional coverage; a complementary view can expose displacement left ambiguous by the others.

For a unit spatial direction $\mathbf{v}$, weighting this component by effective support gives

$$\mathbf{M}_i = \sum_t w_{it} \mathbf{P}_{it}, \qquad \mathbf{v}^\top \mathbf{M}_i \mathbf{v} = \sum_t w_{it} \|\mathbf{P}_{it}\mathbf{v}\|_2^2. \tag{3}$$

The quadratic form measures accumulated transverse support along $\mathbf{v}$. A small value can result from little support overall or from supporting rays that leave $\mathbf{v}$ poorly covered. The matrix retains support and ray geometry; distance-dependent scaling and appearance derivatives of the full photometric objective are outside this descriptor. We next test whether a score combining support magnitude and directional coverage is associated with local photometric response.

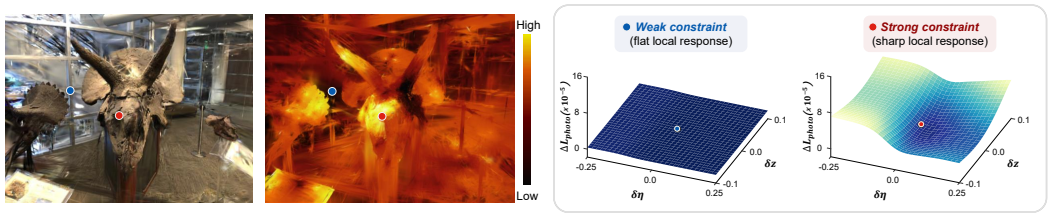

Figure 2: Constraint strength and local photometric response. (a) Rendered image and selected Gaussians. (b) Constraint score heatmap. (c) Depth–opacity-logit response surfaces with other Gaussians fixed. The lower-scoring Gaussian (blue) exhibits a flatter response than the higher-scoring one (red). Both surfaces share the same axis ranges and color scale.

3.3 Relation to Local Photometric Response

To examine the photometric relevance of this characterization, we use the opacity-normalized constraint score defined in Section 4.1. We probe local photometric responses by perturbing an individual Gaussian's depth and opacity logit while holding the remaining primitives fixed. Figure 2 illustrates two primitives with different scores. Under the same perturbation ranges, the lower-scoring primitive exhibits a flatter loss response, whereas the higher-scoring primitive produces larger photometric changes. We further examine this relationship across scenes from LLFF, Mip-NeRF 360, and Blender. Within each scene, primitives are grouped into constraint-score deciles, and their normalized local photometric responses are summarized by the median within each group. Figure 3 shows that higher-scoring groups generally exhibit stronger responses across all three datasets, although the trend varies across scenes and levels off in the higher LLFF deciles.

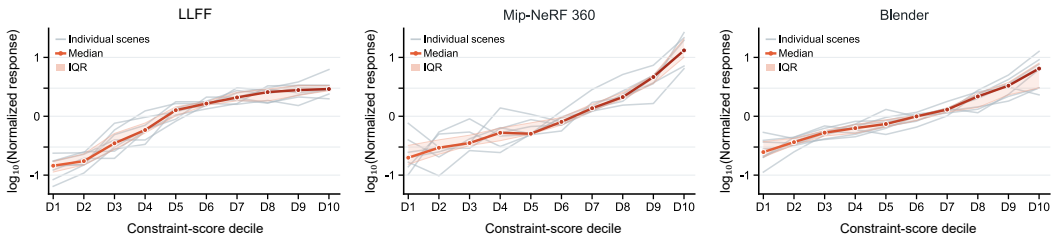

Figure 3: Constraint score and local photometric response. Within each scene, Gaussian primitives are grouped into constraint-score deciles, from lowest (D1) to highest (D10). Thin lines show the median normalized response within each decile for individual scenes; the highlighted line and shaded band show the across-scene median and interquartile range, respectively. Responses are displayed in $\log_{10}$ space.

These diagnostics show an empirical association between the constraint score and local photometric response, motivating stronger regularization of weakly constrained contributions. The following section introduces a bounded opacity attenuation rule based on this score.

4 CONSTRAINTGS

ConstraintGS uses the cross-view support characterized above to allocate opacity attenuation across Gaussian primitives, while retaining the host's geometric optimization. As illustrated in Figure 4, a self-opacity-normalized cross-view descriptor determines relative attenuation priorities across primitives. A prescribed budget fixes the mean relative opacity reduction, while a per-primitive cap limits individual attenuation. We operate on an eligible set $\mathcal{A}$ of primitives whose activated opacity exceeds a small floor $o_{\min}$ and that have at least one valid support view.

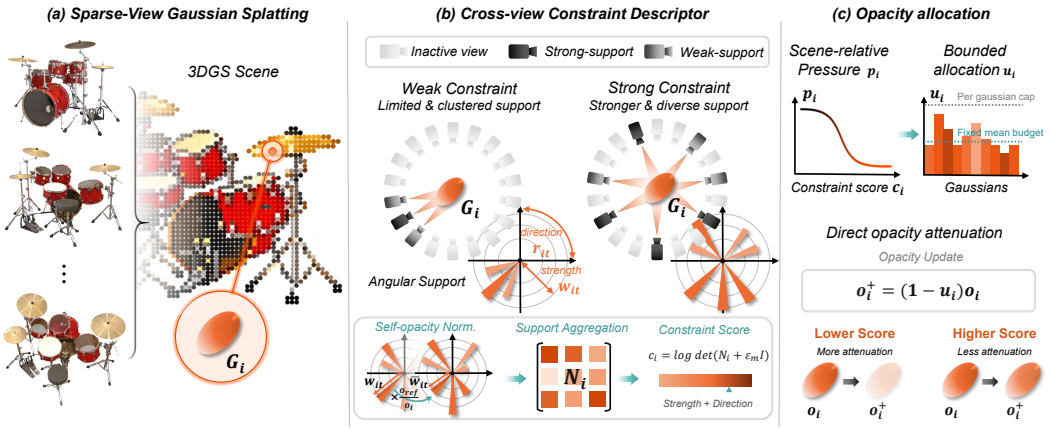

Figure 4: Overview of ConstraintGS. (a) Gaussian reconstruction from sparse input views. (b) Self-opacity-normalized rendering support and viewing directions jointly form a continuous constraint score. (c) Scene-relative scores guide bounded opacity attenuation under a fixed mean budget, with stronger suppression assigned to lower-scoring Gaussians.

4.1 CROSS-VIEW CONSTRAINT DESCRIPTOR

We construct the descriptor from per-view rendering support and the viewing projectors introduced in Section 3.2. Specifically, we evaluate w_{it} at the projected Gaussian center using a compositing-order replay. If the center is invalid but the rasterizer accepts contributions elsewhere in the projected footprint, we use the largest accepted $T_{it}a_{it}$ in that footprint; views with neither form of support contribute zero. These weights define the directional support matrix $\mathbf{M}_i$.

Support at a common opacity scale. Raw rendering support depends on the primitive's own opacity. Using it directly for allocation can therefore create a feedback effect: attenuation lowers support and may trigger further suppression even when the cameras and surrounding scene remain unchanged. We factor out this direct opacity dependence and compare primitives at a common reference opacity.

Hold geometry, other primitives, compositing order, and accepted samples fixed. On an unsaturated alpha branch, $a_{it}(\mathbf{x}) = o_i g_{it}(\mathbf{x})$, where $o_i \in (0,1)$ is activated opacity and g_{it} is the projected Gaussian profile. The per-view support then factors as $w_{it} = o_i \xi_{it}$, where ξ_{it} contains the profile and preceding transmittance and is independent of o_i. This factorization holds for both the center coefficient and the fixed-set footprint maximum. We therefore define

$$\bar{w}_{it} = \frac{o_{\text{ref}}}{o_i} w_{it}, \qquad \mathbf{N}_i = \sum_t \bar{w}_{it} \mathbf{P}_{it} = \frac{o_{\text{ref}}}{o_i} \mathbf{M}_i, \tag{4}$$

where $o_{\text{ref}} > 0$ is the median opacity in $\mathcal{A}$ at regularization activation and remains fixed thereafter. This common reference retains a representative opacity scale. Under the stated local conditions,

$\bar{w}_{it} = o_{\text{ref}}\xi_{it}$ is invariant to rescaling the primitive's own opacity. The descriptor retains attenuation by other primitives and the support's directional coverage.

A scalar constraint score. We summarize the normalized matrix by

$$c_i = \log\det(\mathbf{N}_i + \epsilon_m \mathbf{I}), \tag{5}$$

where $\epsilon_m > 0$ keeps the score finite for rank-deficient support. To see how the score combines support and geometry, let $\bar{s}_i = \sum_t \bar{w}_{it} > 0$, $\pi_{it} = \bar{w}_{it}/\bar{s}_i$, and $\mathbf{Q}_i = \sum_t \pi_{it}\mathbf{r}_{it}\mathbf{r}_{it}^\top$. Its eigenvalues q_{ik} are nonnegative and sum to one, yielding

$$\mathbf{N}_i = \bar{s}_i(\mathbf{I} - \mathbf{Q}_i), \qquad c_i = \sum_{k=1}^{3} \log[\bar{s}_i(1 - q_{ik}) + \epsilon_m]\,. \tag{6}$$

At fixed directional configuration, increasing $\bar{s}_i$ strictly increases c_i. At fixed positive support, the symmetric, strictly concave dependence on q_{ik} favors a more balanced spectrum. By comparison, $\operatorname{tr}(\mathbf{N}_i) = 2\bar{s}_i$ cannot distinguish directional configurations. The log-determinant retains both support magnitude and directional balance without a separate mixing weight. It also inherits the local opacity invariance of $\mathbf{N}_i$.

4.2 Constraint-Adaptive Opacity Allocation

We convert the score into a scene-relative pressure. For a nonempty $\mathcal{A}$, let $\widetilde{c}$ and d_c be the median and consistency-scaled median absolute deviation of its scores. We compute

$$\widehat{c}_i = \frac{c_i - \widetilde{c}}{d_c + \epsilon_n}, \qquad p_i = \sigma(-\widehat{c}_i), \tag{7}$$

where $\epsilon_n > 0$ ensures numerical stability and σ is the sigmoid function. This bounded, monotone mapping assigns larger pressure to lower relative constraint scores. Robust normalization accounts for the scene's score distribution, while the sigmoid provides a continuous transition across constraint levels.

To translate pressure into an attenuation fraction, prescribe a mean budget b and upper bound $u_{\max}$, with $0 < b \le u_{\max} < 1$. We first set $u_i^{(0)} = bp_i/\bar{p}$, where $\bar{p} = |\mathcal{A}|^{-1}\sum_{j\in\mathcal{A}} p_j$, and compute

$$u_i = \operatorname{clip}(u_i^{(0)} - \beta, 0, u_{\max}), \qquad \frac{1}{|\mathcal{A}|}\sum_{i\in\mathcal{A}} u_i = b. \tag{8}$$

The scalar β is chosen to satisfy the mean constraint. This is the Euclidean projection of $\mathbf{u}^{(0)}$ onto the bounded budget set: it preserves the pressure ordering, with possible ties at the bounds, while changing the initial allocation as little as possible.

After the host optimizer produces opacity o_i^{host}, we apply

$$o_i^{+} = (1 - u_i)o_i^{\text{host}}. \tag{9}$$

Primitives outside $\mathcal{A}$ receive $u_i = 0$, and attenuation is skipped when the set is empty. The score-derived allocation therefore specifies the actual relative opacity reduction. Our attenuation is applied separately from the host optimizer and uses cross-view support to distribute a prescribed relative-opacity budget. The host optimizer retains its moment states. The budget controls mean relative attenuation, not total opacity removal or image change.

Constraint scores and allocation pressures are computed without gradient tracking and refreshed periodically, as well as after topology changes or opacity resets. Between refreshes, we reuse the cached pressures and recompute the bounded allocation over the current eligible set. ConstraintGS requires no additional learned model and leaves the host's inference procedure unchanged. Appendix B provides further analysis of the descriptor, the derivation of the bounded allocation, and the training procedure.

Table 1: Quantitative comparison on the LLFF dataset. The best, second-best, and third-best results in each metric column are highlighted in red, orange, and yellow, respectively. Bold indicates the better result within each baseline pair.

Method	Setting	LLFF (3-view 1/8 Resolution)			LLFF (3-view 1/4 Resolution)		
		PSNR(↑)	SSIM(↑)	LPIPS(↓)	PSNR(↑)	SSIM(↑)	LPIPS(↓)
3DGS (Kerbl et al., 2023)	baseline	19.36	0.651	0.232	16.94	0.488	0.402
	w/ ours	**21.05**	**0.725**	**0.191**	**20.19**	**0.686**	**0.257**
FSGS (Zhu et al., 2024)	baseline	20.43	0.682	0.248	19.71	0.642	0.283
	w/ ours	**21.26**	**0.733**	**0.196**	**20.45**	**0.691**	**0.252**
CoR-GS (Zhang et al., 2024a)	baseline	20.17	0.703	0.202	19.96	0.696	0.250
	w/ ours	**21.28**	**0.737**	**0.197**	**20.51**	**0.699**	**0.237**
DropGaussian (Park et al., 2025)	baseline	20.76	0.713	0.200	20.07	0.690	0.258
	w/ ours	**21.48**	**0.743**	**0.186**	**20.33**	**0.697**	**0.239**
D^{2}GS (Song et al., 2026)	baseline	21.35	0.746	0.179	20.56	0.695	0.254
	w/ ours	**21.57**	**0.754**	**0.173**	**20.71**	**0.700**	**0.204**

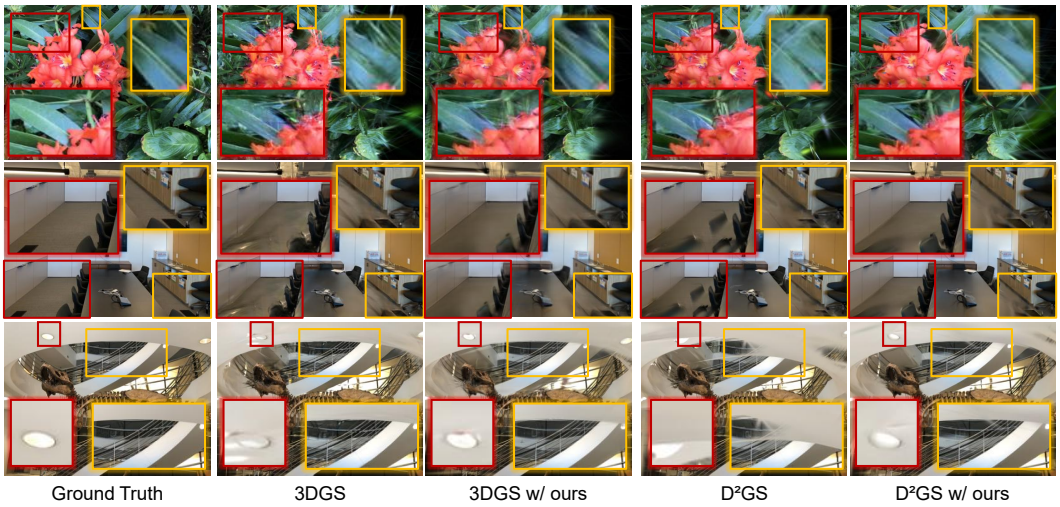

Figure 5: Qualitative comparison on LLFF. ConstraintGS produces clearer primary leaf veins (top), reduces floating artifacts over the floor (middle), and restores light boundaries in the red box while reducing ceiling floaters in the yellow box (bottom).

5 Experiments

Datasets and metrics. We evaluate on LLFF (Mildenhall et al., 2019), Mip-NeRF 360 (Barron et al., 2022), and Blender (Mildenhall et al., 2021). LLFF uses 3 training views at 1/8 and 1/4 resolution, while Mip-NeRF 360 uses 12 views at 1/8 resolution. For Blender, we follow FSGS (Zhu et al., 2024), using 8 training and 25 test views per scene at 400×400 resolution across eight scenes. We report PSNR, SSIM (Wang et al., 2004), and LPIPS (Zhang et al., 2018).

Baselines and implementation. We integrate ConstraintGS into the official implementations of 3DGS (Kerbl et al., 2023), FSGS (Zhu et al., 2024), CoR-GS (Zhang et al., 2024a), DropGaussian (Park et al., 2025), and D^{2}GS (Song et al., 2026). All training runs are conducted on a single NVIDIA GeForce RTX 4090 GPU. All main comparisons and ablation studies are conducted with three random seeds, and we report the mean results. D^{2}GS is omitted on Blender because no official configuration is available for this dataset. We share one regularization configuration across datasets, scenes, and hosts, setting the mean attenuation budget to $b = 0.015$ and the per-primitive cap to $u_{\max} = 4b = 0.06$ (Eq. 8). We set the activation fraction to $t_{\text{start}} = 0.8$, enabling attenuation after 80% of the total training iterations. Scores and pressures are refreshed every $K = 100$ iterations, with additional refreshes after topology changes or opacity resets. ConstraintGS applies attenuation after host optimizer steps, introducing no additional learned model or changes to inference.

Table 2: Quantitative comparison on the Mip-NeRF 360 and Blender datasets. The best, second-best, and third-best results in each metric column are highlighted in red, orange, and yellow, respectively. Bold indicates the better result within each baseline pair.

Method	Setting	Mip-NeRF 360			Blender		
		PSNR(↑)	SSIM(↑)	LPIPS(↓)	PSNR(↑)	SSIM(↑)	LPIPS(↓)
3DGS (Kerbl et al., 2023)	baseline	18.52	0.523	0.415	21.56	0.847	0.130
	w/ ours	**19.15**	**0.557**	**0.409**	**24.39**	**0.890**	**0.092**
FSGS (Zhu et al., 2024)	baseline	18.80	0.531	**0.418**	24.64	**0.895**	0.095
	w/ ours	**19.69**	**0.606**	0.447	**25.14**	**0.895**	**0.090**
CoR-GS (Zhang et al., 2024a)	baseline	19.52	0.558	0.418	24.43	**0.896**	**0.084**
	w/ ours	**19.93**	**0.583**	**0.402**	**25.15**	0.891	0.090
DropGaussian (Park et al., 2025)	baseline	19.74	0.577	0.364	25.42	0.888	**0.089**
	w/ ours	**20.13**	**0.607**	**0.342**	**25.80**	**0.893**	0.096
D^{2}GS (Song et al., 2026)	baseline	20.09	0.587	0.356	–	–	–
	w/ ours	**20.30**	**0.590**	**0.355**	–	–	–

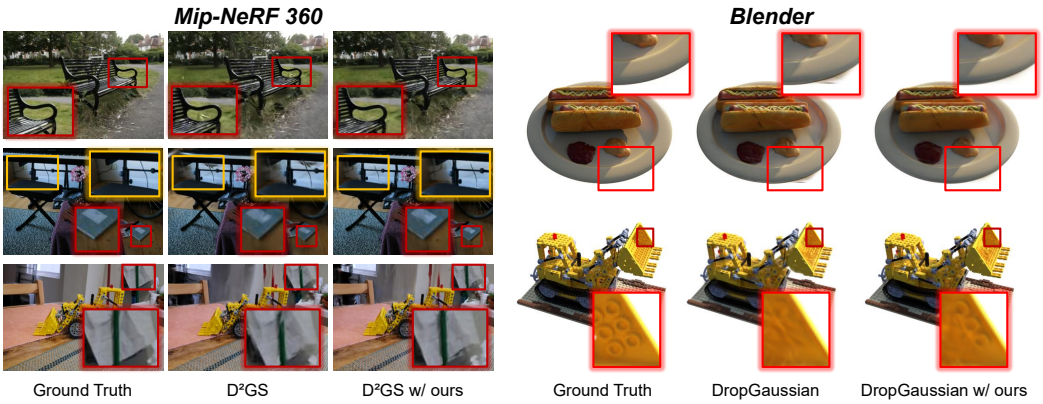

Figure 6: Qualitative comparisons on Mip-NeRF 360 and Blender. Left: ConstraintGS suppresses white floaters within the chair-armrest opening (top), reduces floaters in the yellow boxes and recovers ground texture and speckled details in the red boxes (middle), and improves the structural fidelity of the green background pole (bottom). Right: ConstraintGS improves shadow fidelity and plate-boundary definition in the *hotdog* scene (top) and better resolves the repeated circular surface details on the bucket in the *lego* scene (bottom).

Quantitative evaluation. Tables 1 and 2 demonstrate the applicability of ConstraintGS across datasets and Gaussian splatting methods. For vanilla 3DGS, PSNR gains reach 3.25 dB on LLFF and 2.83 dB on Blender. Gains also extend to specialized sparse-view methods, with improvements of 0.15–0.22 dB across the three evaluated D^{2}GS settings. All three metrics improve on LLFF, while SSIM and LPIPS gains are not uniform on the other datasets. Together, these results support ConstraintGS as a broadly applicable regularizer that complements existing pipelines without dataset- or host-specific regularization tuning.

Qualitative evaluation. Figure 5 shows that ConstraintGS improves leaf-vein definition and light-boundary fidelity while reducing floor and ceiling floaters on LLFF. Figure 6 further illustrates floater suppression and improved structural continuity on Mip-NeRF 360, together with more faithful shadows and plate boundaries in *hotdog* and better-preserved surface details in *lego* on Blender.

Ablation on opacity allocation. Table 3 evaluates allocation designs under the same attenuation budget, cap, and schedule. Uniform allocation sets $u_i = b$. Removing self-opacity normalization in Eq. 4 replaces $\mathbf{N}_i$ with $\mathbf{M}_i$ in Eq. 5. Based on Eq. 6, support-only allocation uses $\log(\bar{s}_i + \epsilon)$, while direction-only allocation uses $\log\det(\mathbf{I} - \mathbf{Q}_i + \epsilon_m \mathbf{I})$. All score-based variants retain the same pressure mapping and bounded allocation. Uniform attenuation improves PSNR from 19.36 to 20.40 dB. Descriptor-guided allocation provides a further gain of 0.65 dB under the same mean attenuation budget, reaching 21.05 dB. The full method also exceeds support-only and direction-only

allocation by 0.28 and 0.21 dB, respectively, with better SSIM and LPIPS. Removing self-opacity normalization degrades all three metrics. These comparisons demonstrate the reconstruction value of the descriptor's normalization and joint use of support magnitude and directional coverage. See Appendix C for further analysis and visualizations.

Parameter sensitivity. Table 4 evaluates four parameters controlling attenuation strength and timing. We vary the mean budget b and cap-to-budget ratio $u_{\max}/b$ in Eq. 8, together with the refresh interval K and activation fraction t_{start}. When varying b, we keep $u_{\max}/b = 4$. Across the tested settings, PSNR remains within 0.14 dB of the default configuration, indicating limited sensitivity within these ranges. Increasing the mean budget from 1.5% to 2.25%, however, raises LPIPS from 0.1910 to 0.2087 despite a small PSNR change, revealing a trade-off between attenuation strength and perceptual quality. The default configuration provides the best overall results.

Table 3: Ablation of ConstraintGS on LLFF with 3 training views at $1/8$ resolution, using 3DGS as the host. Checkmarks indicate enabled components. All attenuation variants share the same mean budget, cap, and schedule. Bold indicates the best result.

Opacity Attenuation	Self-Opacity Normalization	Support Magnitude	Directional Coverage	PSNR(↑)	SSIM(↑)	LPIPS(↓)
				19.36	0.651	0.232
✓				20.40	0.698	0.203
✓		✓	✓	20.70	0.708	0.218
✓	✓	✓		20.77	0.719	0.199
✓	✓		✓	20.84	0.698	0.197
✓	✓	✓	✓	**21.05**	**0.725**	**0.191**

Table 4: Parameter sensitivity of ConstraintGS on LLFF with 3 training views at 1/8 resolution. b denotes the mean relative opacity attenuation budget, $u_{\max}/b$ the cap-to-budget ratio, K the score-refresh interval in iterations, and t_{start} the fraction of total training iterations at which attenuation begins. When varying b, we scale $u_{\max}$ proportionally; otherwise, only the indicated parameter is changed. † marks the default setting. Bold indicates the best result within each group.

b (%)	PSNR(↑)	SSIM(↑)	LPIPS(↓)	$u_{\max}/b$	PSNR(↑)	SSIM(↑)	LPIPS(↓)
0.75	20.91	0.7156	0.1913	2	20.97	0.7222	0.1952
$1.5^{\dagger}$	**21.05**	**0.7250**	**0.1910**	$4^{\dagger}$	**21.05**	**0.7250**	**0.1910**
2.25	20.99	0.7184	0.2087	8	20.91	0.7203	0.1952
K	PSNR(↑)	SSIM(↑)	LPIPS(↓)	t_{start}	PSNR(↑)	SSIM(↑)	LPIPS(↓)
50	20.94	0.7211	0.1947	0.7	21.00	0.7222	0.1954
$100^{\dagger}$	**21.05**	**0.7250**	**0.1910**	$0.8^{\dagger}$	**21.05**	**0.7250**	**0.1910**
200	20.94	0.7209	0.1959	0.9	20.95	0.7196	0.1956

6 Conclusion

We presented ConstraintGS, which models primitive-level reconstruction constraints through rendering support and directional coverage. A self-opacity-normalized descriptor guides continuous opacity attenuation without introducing external geometric targets or changing the host's inference procedure. Under a prescribed mean budget and per-primitive bounds, the allocation applies stronger attenuation to weakly constrained contributions, connecting support analysis to regularization while retaining the host's original geometric optimization. Experiments across three datasets and multiple hosts show consistent PSNR improvements under a shared regularization configuration. Diagnostics and ablations support the descriptor's photometric relevance and its utility for allocating attenuation. Future work will improve support-estimation efficiency and explore more diverse capture conditions.

AI USE STATEMENT

We used generative AI tools to assist with manuscript revision and literature retrieval. Manuscript assistance included reorganizing the abstract and refining the contribution statements to better communicate self-opacity normalization, support magnitude, directional coverage, and bounded opacity allocation; improving transitions between the diagnostic analysis, method, and experiments; and reducing repetitive explanations in the introduction and Related Work section. The tools also helped refine the wording of the ablation and parameter-sensitivity discussions, distinguish empirical observations from analytical properties, and clarify the presentation of the mean relative attenuation budget and computational overhead.

We additionally used these tools to check mathematical notation and cross-references, identify duplicated equation labels in rendered references, add pointers from the main text to the diagnostic protocols, method derivations, and allocation analysis in the appendix, and improve figure and table captions. Specific editorial checks included removing repeated caption text, aligning qualitative descriptions with the displayed scenes and highlighted structures, correcting panel terminology in the ablation figure, and reviewing figure sizing and placement.

Literature assistance included searching for relevant studies and checking their relevance to the Related Work discussion, including CoherentGS (Paliwal et al., 2024), GeoRGS (Liu et al., 2024), DropoutGS (Xu et al., 2025), and binocular-guided Gaussian splatting (Han et al., 2024). We also used these tools to cross-check reference titles, author lists, publication venues, years, and bibliographic formatting against publication records and author-maintained resources.

ETHICS STATEMENT

This work studies sparse-view novel-view synthesis using the publicly available LLFF, Mip-NeRF 360, and Blender datasets. It does not involve human-subject studies or the collection of new personal data. As with other scene reconstruction and image synthesis techniques, applications involving private environments or identifiable individuals should respect consent and privacy. Our experiments focus on reconstruction quality and computational performance on established benchmarks.

REPRODUCIBILITY STATEMENT

We provide the implementation of ConstraintGS, experiment configurations, and instructions for training and evaluation in the supplementary code. Section 4 defines the constraint descriptor and opacity allocation procedure, including the conditions for self-opacity normalization. Appendix A documents the diagnostic measurements, perturbation protocol, and aggregation procedures. Appendix C explains the ablation variants, and Appendix D provides additional qualitative comparisons and computational overhead measurements. Together, these materials specify the method and experimental procedures needed to reproduce our results.

REFERENCES

Zhenyu Bao, Guibiao Liao, Kaichen Zhou, Kanglin Liu, Qing Li, and Guoping Qiu. Loopsparsegs: Loop based sparse-view friendly gaussian splatting. *IEEE TIP*, 2025.

Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*, 2021.

Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *CVPR*, 2022.

Anpei Chen, Zexiang Xu, Fuqiang Zhao, Xiaoshuai Zhang, Fanbo Xiang, Jingyi Yu, and Hao Su. Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 14104–14113, 2021.

Kangle Deng, Andrew Liu, Jun-Yan Zhu, and Deva Ramanan. Depth-supervised nerf: Fewer views and faster training for free. In *CVPR*, 2022.

Liang Han, Junsheng Zhou, Yu-Shen Liu, and Zhizhong Han. Binocular-guided 3d gaussian splatting with view consistency for sparse view synthesis. *Advances in Neural Information Processing Systems*, 37:68595–68621, 2024.

Ajay Jain, Matthew Tancik, and Pieter Abbeel. Putting nerf on a diet: Semantically consistent few-shot view synthesis. In *ICCV*, 2021.

Mohammad Mahdi Johari, Yann Lepoittevin, and François Fleuret. Geonerf: Generalizing nerf with geometry priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18365–18375, 2022.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):139:1–139:14, 2023.

Hanyang Kong, Xingyi Yang, and Xinchao Wang. Generative sparse-view gaussian splatting. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 26745–26755, 2025.

Joo Chan Lee, Daniel Rho, Xiangyu Sun, Jong Hwan Ko, and Eunbyung Park. Compact 3d gaussian representation for radiance field. In *CVPR*, 2024.

Jiahe Li, Jiawei Zhang, Xiao Bai, Jin Zheng, Xin Ning, Jun Zhou, and Lin Gu. Dngaussian: Optimizing sparse-view 3d gaussian radiance fields with global-local depth normalization. In *CVPR*, 2024.

Zhaoliang Liu, Jinhe Su, Guorong Cai, Yidong Chen, Binghui Zeng, and Zongyue Wang. Georgs: Geometric regularization for real-time novel view synthesis from sparse inputs. *IEEE Transactions on Circuits and Systems for Video Technology*, 34(12):13113–13126, 2024.

Ben Mildenhall, Pratul P Srinivasan, Rodrigo Ortiz-Cayon, Nima Khademi Kalantari, Ravi Ramamoorthi, Ren Ng, and Abhishek Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ToG*, 2019.

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*, 2021.

Michael Niemeyer, Jonathan T Barron, Ben Mildenhall, Mehdi SM Sajjadi, Andreas Geiger, and Noha Radwan. Regnerf: Regularizing neural radiance fields for view synthesis from sparse inputs. In *CVPR*, 2022.

Avinash Paliwal, Wei Ye, Jinhui Xiong, Dmytro Kotovenko, Rakesh Ranjan, Vikas Chandra, and Nima Khademi Kalantari. Coherentgs: Sparse novel view synthesis with coherent 3d gaussians. In *European Conference on Computer Vision*, pp. 19–37. Springer, 2024.

Hyunwoo Park, Gun Ryu, and Wonjun Kim. Dropgaussian: Structural regularization for sparse-view gaussian splatting. In *CVPR*, 2025.

Barbara Roessle, Jonathan T Barron, Ben Mildenhall, Pratul P Srinivasan, and Matthias Nießner. Dense depth priors for neural radiance fields from sparse input views. In *CVPR*, 2022.

Meixi Song, Xin Lin, Dizhe Zhang, Haodong Li, Xiangtai Li, Bo Du, and Lu Qi. D^2GS: Depth-and-density guided gaussian splatting for stable and accurate sparse-view reconstruction. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=7yvz93kBw9.

Prune Truong, Marie-Julie Rakotosaona, Fabian Manhardt, and Federico Tombari. Sparf: Neural radiance fields from sparse and noisy poses. In *CVPR*, 2023.

Guangcong Wang, Zhaoxi Chen, Chen Change Loy, and Ziwei Liu. Sparsenerf: Distilling depth ranking for few-shot novel view synthesis. In *ICCV*, 2023.

Qianqian Wang, Zhicheng Wang, Kyle Genova, Pratul P. Srinivasan, Howard Zhou, Jonathan T. Barron, Ricardo Martin-Brualla, Noah Snavely, and Thomas A. Funkhouser. Ibrnet: Learning multi-view image-based rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4690–4699, 2021.

Qisen Wang, Yifan Zhao, Jiawei Ma, and Jia Li. How to use diffusion priors under sparse views? *Advances in Neural Information Processing Systems*, 37:30394–30424, 2024.

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.

Yexing Xu, Longguang Wang, Minglin Chen, Sheng Ao, Li Li, and Yulan Guo. Dropoutgs: Dropping out gaussians for better sparse-view rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2025.

Jiawei Yang, Marco Pavone, and Yue Wang. Freenerf: Improving few-shot neural rendering with free frequency regularization. In *CVPR*, 2023.

Zongxin Ye, Wenyu Li, Sidun Liu, Peng Qiao, and Yong Dou. Absgs: Recovering fine details in 3d gaussian splatting. In *ACM MM*, 2024.

Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4578–4587, 2021.

Jiawei Zhang, Jiahe Li, Xiaohan Yu, Lei Huang, Lin Gu, Jin Zheng, and Xiao Bai. Cor-gs: sparse-view 3d gaussian splatting via co-regularization. In *ECCV*, 2024a.

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*, pp. 586–595, 2018.

Zheng Zhang, Wenbo Hu, Yixing Lao, Tong He, and Hengshuang Zhao. Pixel-gs: Density control with pixel-aware gradient for 3d gaussian splatting. In *ECCV*, 2024b.

Zehao Zhu, Zhiwen Fan, Yifan Jiang, and Zhangyang Wang. Fsgs: Real-time few-shot view synthesis using gaussian splatting. In *ECCV*, 2024.

A Diagnostic Protocols for Primitive-Level Reconstruction Constraints

This appendix provides the measurement and aggregation procedures for Section 3, "What Constrains a Gaussian?" We describe how rendering support is measured, how individual primitives are perturbed, and how the resulting photometric responses are summarized within and across scenes.

A.1 Diagnostic Setup and Constraint Scores

All diagnostics are performed on vanilla 3DGS models. The model parameters are frozen during measurement, except for the parameters explicitly varied in each perturbation probe. No optimization or ConstraintGS attenuation is performed as part of the diagnostic procedure. Figure 1 uses the eight LLFF scenes with three training views and an image downsampling factor of eight. Figure 2 uses the horns scene under the same LLFF configuration. Figure 3 covers eight LLFF scenes, seven Mip-NeRF 360 scenes, and eight Blender scenes, with three, twelve, and eight training views per scene, respectively. The diagnostics use checkpoints at iteration 10,000.

Figures 2 and 3 use the self-opacity-normalized constraint score defined in Section 4.1. For primitive i and training view t, let w_{it} denote effective rendering support, o_i its activated opacity, and $\mathbf{r}_{it}$ the unit direction from the camera center to the primitive center. The descriptor is

$$M_i = \sum_t w_{it}\big(I - \mathbf{r}_{it}\mathbf{r}_{it}^\top\big), \qquad N_i = \frac{o_{\mathrm{ref}}}{o_i} M_i, \qquad c_i = \log\det\big(N_i + 10^{-8} I\big). \tag{10}$$

This score combines support magnitude and directional coverage.

The reference opacity is determined separately for each scene and training seed. At iteration 8,000, after the host Adam update and gradient clearing, the active set and reference opacity are

$$\mathcal{A}_{8000} = \{i \mid o_i > 0.01,\ \text{valid_view_count}_i > 0\}, \qquad o_{\text{ref}} = \text{median}_{i \in \mathcal{A}_{8000}}\, o_i. \tag{11}$$

In ConstraintGS training, this value is recorded before the first opacity attenuation. It remains fixed during subsequent score refreshes. Offline diagnostics read the reference saved for the corresponding run and evaluate o_i and M_i at the diagnostic checkpoint.

A.2 Effective Support Statistics

Per-view support. The support w_{it} is derived from the alpha-compositing coefficient $T_{it}a_{it}$. At a valid projected primitive center, we estimate this coefficient by replaying the front-to-back compositing order. If the center is invalid but the rasterizer accepts contributions elsewhere in the projected footprint, we use the largest accepted $T_{it}a_{it}$ within that footprint. A view with neither form of support contributes zero.

Unit-opacity support. Figure 1 reports

$$s_i = \sum_t w_{it}, \qquad s_i^{\text{unit}} = \frac{s_i}{o_i}. \tag{12}$$

The division is applied to support measured at the frozen model state; we do not set the opacity to one and render the model again. No additional constant is added to the denominator, and primitives with nonfinite or nonpositive activated opacity are excluded. This unit-opacity diagnostic uses a reference scale of one, whereas the scores in Eq. 10 use the saved scene-specific reference. Unit-opacity support retains the effects of foreground transmittance, the projected Gaussian profile, and rasterization rules.

Histograms and cumulative curves. The histograms contain finite, strictly positive unit-opacity support values. We take their base-10 logarithms and use 24 common bins across the eight scenes. Each bin is normalized by the number of positive-support primitives in its scene. Zero-support primitives are omitted from the logarithmic histograms but retained in the cumulative curves.

For scene s, let $u_{s,(1)} \le \cdots \le u_{s,(N_s)}$ be the sorted unit-opacity support values, including zeros. We define

$$F_s(0) = 0, \qquad F_s\left(\frac{k}{N_s}\right) = \frac{\sum_{j=1}^{k} u_{s,(j)}}{\sum_{j=1}^{N_s} u_{s,(j)}}, \quad k = 1, \ldots, N_s. \tag{13}$$

We linearly interpolate between these points, evaluate each curve on a common grid of 1,001 points over $[0, 1]$, and take the pointwise median across scenes. Each scene receives equal weight. The reported 19.9% is

$$\text{median}_s\, F_s(0.5) \approx 0.199. \tag{14}$$

It therefore summarizes the support share of the least-supported half of the primitives within each scene, rather than a ranking obtained by pooling primitives from all scenes.

A.3 Local Photometric Response

Perturbation coordinates. Figures 2 and 3 use the same perturbation definitions and photometric loss. For each probed primitive, let $\mathbf{d}_i$ be the unit ray from the selected training camera to its unperturbed center. Let $\ell_i = (\sigma_{i,x} + \sigma_{i,y} + \sigma_{i,z})/3$ be the mean of its three activated scale parameters. We perturb the center and opacity logit as

$$\boldsymbol{\mu}_i(\delta_z) = \boldsymbol{\mu}_i + \delta_z \ell_i \mathbf{d}_i, \qquad o_i(\delta_\eta) = \text{sigmoid}(\eta_i + \delta_\eta), \tag{15}$$

where η_i is the unperturbed opacity logit. Thus, $\delta_z = 0.1$ corresponds to a displacement of $0.1\ell_i$, while δ_η is measured in logit units. The ray and scale used to define these coordinates are fixed at the unperturbed state. All other primitives, as well as the probed primitive's scale, rotation, and appearance parameters, remain fixed. Each perturbed state is rendered through the standard rasterizer.

Diagnostic loss. Let $\widehat{I}_t(\delta_z, \delta_\eta)$ denote the perturbed rendering and I_t the training image. The loss is averaged over all training views $\mathcal{T}$:

$$L_i(\delta_z, \delta_\eta) = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \Big[0.8\, \mathcal{L}_1\big(\widehat{I}_t(\delta_z, \delta_\eta), I_t\big) + 0.2\big(1 - \mathrm{SSIM}\big(\widehat{I}_t(\delta_z, \delta_\eta), I_t\big)\big) \Big]. \tag{16}$$

Both terms are evaluated on the full images. Figure 2 displays the loss change relative to the unperturbed state,

$$\Delta L_i(\delta_z, \delta_\eta) = L_i(\delta_z, \delta_\eta) - L_i(0, 0). \tag{17}$$

Response magnitude. For the population analysis in Figure 3, we estimate a symmetric 2×2 second-order response matrix using nine loss evaluations: the center, four axis neighbors, and four corners. With $a = 0.10$ and $b = 0.25$, the entries are

$$H_{zz} = \frac{L_i(a, 0) - 2L_i(0, 0) + L_i(-a, 0)}{a^2}, \tag{18}$$

$$H_{\eta\eta} = \frac{L_i(0, b) - 2L_i(0, 0) + L_i(0, -b)}{b^2}, \tag{19}$$

$$H_{z\eta} = H_{\eta z} = \frac{L_i(a, b) - L_i(a, -b) - L_i(-a, b) + L_i(-a, -b)}{4ab}. \tag{20}$$

We define the local response magnitude as

$$H_i = \begin{bmatrix} H_{zz} & H_{z\eta} \\ H_{\eta z} & H_{\eta\eta} \end{bmatrix}, \qquad m_i = \sigma_1(H_i) + \sigma_2(H_i), \tag{21}$$

where σ_1 and σ_2 are singular values. Since H_i is symmetric, m_i also equals the sum of the absolute values of its eigenvalues; positive definiteness is not required. This statistic measures the overall second-order response in the specified depth–opacity coordinates, and its scale depends on the perturbation parameterization.

A.4 Qualitative Examples and Score Visualization

Figure 2 presents two primitives selected according to their constraint scores in the horns scene. The low-score primitive A and high-score primitive B have checkpoint indices 33667 and 1681, respectively. Their locations are marked in the rendered view. Both response surfaces use $\delta_z \in [-0.1, 0.1]$ and $\delta_\eta \in [-0.25, 0.25]$, with shared axis ranges and a common color scale.

To render the score heatmap, we map primitive scores to colors using the 5th and 95th percentiles of the visible-sample scores as reference endpoints. The colormap progresses from black through dark red, red, orange, and bright yellow to pale yellow, so higher scores appear brighter. The assigned colors are rendered with the native 3DGS rasterizer, retaining the checkpoint's Gaussian scales, rotations, and opacities, together with depth ordering and alpha compositing. The heatmap is rendered directly from the colored Gaussian representation without blending it with the RGB image. These two examples illustrate local response differences; the population analysis below assesses the association across scenes.

A.5 Sampling and Aggregation Across Scenes

Sampling and score groups. For Figure 3, a primitive is eligible if its activated opacity exceeds 0.01, it has at least one valid support view, and a reference direction is available for the depth probe. We sample 1,024 eligible primitives uniformly without replacement in each scene. Sampling does not depend on the measured response. Within each scene, the sampled primitives are sorted by c_i and partitioned into ten groups of approximately equal size, from D1 (lowest scores) to D10 (highest scores). Ties are resolved by primitive index. Each group contains 102 or 103 primitives. These are score deciles of the sampled set, rather than quantile boundaries computed from the entire scene population.

Within-scene normalization. Let $\mathcal{S}_s$ denote the sampled set in scene s. Before aggregating the responses by score group, we compute

$$b_s = \mathrm{median}_{i \in \mathcal{S}_s} m_i, \qquad \widetilde{m}_i = \frac{m_i}{b_s}. \tag{22}$$

The same denominator is used for all sampled primitives in a scene. It is computed from the full sampled set, not separately within each score group. This normalization expresses each response relative to the typical response in its scene.

Scene and dataset summaries. For score group $\mathcal{D}_{s,k}$ in scene s, we compute

$$q_{s,k} = \mathrm{median}_{i \in \mathcal{D}_{s,k}} \widetilde{m}_i. \tag{23}$$

The thin curves in Figure 3 show $q_{s,k}$ for individual scenes. Within each dataset, the highlighted curve is the median of $q_{s,k}$ across scenes at each group k, and the shaded band spans the corresponding 25th to 75th percentiles. Scenes are weighted equally; primitives are not pooled across scenes. The medians and quartiles are computed in linear response space before display on a base-10 logarithmic scale.

The shaded bands describe variation across scenes, rather than confidence intervals or variability across training seeds. Since responses are normalized within each scene, the panels characterize relative response patterns and do not compare absolute photometric sensitivity between datasets.

B Additional Details of ConstraintGS

This appendix supplements Section 4 with the geometric interpretation and properties of the cross-view constraint descriptor, the derivation of the bounded opacity allocation, and the training procedure. We use the notation of Sections 4.1 and 4.2 throughout.

B.1 Geometric Interpretation and Spectral Properties

For a unit viewing direction $\mathbf{r}_{it}$, the matrix $\mathbf{P}_{it} = \mathbf{I} - \mathbf{r}_{it}\mathbf{r}_{it}^\top$ is the orthogonal projector onto the plane perpendicular to that direction. It satisfies $\mathbf{P}_{it}^\top = \mathbf{P}_{it}$, $\mathbf{P}_{it}^2 = \mathbf{P}_{it}$, and $\mathbf{P}_{it}\mathbf{r}_{it} = \mathbf{0}$. A single view therefore contributes transverse support while leaving its ray direction unsupported by this geometric descriptor. After weighting and aggregation, the normalized matrix satisfies

$$\mathbf{v}^\top \mathbf{N}_i \mathbf{v} = \sum_t \bar{w}_{it} \, \|\mathbf{P}_{it}\mathbf{v}\|_2^2 \geq 0, \qquad \mathbf{v} \in \mathbb{R}^3. \tag{24}$$

Thus, $\mathbf{N}_i$ is positive semidefinite. Nearly parallel supporting rays leave a common direction weakly supported, whereas complementary rays can provide support along all three spatial directions. Here, directional complementarity concerns the ray axes: reversing a ray leaves its projector unchanged.

Support ellipsoid. Figure 7 visualizes the spectrum through the ellipsoid

$$\mathcal{E}_i = \left\{ (\mathbf{N}_i + \epsilon_m \mathbf{I})^{1/2} \mathbf{v} \;\middle|\; \|\mathbf{v}\|_2 \leq 1 \right\}. \tag{25}$$

Let λ_{ik} denote the eigenvalues of $\mathbf{N}_i$. The semiaxis lengths are $\sqrt{\lambda_{ik} + \epsilon_m}$, so a short axis indicates weak support along the corresponding eigenvector. The ellipsoid volume relates directly to the score:

$$\mathrm{Vol}(\mathcal{E}_i) = \frac{4\pi}{3} \prod_{k=1}^{3} \sqrt{\lambda_{ik} + \epsilon_m}, \qquad c_i = 2 \log\left(\frac{\mathrm{Vol}(\mathcal{E}_i)}{4\pi/3} \right). \tag{26}$$

This construction gives a geometric interpretation of the scalar score: it measures the volume of regularized directional support in logarithmic form.

Dependence on support magnitude. Using the decomposition in Eq. 6, we have $\mathbf{N}_i = \bar{s}_i(\mathbf{I} - \mathbf{Q}_i)$, where $\bar{s}_i = \sum_t \bar{w}_{it} > 0$. Since $\mathbf{Q}_i$ is positive semidefinite with unit trace, its eigenvalues satisfy

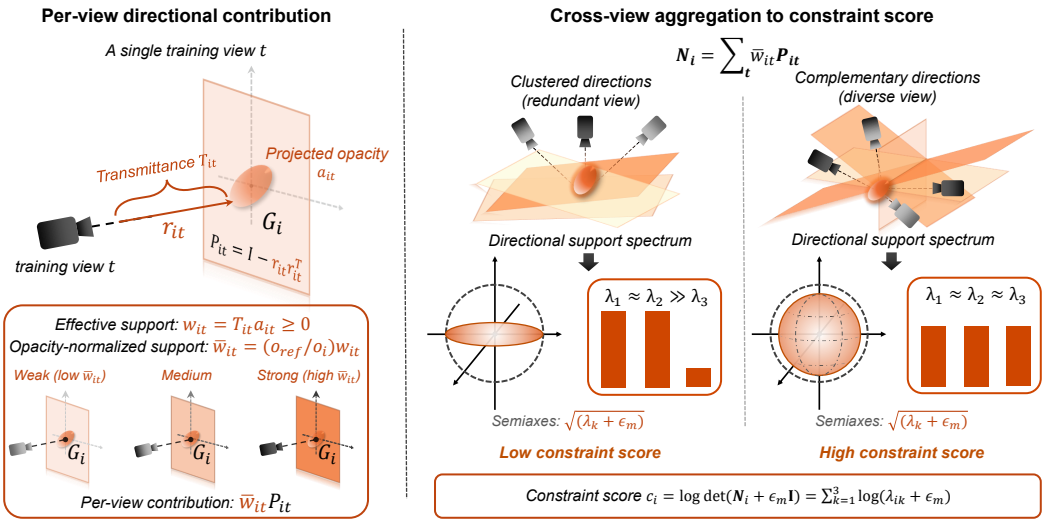

Figure 7: Geometric interpretation of the constraint score. Left: a training view contributes the transverse projector $\mathbf{P}_{it} = \mathbf{I} - \mathbf{r}_{it}\mathbf{r}_{it}^\top$, weighted by opacity-normalized rendering support $\bar{w}_{it} = (o_{\text{ref}}/o_i)w_{it}$. The coefficient $w_{it} = T_{it}a_{it}$ combines foreground transmittance T_{it} and projected alpha a_{it} at the selected rendering sample. Right: cross-view aggregation forms $\mathbf{N}_i = \sum_t \bar{w}_{it}\mathbf{P}_{it}$. The two configurations illustrate matched total normalized support, $\sum_t \bar{w}_{it}$, with spectra shown on a common scale. Clustered ray directions leave one eigenvalue small, while complementary directions produce a more balanced spectrum and a higher log-determinant score. The ellipsoids visualize $\mathbf{N}_i + \epsilon_m\mathbf{I}$, with semiaxes $\sqrt{\lambda_{ik} + \epsilon_m}$ along its eigenvectors; they represent directional support rather than Gaussian covariance or estimation uncertainty. Here, λ_{ik} are the eigenvalues of $\mathbf{N}_i$, and the primitive index i is omitted from the spectral labels in the figure.

$0 \leq q_{ik} \leq 1$ and $\sum_k q_{ik} = 1$. In a shared eigenbasis, $\lambda_{ik} = \bar{s}_i(1 - q_{ik})$. Holding the relative view weights and ray directions fixed gives

$$\frac{\partial c_i}{\partial \bar{s}_i} = \sum_{k=1}^{3} \frac{1 - q_{ik}}{\bar{s}_i(1 - q_{ik}) + \epsilon_m} > 0. \tag{27}$$

The inequality is strict because $\sum_k (1 - q_{ik}) = 2$. Consequently, increasing total support at a fixed directional configuration always increases the score.

Dependence on directional balance. At fixed $\bar{s}_i$, the trace is $\operatorname{tr}(\mathbf{N}_i) = 2\bar{s}_i$. Concavity of the logarithm gives

$$c_i = \sum_{k=1}^{3} \log(\lambda_{ik} + \epsilon_m) \leq 3 \log\left(\frac{2\bar{s}_i}{3} + \epsilon_m\right), \tag{28}$$

with equality if and only if $\lambda_{i1} = \lambda_{i2} = \lambda_{i3} = 2\bar{s}_i/3$, equivalently $\mathbf{Q}_i = \mathbf{I}/3$. Equalizing two unequal eigenvalues while preserving their sum increases the score. For example, perfectly aligned ray axes yield the spectrum $(\bar{s}_i, \bar{s}_i, 0)$, whereas isotropic directional support yields $(2\bar{s}_i/3, 2\bar{s}_i/3, 2\bar{s}_i/3)$. These configurations have identical trace but different scores. This explains the comparison in Figure 7 and why the trace-based support-only variant in Table 3 cannot distinguish directional coverage at a fixed support magnitude. If all weights vanish, $\mathbf{N}_i = \mathbf{0}$ and the regularized score remains finite at $3 \log \epsilon_m$.

B.2 Local Invariance to the Primitive's Own Opacity

The normalization in Eq. 4 removes the direct multiplicative dependence on the primitive's own opacity under the local conditions stated in Section 4.1. Fix the geometry, the other primitives,

the compositing order, and the accepted rendering samples. On an unsaturated alpha branch, $a_{it}(\mathbf{x}) = o_i g_{it}(\mathbf{x})$. Because foreground transmittance does not include the current primitive's alpha, the center-sampled coefficient factors as $w_{it} = o_i \xi_{it}$, where ξ_{it} is independent of o_i. For the footprint fallback, the same factorization follows from

$$\max_{\mathbf{x}\in\mathcal{S}_{it}} T_{it}(\mathbf{x}) o_i g_{it}(\mathbf{x}) = o_i \max_{\mathbf{x}\in\mathcal{S}_{it}} T_{it}(\mathbf{x}) g_{it}(\mathbf{x}), \tag{29}$$

where $\mathcal{S}_{it}$ is the fixed set of accepted footprint samples. It follows that $\bar{w}_{it} = o_{\mathrm{ref}} \xi_{it}$, making $\mathbf{N}_i$ and c_i locally independent of o_i. The reference o_{ref} is selected once from the eligible primitives at activation and then held fixed.

This property prevents the direct reduction of the descriptor caused solely by multiplying the primitive's opacity by an attenuation factor. It remains conditional on the stated rendering configuration: changes in alpha saturation, sample acceptance, geometry, or foreground occlusion can change the descriptor. In particular, normalization preserves the effect of occlusion by other primitives.

B.3 Bounded Allocation Under a Mean Attenuation Budget

For a nonempty eligible set $\mathcal{A}$ with $n = |\mathcal{A}|$, Eq. 7 maps lower scores to larger pressures. The initial allocation $u_i^{(0)} = bp_i/\bar{p}$ has mean b, but individual entries may exceed $u_{\max}$. The allocation in Eq. 8 solves

$$\min_{\mathbf{u}\in\mathbb{R}^n} \frac{1}{2}\sum_{i\in\mathcal{A}} (u_i - u_i^{(0)})^2 \quad \text{subject to} \quad 0 \le u_i \le u_{\max}, \qquad \sum_{i\in\mathcal{A}} u_i = nb. \tag{30}$$

The feasible set is nonempty because the uniform allocation $u_i = b$ is feasible. Strict convexity of the objective gives a unique optimal allocation.

Clipped-shift solution. Introduce a multiplier β for the sum constraint and nonnegative multipliers ℓ_i and h_i for the lower and upper bounds. Stationarity and complementary slackness give

$$u_i - u_i^{(0)} + \beta - \ell_i + h_i = 0, \qquad \ell_i u_i = 0, \qquad h_i(u_i - u_{\max}) = 0. \tag{31}$$

An interior entry therefore satisfies $u_i = u_i^{(0)} - \beta$, while entries outside the interval are clamped to the corresponding bound. Hence,

$$u_i = \mathrm{clip}(u_i^{(0)} - \beta, 0, u_{\max}), \qquad F(\beta) = \sum_{i\in\mathcal{A}} \mathrm{clip}(u_i^{(0)} - \beta, 0, u_{\max}) - nb = 0. \tag{32}$$

The function F is continuous and nonincreasing. A root can be obtained by bisection on $[\min_i(u_i^{(0)} - u_{\max}), \max_i u_i^{(0)}]$, whose endpoints produce total allocations $nu_{\max}$ and zero, respectively. The multiplier need not be unique on a flat segment, but the resulting allocation is unique.

Ordering and budget interpretation. The common shift and monotone clipping preserve the pressure ordering: $p_i \ge p_j$ implies $u_i \ge u_j$, with possible ties at the bounds. Equal pressures give the uniform allocation $u_i = b$. Directly clipping $\mathbf{u}^{(0)}$ would reduce its mean whenever the upper bound is exceeded; the shift restores the prescribed budget. Under Eq. 9,

$$\frac{1}{n}\sum_{i\in\mathcal{A}} \frac{o_i^{\mathrm{host}} - o_i^{+}}{o_i^{\mathrm{host}}} = \frac{1}{n}\sum_{i\in\mathcal{A}} u_i = b. \tag{33}$$

Thus, b controls the mean relative opacity reduction over eligible primitives. The absolute removal $\sum_i u_i o_i^{\mathrm{host}}$ also depends on their opacities. The projection supplies the nearest feasible allocation to the pressure-based proposal.

B.4 Integration into Host Training

ConstraintGS applies its opacity update after the host optimizer step. At activation, it records the median eligible opacity as o_{ref} and initializes the score and pressure cache. Subsequent refreshes recompute rendering support, normalized descriptors, and scene-relative pressures without gradient tracking. A refresh is performed every K iterations and after topology changes or opacity resets.

Algorithm 1 ConstraintGS update after a host optimizer step

Require: Host-updated Gaussian parameters, cached pressures, frozen reference o_{ref} if initialized, budget b, cap $u_{\max}$.

if regularization is not yet active **then**
  Leave the host-updated parameters unchanged.
else
  Form the current eligible set $\mathcal{A}$.
  if $\mathcal{A} \neq \varnothing$ **then**
    if o_{ref} is not initialized **then**
      Set $o_{\mathrm{ref}} \leftarrow \mathrm{median}_{i \in \mathcal{A}}\, o_i^{\mathrm{host}}$ and request a refresh.
    end if
    if a refresh is requested or due, or topology/opacity reset has invalidated the cache **then**
      Recompute w_{it}, $\mathbf{N}_i$, and c_i without gradient tracking using Section 4.1.
      Update cached p_i using Eq. 7.
    end if
    Compute $u_i^{(0)} = bp_i/\bar{p}$ on the current $\mathcal{A}$.
    Find the bounded allocation using Eq. 8.
    Set $o_i^{+} = (1 - u_i)o_i^{\mathrm{host}}$ for $i \in \mathcal{A}$; retain other opacities.
    Write the updated opacities back to the host parameterization, retaining optimizer moment states.
  end if
end if

Between refreshes, the cached pressures are reused, while the bounded allocation is recomputed over the current eligible set. Algorithm 1 summarizes one active update.

The shared configuration uses $b = 0.015$, $u_{\max} = 0.06$, activation after 80% of training, and $K = 100$. Eligibility requires activated opacity above $o_{\min} = 0.01$ and at least one valid support view; the descriptor uses $\epsilon_m = 10^{-8}$. Primitives outside $\mathcal{A}$ receive no attenuation, and an empty eligible set skips the update. The score and pressure computations do not enter the host's gradient graph.

C Further Analysis of Opacity Allocation

The ablation in Table 3 examines whether reconstruction benefits arise from opacity attenuation itself or from how attenuation is distributed across primitives. Comparing vanilla 3DGS with uniform attenuation isolates the effect of applying a prescribed opacity reduction. Comparing uniform allocation with the full method then evaluates the benefit of distributing the same mean budget according to the constraint descriptor. All attenuation variants use the same budget, cap, and schedule.

We further evaluate a shuffled-allocation control that randomly reassigns descriptor-derived attenuation across eligible primitives while retaining the same mean budget, per-primitive cap, and schedule. As shown in Table 5, the full method improves PSNR by 0.63 dB and SSIM by 0.040 over shuffled allocation, while reducing LPIPS by 0.009. These results support the reconstruction benefit of matching attenuation strength to each primitive's constraint descriptor.

Table 5: Allocation comparison on LLFF with 3 training views at 1/8 resolution, using 3DGS as the host. All variants share the same mean attenuation budget, per-primitive cap, and schedule. Bold indicates the best result.

Allocation	PSNR(↑)	SSIM(↑)	LPIPS(↓)
Uniform	20.40	0.698	0.203
Shuffled	20.42	0.685	0.200
Full ConstraintGS	**21.05**	**0.725**	**0.191**

The support-only and direction-only variants examine the two ingredients in Eq. 6. Support-only allocation discards directional coverage, assigning the same score to primitives with equal total

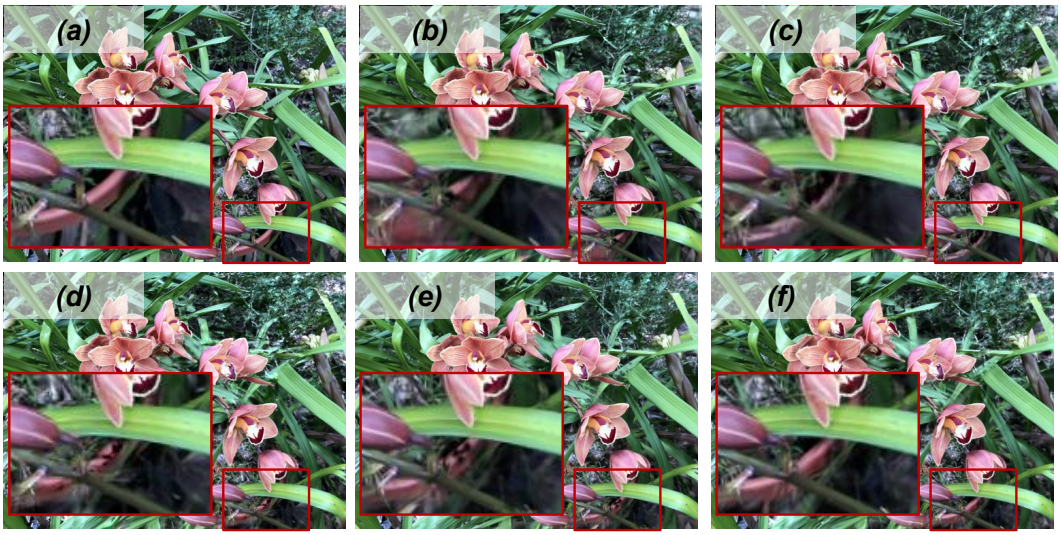

Figure 8: Qualitative allocation ablation. (a) Ground truth. (b) Uniform allocation. (c) Without self-opacity normalization. (d) Support-only. (e) Direction-only. (f) Full method. Panels (b)–(f) correspond to rows 2–6 of Table 3. Insets show identical crops at the same magnification.

normalized support regardless of their viewing directions. Direction-only allocation removes total support magnitude while retaining relative view weights and ray geometry, assigning the same score to primitives with the same weighted directional distribution even when their total support differs. Their comparisons with the full method therefore test the additional value of directional coverage and support magnitude, respectively. Direction-only allocation achieves higher PSNR than support-only allocation, but lower SSIM, indicating that neither simplified descriptor consistently outperforms the other across metrics. The full descriptor achieves the best PSNR, SSIM, and LPIPS, supporting the complementary value of the two ingredients.

The normalization ablation addresses a separate question: whether the primitive's own opacity should directly scale the descriptor used to determine its attenuation. We remove the normalization in Eq. 4 while retaining both support magnitude and directional coverage. As discussed in Section B.2, normalization removes this direct dependence under fixed local rendering conditions. Its removal decreases PSNR by 0.35 dB and SSIM by 0.017, while increasing LPIPS by 0.027. These results demonstrate the reconstruction benefit of self-opacity normalization.

Figure 8 complements the quantitative ablation. In the highlighted region, the full method better preserves the continuity and shape of the pink flower stem and nearby thin branches, while the ablated variants exhibit varying degrees of blur and structural artifacts.

D Additional Experiments

D.1 Additional Qualitative Results

Figures 9 and 10 present additional qualitative comparisons across host methods and datasets. Each comparison includes the ground truth, the original host, and the host augmented with ConstraintGS. Corresponding renderings use identical test viewpoints and crop regions, enabling direct comparison of structural fidelity, fine-detail preservation, and rendering artifacts.

LLFF. Figure 9 demonstrates improved structural completeness and reduced floating artifacts across four hosts. For 3DGS, ConstraintGS improves the structural completeness of the distant column and ceiling light in *fern*, while suppressing floaters near the shelving in *horns*. For FSGS, it better preserves the appearance details of the whiteboard drawing in *room* and the continuity of thin structures, including the pink stem and adjacent branches, in *orchids*. For CoR-GS, it reduces spurious

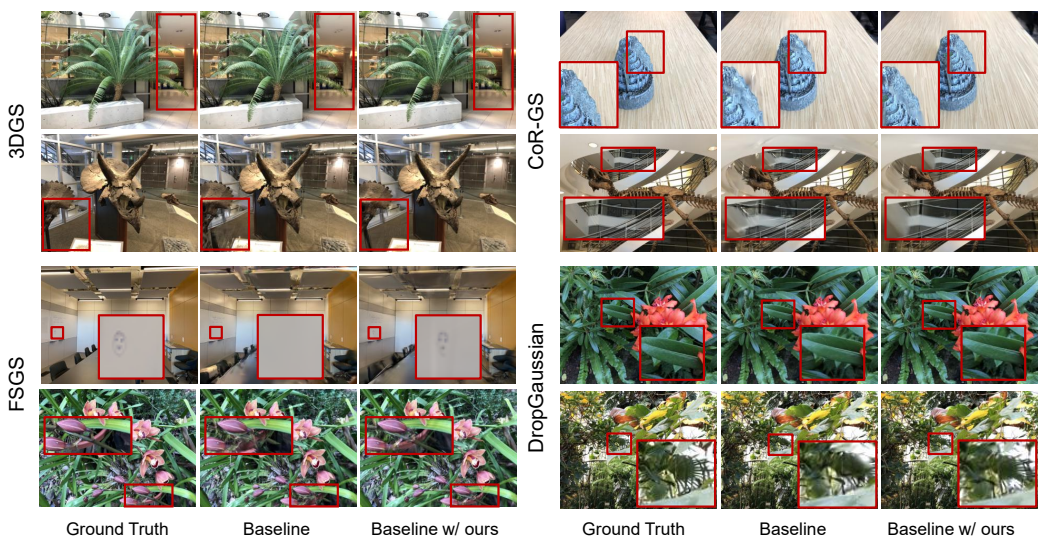

Figure 9: Additional qualitative comparisons on LLFF. The left block shows 3DGS on *fern* and *horns*, followed by FSGS on *room* and *orchids*. The right block shows CoR-GS on *fortress* and *trex*, followed by DropGaussian on *flower* and *leaves*. Within each block, columns show the ground truth, the original host, and the host with ConstraintGS. Colored boxes and enlarged crops highlight differences in structural completeness, fine details, and floating artifacts. Corresponding crops use the same region and magnification.

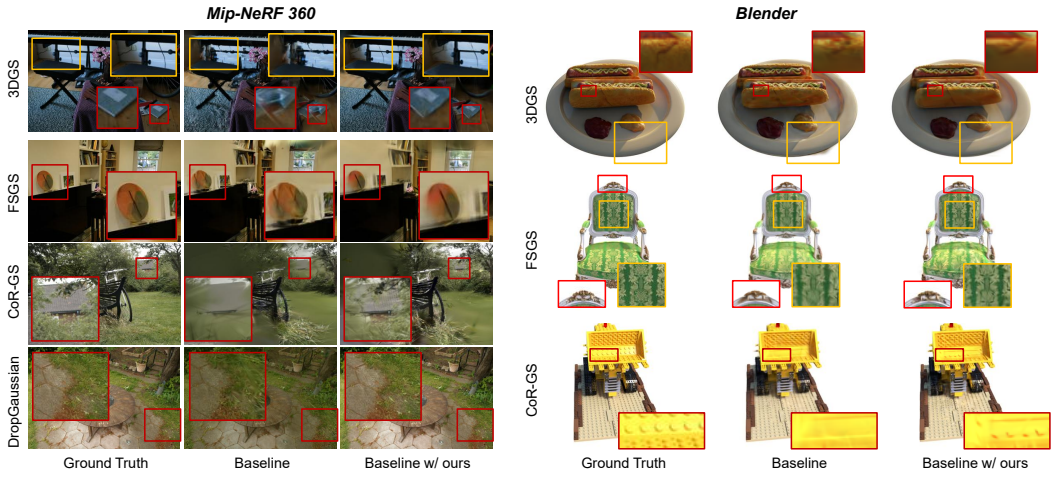

Figure 10: Additional qualitative comparisons on Mip-NeRF 360 and Blender. Left: comparisons for 3DGS, FSGS, CoR-GS, and DropGaussian on Mip-NeRF 360, from top to bottom. Right: comparisons for 3DGS, FSGS, and CoR-GS on the Blender *hotdog*, *chair*, and *lego* scenes, respectively. Within each panel, columns show the ground truth, the original host, and the host with ConstraintGS. Highlighted regions illustrate differences in object boundaries, distant structures, surface details, and rendering artifacts. Corresponding crops use the same region and magnification.

structures along the object boundary in *fortress* and improves the definition of distant railings while suppressing floaters in *trex*. For DropGaussian, the highlighted foliage in *flower* and *leaves* exhibits more faithful leaf contours and better separation of overlapping leaves.

Mip-NeRF 360. The left panel of Figure 10 highlights improvements in boundary fidelity, distant-structure reconstruction, and artifact suppression. For 3DGS, ConstraintGS produces a better-defined book boundary and reduces spurious foreground structures in the yellow-boxed region. For FSGS, it improves the contour fidelity of the clock face and reduces local shape distortion. For CoR-GS, it better resolves the distant building, whose structure is substantially blurred in the baseline rendering. For DropGaussian, it suppresses floating artifacts over the grass and reduces the overall luminance discrepancy relative to the ground truth.

Blender. The right panel of Figure 10 shows improvements in surface appearance and fine-detail reconstruction. For 3DGS, ConstraintGS reduces spurious appearance artifacts on the plate and better preserves the shape and definition of the hotdog's surface creases. For FSGS, it improves the definition of the chair's upper decorative structure and the fidelity of the backrest's upholstery texture. For CoR-GS, it better resolves the repeated raised details on the Lego bucket surface, which are blurred in the baseline rendering.

D.2 Computational Overhead

We evaluate the computational overhead of ConstraintGS using 3DGS as the host across all eight LLFF scenes. Both configurations run on a single NVIDIA GeForce RTX 4090 GPU with identical input views, image resolutions, and training iterations for each paired setting. Table 6 reports the scene-averaged total training time and peak GPU memory consumption, with peak memory measured separately for each run before averaging. Integrating ConstraintGS increases the average training time from 73.46 to 92.84 seconds (+26.4%) and the average peak GPU memory from 1088 to 1164 MB (+7.0%).

The additional computation includes periodic cross-view support estimation and score refreshes, together with bounded allocation and opacity attenuation after host optimizer steps. Scores and pressures are cached between refreshes to avoid recomputing the descriptor at every iteration. These operations are confined to training: ConstraintGS introduces no additional learned parameters or rendering operations at inference and preserves the native 3DGS rendering pipeline.

Table 6: Computational overhead on LLFF. Total training time and peak GPU memory consumption are averaged across all eight scenes. Measurements use a single NVIDIA GeForce RTX 4090 GPU under matched training settings. Relative increases are computed with respect to vanilla 3DGS.

Method	Training time (s)	Peak GPU memory (MB)
3DGS	73.46	1088
3DGS + ConstraintGS	92.84	1164
Relative increase	+26.4%	+7.0%