FACTORGRAD-GS: EXACT-PRIMAL MONTE CARLO STRUCTURAL BACKPROPAGATION FOR EFFICIENT 3D GAUSSIAN SPLATTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Efficient 3D Gaussian Splatting (3DGS) methods reduce Gaussian populations, rasterization work, or the number of optimizer updates, but retained updates commonly evaluate full red–green–blue (RGB) structural derivatives. FactorGrad-GS separates the exact structural dissimilarity (D-SSIM) value from the derivative used for optimization. It retains dense ℓ_1 supervision and samples D-SSIM derivatives across color channels and iterations, with inverse-probability weighting. A densification-aware schedule uses channel-only sampling during growth, restores full RGB after densification, and then enables temporal sampling. A selective fused-SSIM autograd path reuses one render and preserves the scalar objective supplied to a host gate. Matched SkipGS experiments show lower training wall time with small mean quality decreases. Additional-host comparisons identify schedule-dependent gains, while profiling and gradient measurements locate the saving in loss-side backward and document changes in optimization. These results support structural-derivative sampling as a complementary training acceleration mechanism. Project page: `https://factorgrad-gs.github.io/`

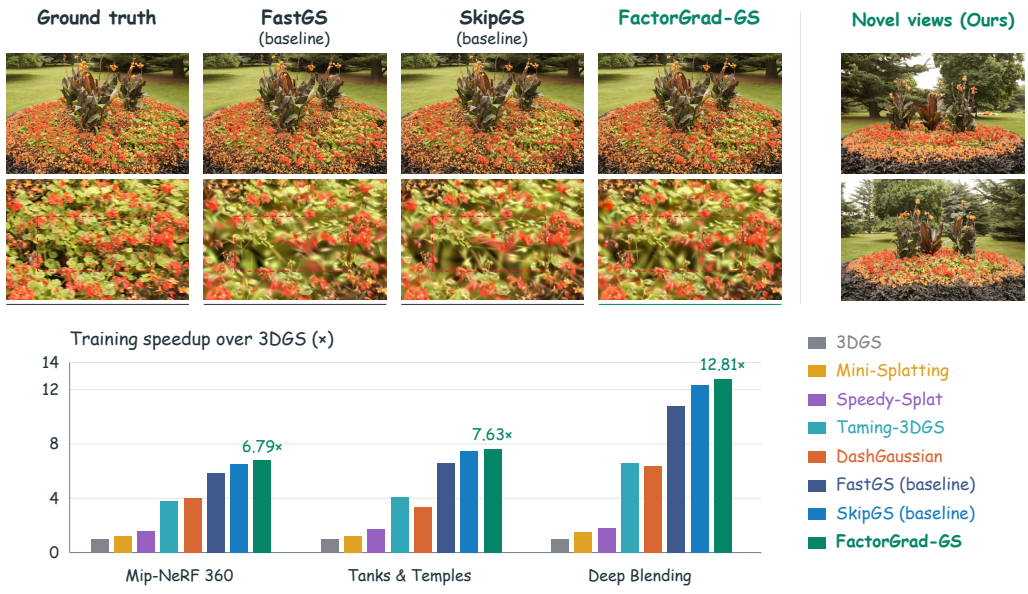

Figure 1: Flowers appearance and training efficiency. Top: aligned views/crops with red dashed detail boxes, and two novel views rendered by FactorGrad-GS from a separate visualization run. Bottom: measured training speedup over 3DGS for seven baselines and Ours, using the configurations in Table 1. Each group of bars represents one dataset; speedup is the 3DGS training time divided by the method's training time.

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.

1 INTRODUCTION

3D Gaussian Splatting (3DGS) combines an explicit scene representation with a differentiable rasterizer to achieve high-quality novel-view synthesis at interactive frame rates (Kerbl et al., 2023). Despite its rendering efficiency, optimizing 3DGS remains computationally demanding. Existing accelerators constrain Gaussian growth, prune rasterization workloads, or skip entire optimizer updates (Mallick et al., 2024; Chen et al., 2025; Ren et al., 2026; Hanson et al., 2025; Liu & Han, 2026; Dong et al., 2026; Li et al., 2026). We study a different source of backward cost: full-RGB structural derivatives within retained updates. Budgeting their temporal and channel components while preserving the exact forward objective offers a complementary route to acceleration.

Our approach is motivated by the observation that loss evaluation and gradient estimation serve fundamentally distinct roles. A training controller uses the scalar loss value to decide whether to accept an update, whereas the optimizer relies on the image-space adjoint to modify Gaussian parameters. Because the structural dissimilarity (D-SSIM) term is channel-separable, computing its exact RGB loss value does not inherently require calculating full-RGB structural gradients during every retained update. The training host can therefore maintain its exact forward objective while allocating a separate derivative budget to determine which structural components to differentiate. This decoupled design ensures that dense ℓ_1 supervision remains continuously active even when the structural derivative computation is selectively bypassed.

While randomized differentiation and component sampling provide a theoretical foundation for this decoupling (Rath et al., 2022; Wilkie et al., 2014; Oktay et al., 2021; Wang et al., 2024), adapting these techniques to 3DGS presents unique challenges due to the dynamically evolving scene representation. Specifically, stochastic image adjoints directly influence Adam momentum states and subsequent topological decisions including cloning, splitting, and pruning. Consequently, satisfying a marginal gradient identity at a frozen state does not guarantee an identical optimization trajectory over time. Estimator normalization and sampling schedules must therefore be assessed together with their end-to-end optimization effects.

We introduce FactorGrad-GS, a Monte Carlo backward path computing the exact forward loss while stochastically sampling the gradient. It evaluates the full-RGB D-SSIM value, estimating selected channel derivatives via reused forward statistics and inverse-probability weighting before merging them with the dense ℓ_1 adjoint. A densification-aware schedule uses full-RGB derivatives during warm-up and stabilization, channel sampling during active densification, and joint channel–temporal sampling in late refinement. Implemented as a fused-SSIM module, it requires a single render and operates behind existing host gates. Evaluations across three benchmarks (Barron et al., 2022; Knapitsch et al., 2017; Hedman et al., 2018) locate the saving primarily in the image-loss backward pass. Figure 1 shows the broader comparison to 3DGS; Section 4.3 isolates the module's incremental effect on the same SkipGS host.

Our contributions are:

- an exact-forward Monte Carlo gradient estimator that decouples the D-SSIM loss evaluation from its stochastic image-space adjoint, preserving dense ℓ_1 supervision and satisfying a frozen-state marginal gradient identity;
- a densification-aware, selective autograd module requiring only a single render to execute balanced channel and temporal sampling, which integrates seamlessly as a plug-and-play extension for existing whole-update gates;
- comprehensive empirical analyses, including matched host comparisons and component profiling, that validate our factorization strategies and pinpoint the exact sources of our efficiency and quality trade-offs within the training loop.

2 RELATED WORK

Efficient 3D Gaussian Splatting. Mini-Splatting and Taming-3DGS constrain Gaussian populations (Fang & Wang, 2024; Mallick et al., 2024); FastGS, LeGS, and DashGaussian revise their growth (Ren et al., 2026; Ning et al., 2026; Chen et al., 2025). Recent work replaces densification with dense-correspondence initialization (EDGS) or guides it with image structure (SAD-GS) and

gradient directions (GDAGS) (Kotovenko et al., 2026; Lyu et al., 2026; Zhou et al., 2026). AdamW-GS decouples optimizer updates and regularization, while 3DGS^2-TR uses diagonal curvature and trust regions (Ding et al., 2026; Hsiao et al., 2026). 3DGS-LM replaces Adam with a Levenberg–Marquardt solver and approximates the SSIM Jacobian by retaining only center-pixel contributions, enabling independent ray processing (Höllein et al., 2025). It evaluates the same three scene benchmarks, but changes the optimizer rather than sampling the host's structural backward. A polarized opacity prior promotes compact representations (Wang et al., 2026). Rendering improvements include sparse pixels and primitives, shorter Gaussian lists, and kernel optimization (Hanson et al., 2025; Liu & Han, 2026; Hahlbohm et al., 2026).

Sparse supervision and backward scheduling. TurboGS samples error-guided image tiles for structural supervision (Dong et al., 2026). S3IM instead changes the structural objective by forming stochastic nonlocal pixel groups for neural-field training (Xie et al., 2023); it does not preserve the host's exact D-SSIM scalar while sampling its backward derivative. The distinct Turbo-GS combines dilated rendering with convergence-aware Gaussian budgets and position–appearance-guided densification (Dhiman et al., 2026). Turbo-GS also evaluates the three scene benchmarks used here. SkipGS skips complete post-densification updates (Li et al., 2026). Laplacian frequency hierarchies reduce the active Gaussian set by progressively optimizing residual frequency bands (Yang et al., 2026). TruncGradGS replaces vanishing Gaussian derivative tails with linear surrogates to improve gradient propagation and reconstruction robustness (Morales et al., 2026). Its static experiments include Mip-NeRF 360; its focus is gradient propagation and reconstruction robustness, not reducing the structural-loss backward budget. FactorGrad-GS instead samples temporal and channel components of the D-SSIM adjoint within host-accepted updates. It preserves the exact RGB loss value and dense ℓ_1 supervision, without changing the host optimizer or rasterizer kernels.

Randomized derivatives. meProp retains top-k gradients (Sun et al., 2017). Randomized automatic differentiation and variance-controlled sampling trade computation for estimator variance (Oktay et al., 2021; Wang et al., 2024). Russian roulette reweights surviving light paths, and hero-wavelength methods sample spectral components (Rath et al., 2022; Wilkie et al., 2014). Autograd systems also prune inactive instructions or reconstruct reverse states from compact traces (Moses & Churavy, 2020; Vicini et al., 2021). We specialize sampling to channel-separable structural derivatives under a densification-aware budget.

3 FactorGrad-GS

At iteration t, the host renders view v_t as $\widehat{\mathbf{I}}_t = \mathcal{R}(\boldsymbol{\theta}_t, v_t)$ and compares it with the target $\mathbf{I}_t$. FactorGrad-GS computes the exact RGB objective but samples its D-SSIM derivative using channel sampling, temporal sampling, and a densification-aware budget (Figure 2). The host renderer, optimizer, and density-control rules are unchanged; sampled gradients can change their inputs.

3.1 Decoupling Deterministic Loss Evaluation and Stochastic Adjoints

For channels $\mathcal{H} = \{\mathrm{R}, \mathrm{G}, \mathrm{B}\}$, channel-separable SSIM gives the contributions and their full-RGB sum:

$$\mathcal{D}_{t,h} = \frac{1 - \mathrm{SSIM}(\widehat{\mathbf{I}}_t^h, \mathbf{I}_t^h)}{3}. \tag{1}$$

$$\mathcal{D}_t = \sum_{h \in \mathcal{H}} \mathcal{D}_{t,h}. \tag{2}$$

For image domain Ω_t and structural weight λ_{str}, the host objective is

$$\begin{cases} \mathcal{L}_t^{(1)} = \dfrac{\|\widehat{\mathbf{I}}_t - \mathbf{I}_t\|_1}{3|\Omega_t|}, \\ \mathcal{L}_t = (1 - \lambda_{\mathrm{str}})\mathcal{L}_t^{(1)} + \lambda_{\mathrm{str}}\mathcal{D}_t. \end{cases} \tag{3}$$

A gated host then decides whether to apply backward:

$$a_t = \mathcal{A}_t(v_t, \mathrm{sg}(\mathcal{L}_t), \mathcal{S}_{t-1}). \tag{4}$$

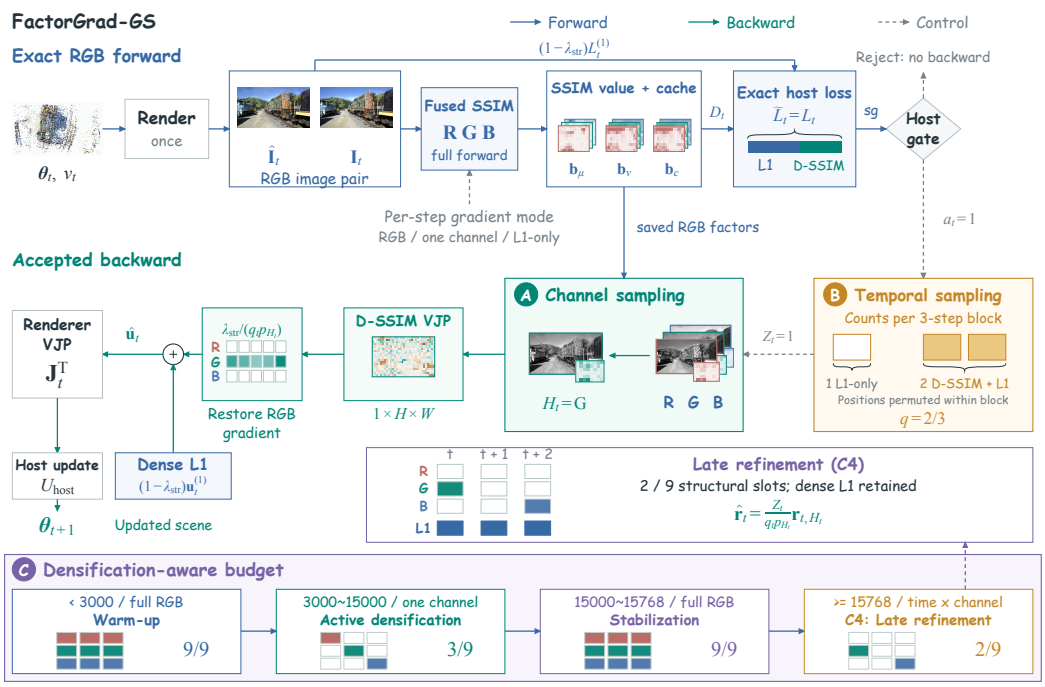

Figure 2: FactorGrad-GS. One render supplies the exact RGB loss and cached SSIM factors. Accepted backward runs right to left: A selects a channel for the vector–Jacobian product (VJP), restores RGB, and adds dense ℓ_1; B selects temporal positions, shown grouped but applied in shuffled order. C sets the budget; its C4 inset assumes all three host updates accept. Slot counts indicate structural work, not speedup. Cache factors and tensor tiles are detailed in Appendix D.2. Blue, teal, and dashed arrows denote values, adjoints, and control.

Here $a_t \in \{0, 1\}$, sg is stop-gradient, and $\mathcal{S}_{t-1}$ is the host gate state, including per-view exponential moving averages (EMAs) in SkipGS. Ungated hosts use $a_t = 1$.

Let H_t select a channel with marginal $p_h = 1/3$, and let Z_t retain structural backward with marginal $q_t > 0$. The selective autograd path has the same value and derivative as the surrogate

$$\widetilde{\mathcal{D}}_t = \mathrm{sg}(\mathcal{D}_t) + \frac{Z_t}{q_t p_{H_t}} \big[\mathcal{D}_{t,H_t} - \mathrm{sg}(\mathcal{D}_{t,H_t})\big]. \tag{5}$$

$$\widetilde{\mathcal{L}}_t = (1 - \lambda_{\mathrm{str}})\mathcal{L}_t^{(1)} + \lambda_{\mathrm{str}}\widetilde{\mathcal{D}}_t. \tag{6}$$

The scalar equals $\mathcal{L}_t$ for every draw at the current rendering, but gate decisions can differ across optimization trajectories.

Let $\mathbf{J}_t = \partial \mathrm{vec}(\widehat{\mathbf{I}}_t)/\partial \boldsymbol{\theta}_t \in \mathbb{R}^{3|\Omega_t| \times d}$, where d is the parameter count. The matrix $\mathbf{P}_h \in \{0, 1\}^{3|\Omega_t| \times |\Omega_t|}$ inserts one channel into RGB with zeros elsewhere. Define

$$\begin{cases} \mathbf{u}_t^{(1)} = \nabla_{\mathrm{vec}(\widehat{\mathbf{I}}_t)} \mathcal{L}_t^{(1)}, \\ \mathbf{r}_{t,h} = \mathbf{P}_h \nabla_{\mathrm{vec}(\widehat{\mathbf{I}}_t^h)} \mathcal{D}_{t,h}, \\ \widehat{\mathbf{r}}_t = \dfrac{Z_t}{q_t p_{H_t}} \mathbf{r}_{t,H_t}. \end{cases} \tag{7}$$

The host receives the parameter derivative

$$\widehat{\mathbf{g}}_t = a_t \mathbf{J}_t^\top \big[(1 - \lambda_{\mathrm{str}})\mathbf{u}_t^{(1)} + \lambda_{\mathrm{str}}\widehat{\mathbf{r}}_t\big]. \tag{8}$$

Dense ℓ_1 remains when $Z_t = 0$. Full-RGB stages instead use $\widetilde{\mathcal{D}}_t = \mathcal{D}_t$ and $\widehat{\mathbf{r}}_t = \sum_h \mathbf{r}_{t,h}$.

Estimator properties. Fix the image, target, Jacobian, and gate decision, denoted collectively by $\mathcal{F}_t$. For a fresh assignment Π with $\Pr_\Pi(Z_t = 1, H_t = h) = q_t p_h$, disjoint RGB supports give

$$\mathbb{E}_\Pi[\widehat{\mathbf{r}}_t; \mathcal{F}_t \text{ fixed}] = \mathbf{r}_t, \qquad \mathbf{r}_t = \sum_h \mathbf{r}_{t,h}. \tag{9}$$

$$\frac{\mathbb{E}_\Pi\|\widehat{\mathbf{r}}_t - \mathbf{r}_t\|^2}{\|\mathbf{r}_t\|^2} = \frac{3}{q_t} - 1, \qquad \mathbf{r}_t \neq \mathbf{0}. \tag{10}$$

Uniform sampling balances equal-cost kernels without estimating three derivative norms, but is not generally variance-optimal. The identities apply to a fixed state: without-replacement blocks and Adam couple training steps, precluding a trajectory-wise conditional-unbiasedness claim. Appendix D.3 derives the identities; Section 4.6 measures optimizer and topology changes.

3.2 Densification-Aware Sampling

Let $t_0 = 3000$, T_d be the densification endpoint (15000 for FastGS-based hosts), and $\Delta = 768$. The schedule is

$$\mathrm{mode}(t) = \begin{cases} \text{full RGB}, & t < t_0, \\ \text{one channel}, & t_0 \le t < T_d, \\ \text{full RGB}, & T_d \le t < T_d + \Delta, \\ \text{time–channel sampling}, & t \ge T_d + \Delta. \end{cases} \tag{11}$$

Warm-up and stabilization use full structural supervision. Active densification uses $q_t = 1$; late refinement retains two of three iteration positions in shuffled order ($q_t = q = 2/3$). An independent RGB permutation advances only on applied selected-channel updates: host rejections do not advance it. The schedule reads iteration boundaries, not topology events, and uses its own random stream.

Nominal structural-work fractions are 1, $1/3$, 1, and $2/9$. Channel-only and joint sampling have relative variance 2 and 3.5 (Eq. (10)), motivating the later introduction of temporal thinning. Section 4.5 tests this policy.

3.3 Single-Render Selective Autograd Path and Host Composition

All modes reuse one render and the cached RGB SSIM statistics. On selected updates, the autograd function slices one channel, runs its fused VJP, and restores the RGB adjoint with zeros in the other channels. Single-channel normalization supplies $1/p_h$; the VJP seed supplies $1/q_t$. Dense ℓ_1 is then added before the renderer backward. Unselected structural updates use forward-only D-SSIM with ℓ_1 backward.

The adapter sets the gradient mode before evaluating the exact loss. The host gate reads its detached scalar and decides whether to run backward; rejected updates execute no backward kernels and do not advance the channel cycle. The selected-channel VJP operates on $1 \times H \times W$ tensors; restoring three channels preserves the rasterizer's RGB gradient interface. Appendices E and D.2 give the tensor flow, normalization, and kernel details.

4 Experiments

4.1 Experimental Setup

Datasets and metrics. We use the released splits of 13 scenes from Mip-NeRF 360 (Barron et al., 2022), Tanks & Temples (Knapitsch et al., 2017), and Deep Blending (Hedman et al., 2018), reporting peak signal-to-noise ratio (PSNR), structural similarity (SSIM) (Wang et al., 2004), and learned perceptual image patch similarity (LPIPS) (Zhang et al., 2018). Quality averages scenes; time sums them. Process-wall time includes data setup and saving, but excludes separate rendering/evaluation; native timers are labeled. Appendix B.3 defines frames per second (FPS) and peak-memory measurement.

Implementation and matched protocol. We reproduce released methods locally on the same NVIDIA RTX 3090 to control hardware-dependent runtime, using each evaluated configuration's

Table 1: Local benchmark: PSNR in dB, SSIM/LPIPS unitless, and training-process wall time in seconds. Upper: released method-specific settings; lower: matched FastGS Base compositions, averaged over three executions. Shading marks ours; bold marks the lowest displayed time per scene. DrJ means Dr Johnson, T&T means Tanks & Temples, and SA means SIGGRAPH Asia. Dataset means are in Table S2.

		Mip-NeRF 360									T&T		Deep Blending	
Method	Metric	Bicycle	Flowers	Garden	Stump	Treehill	Room	Counter	Kitchen	Bonsai	Train	Truck	DrJ	Playroom
Released systems: method-specific configurations														
3DGS	PSNR↑	25.22	21.51	27.37	26.62	22.64	31.61	29.04	31.35	32.23	22.20	25.46	29.34	30.04
(SIGGRAPH 2023)	SSIM↑	0.765	0.603	0.864	0.771	0.635	0.919	0.907	0.926	0.940	0.819	0.881	0.902	0.903
	LPIPS↓	0.210	0.339	0.108	0.215	0.325	0.220	0.201	0.126	0.205	0.199	0.143	0.239	0.243
	Time↓	1883	1257	1812	1512	1394	1511	1442	1777	1178	850	1020	1838	1351
Mini-Splatting	PSNR↑	25.18	21.48	26.78	27.23	22.74	31.21	28.57	31.20	30.98	21.61	25.12	29.49	30.30
(ECCV 2024)	SSIM↑	0.771	0.624	0.844	0.805	0.653	0.919	0.903	0.924	0.937	0.796	0.870	0.902	0.907
	LPIPS↓	0.227	0.328	0.150	0.198	0.316	0.214	0.200	0.130	0.203	0.246	0.160	0.260	0.253
	Time↓	964	1049	1067	963	1053	1473	1740	1748	1438	759	755	1130	989
Speedy-Splat	PSNR↑	24.84	21.32	26.69	26.72	22.48	30.65	28.09	29.82	31.00	21.67	25.16	29.08	29.98
(CVPR 2025)	SSIM↑	0.721	0.576	0.827	0.770	0.592	0.893	0.867	0.888	0.918	0.769	0.864	0.897	0.901
	LPIPS↓	0.308	0.398	0.189	0.259	0.447	0.282	0.276	0.203	0.253	0.291	0.191	0.269	0.274
	Time↓	1064	866	1099	934	850	991	1004	1095	950	471	609	979	800
Taming-3DGS	PSNR↑	24.85	21.05	27.38	25.92	22.92	31.50	28.97	31.28	32.15	22.62	25.27	29.24	30.10
(SA 2024)	SSIM↑	0.718	0.556	0.855	0.731	0.626	0.908	0.900	0.922	0.936	0.807	0.865	0.896	0.897
	LPIPS↓	0.295	0.409	0.127	0.294	0.386	0.251	0.221	0.140	0.220	0.233	0.184	0.270	0.278
	Time↓	391	316	689	267	376	332	399	490	387	250	205	276	210
DashGaussian	PSNR↑	25.41	21.86	27.42	27.01	23.00	31.66	28.91	31.35	31.93	22.18	25.73	29.57	30.25
(CVPR 2025)	SSIM↑	0.770	0.615	0.859	0.781	0.639	0.914	0.901	0.918	0.937	0.814	0.880	0.902	0.901
	LPIPS↓	0.214	0.329	0.124	0.217	0.321	0.233	0.212	0.142	0.208	0.211	0.154	0.250	0.252
	Time↓	502	383	456	308	425	310	312	469	284	302	262	282	217
Matched compositions: shared FastGS Base settings														
FastGS	PSNR↑	24.83	21.20	27.13	26.66	22.90	31.94	29.16	31.78	32.23	22.51	25.68	29.54	30.44
(CVPR 2026)	SSIM↑	0.715	0.560	0.835	0.755	0.612	0.920	0.907	0.929	0.942	0.805	0.872	0.898	0.905
	LPIPS↓	0.310	0.406	0.174	0.297	0.427	0.217	0.203	0.127	0.191	0.242	0.178	0.275	0.265
	Time↓	266	265	331	220	233	218	243	318	245	143	140	157	138
SkipGS	PSNR↑	24.79	21.19	27.13	26.64	22.84	31.75	29.06	31.73	32.07	22.31	25.63	29.45	30.64
(ECCV 2026)	SSIM↑	0.713	0.558	0.834	0.755	0.610	0.919	0.906	0.928	0.941	0.802	0.871	0.897	0.904
	LPIPS↓	0.312	0.408	0.176	0.298	0.430	0.219	0.205	0.128	0.192	0.245	0.181	0.278	0.267
	Time↓	243	237	299	202	211	201	216	283	221	130	121	137	122
FactorGrad-GS	PSNR↑	24.75	21.18	27.13	26.68	22.87	31.71	28.98	31.60	31.96	22.29	25.55	29.30	30.56
	SSIM↑	0.712	0.558	0.834	0.756	0.610	0.917	0.905	0.927	0.940	0.800	0.870	0.895	0.904
	LPIPS↓	0.313	0.407	0.176	0.298	0.431	0.220	0.207	0.129	0.193	0.248	0.183	0.280	0.266
	Time↓	**229**	**228**	**290**	**193**	**203**	**191**	**207**	**272**	**212**	**126**	**119**	**130**	**119**

default settings. Primary methods and ablations share FastGS Base settings: 30K iterations and densification interval 500. *SkipGS* is its original gate on FastGS; *FactorGrad-GS* adds Full with $\lambda_{\mathrm{str}} = 0.2$ and $q = 2/3$ across scenes. Three fixed-seed executions measure execution-level, not cross-seed, variability. Additional hosts use separate matched campaigns.

4.2 Locally Reproduced Benchmark

FactorGrad-GS has lower dataset-summed time than the released systems but does not dominate their quality (Table 1, upper group). The lower group isolates incremental effects at shared FastGS Base settings. Dataset radar profiles appear in Appendix B.1 (Figure S1).

4.3 Matched Composition with SkipGS

Adding FactorGrad-GS to SkipGS saves 3.91% of summed wall time on average, with a sample standard deviation of 0.47 percentage points (Table 2; Figure 3). All three executions are faster, and every scene has a lower mean time. Mean PSNR changes by -0.03, -0.05, and -0.11 dB on Mip-NeRF 360, Tanks & Temples, and Deep Blending, respectively. PSNR increases on three scenes and decreases on ten; Dr Johnson has the largest decrease (-0.146 dB).

Table 2: Three fixed-seed executions: mean ± sample standard deviation. Each execution averages quality and sums wall time over 13 scenes. PSNR is in dB, SSIM/LPIPS are unitless, and time is in seconds.

Method	PSNR↑	SSIM↑	LPIPS↓	Time↓
FastGS (CVPR 2026)	27.38 ± 0.03	0.8197 ± 0.0003	0.2548 ± 0.0002	2918.3 ± 37.4
SkipGS (ECCV 2026)	27.32 ± 0.02	0.8185 ± 0.0001	0.2569 ± 0.0001	2622.7 ± 17.7
FactorGrad-GS	27.28 ± 0.02	0.8176 ± 0.0003	0.2578 ± 0.0001	**2520.0** ± 16.8

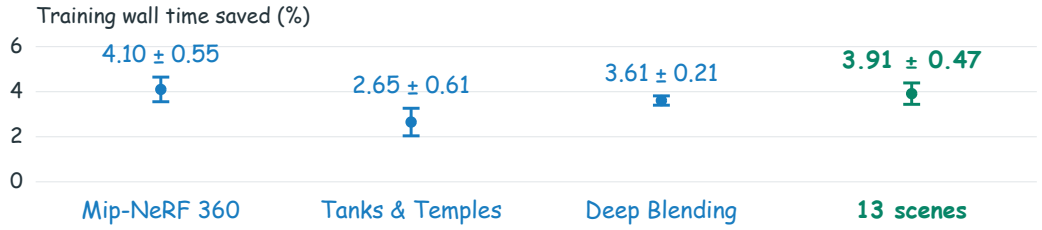

Figure 3: Incremental wall-time saving on the same SkipGS host: mean and sample standard deviation across three paired fixed-seed executions. Each saving uses summed scene times; error bars are in percentage points.

Separate matched-view renders compare Flowers and Playroom (Figure 4) and show a local loss on Room (Appendix F.5). FPS and peak memory have mixed changes; resource and per-scene quality results appear in Appendices B.3–B.4.

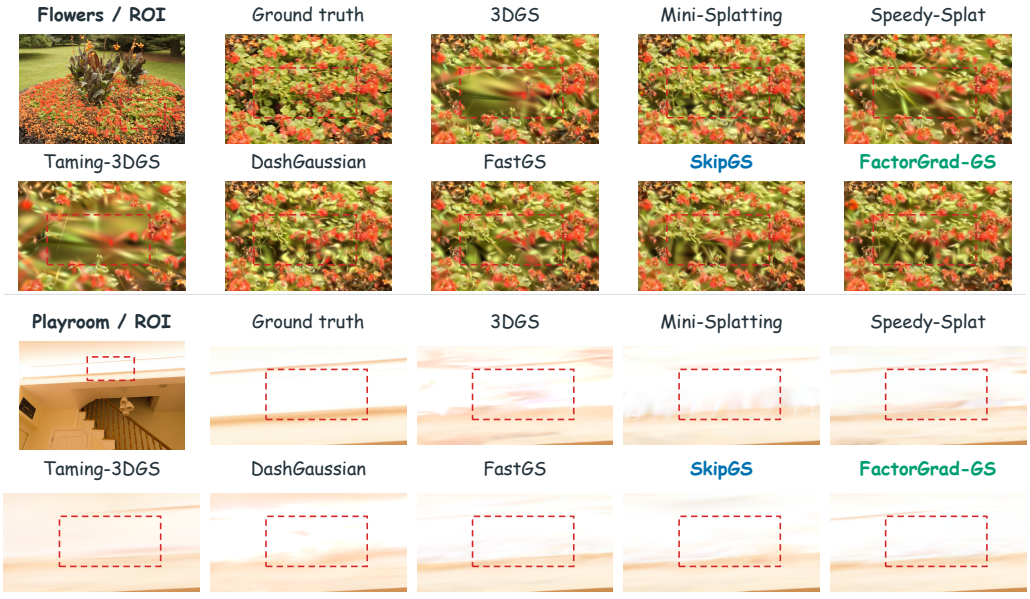

Figure 4: Aligned Flowers and Playroom crops from a separate visualization run. Red dashed rectangles locate crops and mark the same flower boundaries and ceiling edges in every method. Further views, a Room regression, and shared-scale error maps appear in Appendix Figures S7 and S8.

4.4 Composition with Additional Hosts

Without SkipGS, Full saves 98 s on FastGS at a 0.05 dB mean PSNR cost (Table 3). LeGS retains nearly the same PSNR; DashGaussian and Shorter-GS have smaller time savings. On Vanilla 3DGS, Late reduces time, whereas Full is slower.

Table 3: Host / host + FactorGrad-GS, one execution each: mean quality and summed time over 13 scenes. PSNR is in dB, SSIM/LPIPS are unitless, and time is in seconds. Compare times within pairs, not across timer groups. Shading marks composition; bold marks shorter time. Vanilla uses Late; others use Full.

Host	PSNR↑	SSIM↑	LPIPS↓	Time↓
External training-process wall time				
FastGS (CVPR 2026)	27.36 / 27.31	0.819 / 0.819	0.255 / 0.256	2903 / **2805**
Shorter-GS (CVPR 2026)	27.00 / 26.95	0.828 / 0.826	0.220 / 0.224	2547 / **2538**
Native training time				
Vanilla 3DGS (SIGGRAPH 2023)	27.30 / 27.25	0.832 / 0.831	0.216 / 0.218	22098 / **21071**
Taming-3DGS (SA 2024)	27.21 / 27.15	0.817 / 0.816	0.255 / 0.256	4842 / **4735**
DashGaussian (CVPR 2025)	27.41 / 27.34	0.833 / 0.832	0.220 / 0.222	4445 / **4416**
LeGS (ICML 2026)	28.02 / 28.00	0.853 / 0.853	0.184 / 0.186	8256 / **8026**

4.5 Component Ablation

Table 4: SkipGS-host ablation on nine Mip-NeRF 360 scenes: mean quality and summed wall time (mean $\pm$ sample standard deviation, three fixed-seed executions). C: densification-aware budget; IPW: inverse-probability weighting. PSNR is in dB, SSIM/LPIPS are unitless, time is in seconds, and #G is in millions.

Variant	PSNR↑	SSIM↑	LPIPS↓	Time↓	#G
SkipGS (ECCV 2026)	27.46	0.796	0.263	2113 ± 14	0.401
FactorGrad-GS	27.43	0.796	0.264	2026 ± 12	0.406
Module removals from Full					
w/o Channel sampling	27.45	0.796	0.264	2085 ± 20	0.401
w/o Temporal sampling	27.43	0.796	0.263	2044 ± 15	0.406
w/o Stage budget (C)	27.41	0.795	0.265	2026 ± 13	0.410
Estimator stress tests					
w/o IPW	27.05	0.762	0.307	1803 ± 18	0.236
Unweighted RGB (BM-Full)	26.99	0.759	0.310	**1755** ± 23	0.229

Removing channel or temporal sampling adds 59 or 18 s, respectively, with increases in all three executions (Table 4). Without C, joint sampling starts at 3K with no stabilization: mean time is unchanged, PSNR falls by 0.02 dB, and Gaussian count rises. On Deep Blending, however, PSNR rises from 29.93 to 29.95 dB (Appendix B.5).

Removing IPW reduces gradient scale and Gaussian count, saving time at a 0.38 dB PSNR loss. Unweighted RGB thinning (BM-Full) also removes stage protection; it matches nominal channel work, not CUDA time, so it cannot isolate sampling granularity.

Full outpaces Late on each dataset in the matched-host budget sweep (Table 5), but reducing q to 0.5 saves no further wall time. Stabilization length has mixed quality effects; the chosen schedule represents a time–quality operating point, not a universal optimum.

Table 5: Separate budget sweep, one execution per setting. Cells give mean PSNR (dB) / summed wall time (seconds). Full uses $q = 2/3$, $\Delta = 768$; Late omits early channel sampling. Other rows change one Full setting. Complete metrics are in Appendix B.6.

Setting	Mip-NeRF 360	Tanks & Temples	Deep Blending
Full	27.42 / **2031**	23.92 / **245**	30.05 / **250**
Late	27.45 / 2100	23.96 / 253	30.10 / 258
$q = 0.5$	27.41 / 2057	23.91 / 247	29.95 / 251
$q = 0.8$	27.42 / 2063	23.94 / 250	29.78 / 254
$\Delta = 0$	27.43 / 2072	24.05 / 251	29.93 / 255
$\Delta = 1536$	27.43 / 2058	23.94 / 248	29.83 / 253

4.6 Backward Cost and Optimization Effects

Synchronized CUDA timing (20 warmups, 100 trials) gives 0.077/0.255 ms for one-channel D-SSIM backward at $800 \times 533/1600 \times 1066$, versus 0.156/0.537 ms for RGB. The larger-resolution objective takes 1.335 versus 1.613 ms. Across Bicycle, Kitchen, Train, and Dr Johnson, cumulative image-loss backward falls from 73.66 to 49.16 s; rasterizer backward changes from 311.57 to 309.37 s. CPU budget decisions take 9–11 μs.

Frozen-image RGB-cycle error is at most 2.22×10^{-5}, verifying normalization, but training changes Adam states and density events (Table 6). Late gate acceptance shifts by -1.81 to $+0.37$ percentage points (Kitchen/Bicycle); objective trajectories differ despite similar trends (Figure 5). These observations characterize optimization effects, not event-level causality (Appendix C).

Table 6: Four scene pairs. Top: channel/RGB gradient cosine and norm ratio, joint-sampling relative variance, and root-mean-square (RMS) Adam ratios (ours/SkipGS), ranged over six parameter groups. Bottom: density-count differences (ours minus SkipGS) and late gate acceptance (SkipGS / ours).

Stage	Cosine	Norm ratio	Rel. variance	First moment	Root second moment
Active densification	0.576	1.729	2.00	0.99–1.20	1.00–1.17
Late refinement	0.576	1.728	3.50	1.07–1.21	0.93–1.11

Scene	Final Gaussians	Clone parents	Split parents	Pruned	Late gate
Bicycle	-1087	$+123$	$+7728$	$+8938$	0.613 / 0.617
Kitchen	$+2051$	$+475$	$+17452$	$+15876$	0.612 / 0.594
Train	-7256	$+2205$	$+2655$	$+12116$	0.619 / 0.612
Dr Johnson	-2945	$+6999$	$+4412$	$+14356$	0.533 / 0.525

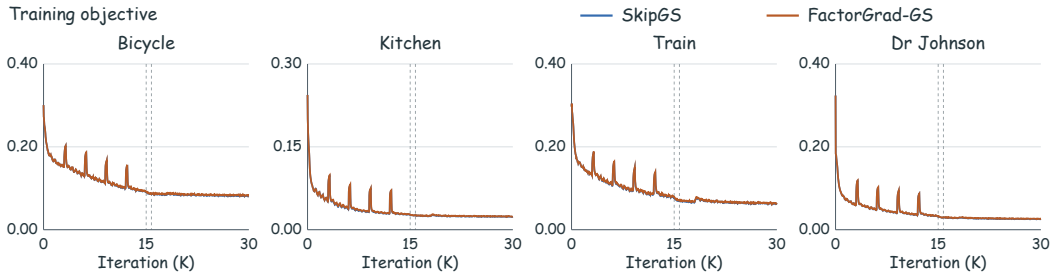

Figure 5: Exact RGB ℓ_1+D-SSIM training objectives in four paired runs, smoothed over 256 logged samples. The horizontal axis is iteration, not wall time; these are not held-out PSNR curves.

5 Conclusion

FactorGrad-GS separates the evaluation of the reconstruction objective from the computation of its structural derivative. It preserves the exact RGB loss and dense ℓ_1 supervision while reducing D-SSIM backward through channel sampling, temporal sampling, and a densification-aware budget. The single-render implementation reuses forward statistics and operates within host-accepted updates, complementing whole-update gates without replacing the renderer or optimizer. Experiments across three benchmarks and multiple hosts show that this loss-side intervention can reduce training time, with modest quality trade-offs in the matched SkipGS setting. Direct profiling locates the saving in structural backward, while gradient and topology measurements characterize its effect on optimization. These results support treating structural derivatives as a separate computational budget. Adapting this budget to measured variance and execution cost, and extending it beyond channel-separable D-SSIM, are directions for further study.

AI USE STATEMENT

We used OpenAI Codex solely for writing assistance and language refinement during the preparation of the manuscript. The authors take full responsibility for the final manuscript and its content.

REPRODUCIBILITY STATEMENT

Section 3 specifies the estimator and sampling schedule. Appendix D provides the assumptions and derivations, and Appendix E documents the execution order, hyperparameters, software versions, data splits, and evaluation procedure. Section 4 describes the hardware and comparison protocol. Complete scene-level results, fixed-seed repeated executions, ablations, and schedule sensitivities appear in Appendix B. Appendix C details the synchronized kernel timing and gradient measurements, distinguishing them from end-to-end training time. The supplementary package contains the FactorGrad module, host integration sources, fixed configurations, aggregation scripts, and archived evidence needed to reproduce the reported tables; it does not bundle the benchmark datasets or the full collection of trained models.

REFERENCES

Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5460–5469, 2022. doi: 10.1109/CVPR52688.2022.00539.

Youyu Chen, Junjun Jiang, Kui Jiang, Xiao Tang, Zhihao Li, Xianming Liu, and Yinyu Nie. Dash-Gaussian: Optimizing 3D gaussian splatting in 200 seconds. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11146–11155, 2025. doi: 10.1109/CVPR52734.2025.01041.

Ankit Dhiman, Tao Lu, R Srinath, Emre Arslan, Angela Xing, Yuanbo Xiangli, Venkatesh Babu Radhakrishnan, and Srinath Sridhar. Turbo-GS: Accelerating 3D gaussian fitting for high-resolution radiance fields. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15454–15464, 2026.

Renjie Ding, Yaonan Wang, Min Liu, Jialin Zhu, Jiazheng Wang, Jiahao Zhao, Wenting Shen, Feixiang He, and Xiang Chen. A step to decouple optimization in 3DGS. In *International Conference on Learning Representations (ICLR)*, 2026.

Zheng Dong, Daifei Qiu, Pinxuan Dai, Ke Xu, Jiamin Xu, Lili He, Rynson W. H. Lau, and Weiwei Xu. TurboGS: Accelerating 3D gaussian splatting via error-guided sparse pixel sampling and optimization. In *International Conference on Machine Learning (ICML)*, 2026.

Guangchi Fang and Bing Wang. Mini-Splatting: Representing scenes with a constrained number of gaussians. In *European Conference on Computer Vision (ECCV)*, pp. 165–181, 2024. doi: 10.1007/978-3-031-72980-5_10.

Florian Hahlbohm, Linus Franke, Martin Eisemann, and Marcus Magnor. Faster-GS: Analyzing and improving gaussian splatting optimization. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18946–18957, 2026.

Alex Hanson, Allen Tu, Geng Lin, Vasu Singla, Matthias Zwicker, and Tom Goldstein. Speedy-Splat: Fast 3D gaussian splatting with sparse pixels and sparse primitives. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 21537–21546, 2025. doi: 10.1109/CVPR52734.2025.02006.

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*, 37(6), 2018. doi: 10.1145/3272127.3275084.

Lukas Höllein, Aljaž Božič, Michael Zollhöfer, and Matthias Nießner. 3DGS-LM: Faster gaussian-splatting optimization with Levenberg-Marquardt. In *International Conference on Computer Vision (ICCV)*, pp. 26740–26750, 2025. doi: 10.1109/ICCV51701.2025.02482.

Roger Hsiao, Yuchen Fang, Xiangru Huang, Ruilong Li, Hesam Rabeti, Zan Gojcic, Javad Lavaei, James Demmel, and Sophia Shao. 3DGS^2-TR: A scalable second-order trust-region method for 3D gaussian splatting. In *International Conference on Machine Learning (ICML)*, 2026.

Angela H. Jiang, Daniel L.-K. Wong, Giulio Zhou, David G. Andersen, Jeffrey Dean, Gregory R. Ganger, Gauri Joshi, Michael Kaminksy, Michael Kozuch, Zachary C. Lipton, and Padmanabhan Pillai. Accelerating deep learning by focusing on the biggest losers. *arXiv preprint arXiv:1910.00762*, 2019. doi: 10.48550/arXiv.1910.00762.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4), 2023. doi: 10.1145/3592433.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics*, 36(4), 2017. doi: 10.1145/3072959.3073599.

Dmytro Kotovenko, Olga Grebenkova, and Björn Ommer. EDGS: Eliminating densification for efficient convergence of 3DGS. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 41065–41076, 2026.

Yongjae Lee, Jingxing Li, Abhay Kumar Yadav, Rama Chellappa, and Deliang Fan. AdpSplit: Error-driven adaptive splitting for faster geometry discovery in 3D gaussian splatting. *arXiv preprint arXiv:2605.06876*, 2026. doi: 10.48550/arXiv.2605.06876.

Jingxing Li, Yongjae Lee, and Deliang Fan. SkipGS: Post-densification backward skipping for efficient 3DGS training. In *European Conference on Computer Vision (ECCV)*, pp. 20–36, 2026. doi: 10.1007/978-3-032-37577-3_2.

Jiaqi Liu and Zhizhong Han. Speeding up the learning of 3D gaussians with much shorter gaussian lists. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1231–1240, 2026.

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

Saswat Subhajyoti Mallick, Rahul Goel, Bernhard Kerbl, Markus Steinberger, Francisco Vicente Carrasco, and Fernando De La Torre. Taming 3DGS: High-quality radiance fields with limited resources. In *SIGGRAPH Asia Conference Papers*, 2024. doi: 10.1145/3680528.3687694.

Théo Morales, Nhat-Quynh Le-Pham, Robin Atkins, and Binh-Son Hua. TruncGradGS: Improved 3D gaussian splatting via truncated gradient updates. In *Pacific Graphics (PG) Conference Papers and Posters*, 2026. doi: 10.2312/pg.20261050.

William S. Moses and Valentin Churavy. Instead of rewriting foreign code for machine learning, automatically synthesize fast gradients. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

Zhenhua Ning, Xin Li, Jun Yu, Guangming Lu, Yaowei Wang, and Wenjie Pei. Beyond heuristics: Learnable density control for 3D gaussian splatting. In *International Conference on Machine Learning (ICML)*, 2026.

Deniz Oktay, Nick McGreivy, Joshua Aduol, Alex Beatson, and Ryan P. Adams. Randomized automatic differentiation. In *International Conference on Learning Representations (ICLR)*, 2021.

Alexander Rath, Pascal Grittmann, Sebastian Herholz, Philippe Weier, and Philipp Slusallek. EARS: Efficiency-aware russian roulette and splitting. *ACM Transactions on Graphics*, 41(4), 2022. doi: 10.1145/3528223.3530168.

Shiwei Ren, Tianci Wen, Yongchun Fang, and Biao Lu. FastGS: Training 3D gaussian splatting in 100 seconds. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 26094–26103, 2026.

Xu Sun, Xuancheng Ren, Shuming Ma, and Houfeng Wang. meProp: Sparsified back propagation for accelerated deep learning with reduced overfitting. In *International Conference on Machine Learning (ICML)*, volume 70 of *Proceedings of Machine Learning Research*, pp. 3299–3308, 2017.

Delio Vicini, Sébastien Speierer, and Wenzel Jakob. Path replay backpropagation: Differentiating light paths using constant memory and linear time. *ACM Transactions on Graphics*, 40(4), 2021. doi: 10.1145/3450626.3459804.

Laurent Vit, Oliver Batchelor, and Richard Green. 3D gaussian accelerated ray tracing: Fast training through particle-based backward propagation. *arXiv preprint arXiv:2608.17298*, 2026. doi: 10.48550/arXiv.2608.17298.

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. doi: 10.1109/TIP.2003.819861.

Zi-Ming Wang, Kai-Wen Duan, Kowei Huang, Akihiro Sugimoto, and Shang-Hong Lai. Fast and compact 3D gaussian splatting with polarized opacity prior. In *European Conference on Computer Vision (ECCV)*, pp. 137–154, 2026. doi: 10.1007/978-3-032-37461-5_8.

Ziteng Wang, Jianfei Chen, and Jun Zhu. Efficient backpropagation with variance-controlled adaptive sampling. In *International Conference on Learning Representations (ICLR)*, 2024.

Alexander Wilkie, Sehera Nawaz, Marc Droske, Andrea Weidlich, and Johannes Hanika. Hero wavelength spectral sampling. *Computer Graphics Forum*, 33(4):123–131, 2014. doi: 10.1111/cgf.12419.

Zeke Xie, Xindi Yang, Yujie Yang, Qi Sun, Yixiang Jiang, Haoran Wang, Yunfeng Cai, and Mingming Sun. S3IM: Stochastic structural SIMilarity and its unreasonable effectiveness for neural fields. In *International Conference on Computer Vision (ICCV)*, pp. 17978–17988, 2023. doi: 10.1109/ICCV51070.2023.01652.

Yixiong Yang, Sisheng Zhang, Qingsong Yan, Shaohuai Shi, and Qiang Wang. Laplacian frequency hierarchies for efficient 3D gaussian splatting training. In *Pacific Graphics (PG) Conference Papers and Posters*, 2026. doi: 10.2312/pg.20261064.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 586–595, 2018. doi: 10.1109/CVPR.2018.00068.

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 (ICLR)*, 2026.

FactorGrad-GS: Supplementary Material

Exact-Primal Monte Carlo Structural Backpropagation

for Efficient 3D Gaussian Splatting

Overview. This supplement develops the connection between the exact reconstruction objective and its sampled structural derivative. After the highlights and comparison with related mechanisms, the experimental sections report scene-level quality and resources, component ablations, schedule sensitivity, and direct measurements of the backward path. The subsequent sections give the fused-SSIM derivation, sampling schedule, and host settings needed to reproduce these experiments. The final section shows camera-aligned ground truth and reconstructions across the three benchmarks.

The principal comparison uses FastGS Base, SkipGS on FastGS Base, and FactorGrad-GS on the same SkipGS host. Three independent executions share the same seeds and configuration. Additional host-composition experiments are reported separately. Tables identify their aggregation and timing conventions so that a scene-level measurement is not confused with a dataset total.

CONTENTS

All section names and page numbers below link to their destinations.

Section	Page
A. Highlights and relation to closest methods	14
B. Additional experiments and ablations	15
Dataset summaries and cross-method resources	15
Scene-level resource measurements	18
Repeated executions and paired changes	19
Component ablation on each dataset	20
Schedule sensitivity	21
Composition with other hosts	22
C. Backward-path and optimization measurements	23
Gradient, Adam, and density evolution	24
Training-objective trajectories	25
D. Structural objective and derivative estimator	26
3D Gaussian representation and reconstruction objective	26
SSIM statistics and fused backward	27
Exact value, sampling, and variance	28
E. Implementation and hyperparameters	29
Resolved host and module settings	30
F. Additional qualitative results	32
Extended multi-method comparison	35
Selected local error maps	36
References (following the main text)	10

Reading guide. Sections B and C separate the measured time–quality trade-off from the cost of an individual loss kernel. Sections D and E then specify the derivative estimator and its implementation. Section F uses the same held-out cameras for each ground-truth/render pair.

A HIGHLIGHTS AND RELATION TO CLOSEST METHODS

Insight: the loss value and its derivative have different jobs. A reconstruction loss evaluates the current image, but its derivative supplies an update direction. Controllers such as SkipGS use the former to decide whether to update; the optimizer consumes the latter. FactorGrad-GS keeps the complete RGB ℓ_1+D-SSIM value and reduces the work needed to produce the structural derivative. This separation allows the host to observe the same objective at a fixed state while the derivative budget varies.

Channel sampling targets a separable backward operator. The RGB SSIM reduction is a sum of channel-wise terms. A selected channel can therefore enter the fused backward kernel as a one-channel tensor. Its output is inverse-probability weighted and inserted into an RGB gradient tensor. This is a reduction in the executed structural backward workload, not a mask applied after computing all three channel derivatives.

Temporal sampling acts inside an accepted host update. In late refinement, selected iterations carry the structural derivative and the remaining accepted iterations retain the dense ℓ_1 derivative. SkipGS still decides whether the whole backward pass is executed. Its view gate and FactorGrad-GS thus control different units of computation.

Stage scheduling separates two uses of image gradients. Image derivatives affect continuous parameter fitting and the statistics used for Gaussian densification. The Full schedule delays channel sampling until iteration 3K, restores full RGB derivatives around the configured end of densification, and then enables temporal sampling. The boundaries are fixed by iteration, not inferred from per-Gaussian state.

A.1 MECHANISTIC COMPARISON

The closest methods differ in both the information they read and the computation they change (Table S1). Randomized automatic differentiation provides the general derivative-sampling perspective (Oktay et al., 2021); VCAS adapts sampling ratios using variance measurements (Wang et al., 2024). FactorGrad-GS uses a fixed stage schedule and a channel-separable image loss, rather than layer-wise adaptive sampling.

Selective-Backprop selects training examples for backward using their forward losses (Jiang et al., 2019). Its experiments concern CIFAR-10, CIFAR-100, and SVHN image classification, rather than the 3DGS scene benchmarks used here. The connection is selective derivative computation; its unit of selection is an example, whereas FactorGrad-GS selects structural-loss components inside an accepted scene update.

Table S1: Closest mechanisms. "Input" refers to the control signal or sampled operator, not the camera dataset shared by 3DGS methods.

Method	Input to mechanism	Output or intervention	Difference from FactorGrad-GS
FastGS (CVPR 2026)	Multi-view Gaussian importance	Densification and pruning decisions	Changes the primitive population.
SkipGS (ECCV 2026)	Current view loss and loss EMA	Execute or skip the entire backward	Gates views; retained updates keep the host loss derivative.
TurboGS (ICML 2026)	Reconstruction errors and tiles	Sparse pixels, structure-aware loss, density and optimizer changes	Changes supervision and several training components.
RAD (ICLR 2021)	Computation graph derivatives	Randomized reverse-mode estimates	General AD framework, rather than an SSIM-specific execution path.

Continued on the next page.

Table S1 (continued).

Method	Input to mechanism	Output or intervention	Difference from FactorGrad-GS
VCAS (ICLR 2024)	Layer-wise gradient statistics	Adaptive importance-sampled backward	Learns sampling ratios to control variance.
FactorGrad-GS	RGB structural loss, iteration, densification endpoint	Exact scalar and sampled structural image derivative	Keeps the host renderer and gate; specializes time/channel sampling to fused SSIM.

A.2 Additional Recent Work

AdpSplit adapts the number of split children to reconstruction errors and initializes them from image-error region statistics (Lee et al., 2026). It changes Gaussian growth and initialization, whereas FactorGrad-GS samples the structural-loss derivative within the host training schedule.

3DGART reorganizes Gaussian ray-tracing backpropagation from pixel-centric gradient scattering to primitive-centric accumulation, reducing atomic contention (Vit et al., 2026). Its intervention is in the renderer's gradient accumulation path. FactorGrad-GS instead reduces the structural image-loss backward workload while retaining the host renderer.

B Additional Experiments and Ablations

B.1 Dataset Summaries and Cross-Method Resources

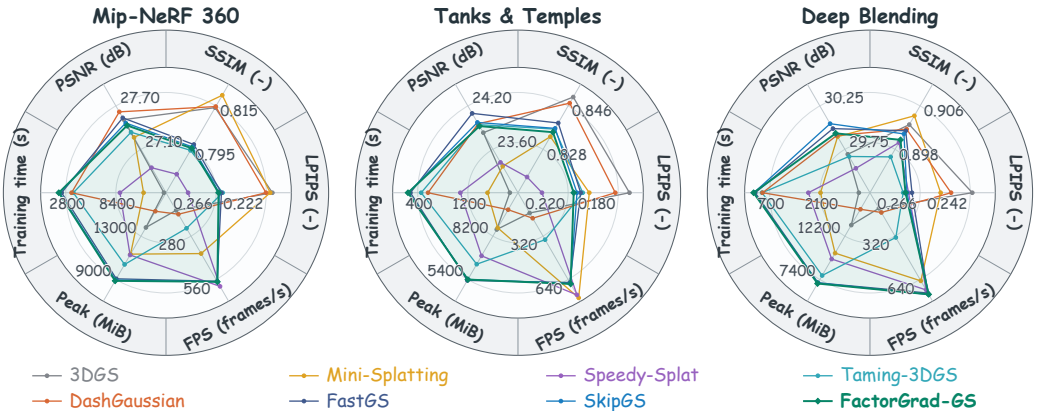

Figure S1: Profiles for Mip-NeRF 360, Tanks & Temples, and Deep Blending. Outward is favorable, with independent linear ranges per dataset. Time sums scenes; other metrics average scenes. SSIM/LPIPS are unitless; Peak is peak training memory. Primary methods average three executions.

The per-scene measurements in Table 1 are aggregated within each dataset below. The resource comparison additionally includes LeGS and Shorter-GS from the local baseline campaign. All timings in these two tables use the external training-process wall clock; native host timers are reported separately in Section B.7.

Table S2: Dataset-level summaries of Table 1. Quality is averaged and training-process wall time is summed within each dataset. PSNR is in dB, SSIM and LPIPS are unitless, and time is in seconds. Shading identifies FactorGrad-GS; boldface marks the lowest time in each dataset.

	Mip-NeRF 360				T&T				Deep Blending			
Method	PSNR↑	SSIM↑	LPIPS↓	Time↓	PSNR↑	SSIM↑	LPIPS↓	Time↓	PSNR↑	SSIM↑	LPIPS↓	Time↓
3DGS (SIGGRAPH 2023)	27.51	0.814	0.217	13766	23.83	0.850	0.171	1870	29.69	0.903	0.241	3189
Mini-Splatting (ECCV 2024)	27.26	0.820	0.219	11495	23.37	0.833	0.203	1514	29.90	0.904	0.256	2119
Speedy-Splat (CVPR 2025)	26.85	0.784	0.291	8853	23.42	0.817	0.241	1080	29.53	0.899	0.272	1779
Taming-3DGS (SA 2024)	27.34	0.795	0.260	3647	23.94	0.836	0.208	455	29.67	0.897	0.274	486
DashGaussian (CVPR 2025)	27.62	0.815	0.222	3449	23.95	0.847	0.182	564	29.91	0.902	0.251	499
FastGS (CVPR 2026)	27.54	0.797	0.261	2340	24.10	0.839	0.210	283	29.99	0.901	0.270	295
SkipGS (ECCV 2026)	27.46	0.796	0.263	2113	23.97	0.837	0.213	251	30.05	0.901	0.272	259
FactorGrad-GS	27.43	0.796	0.264	**2026**	23.92	0.835	0.216	**245**	29.93	0.900	0.273	**249**

Table S3: Complete quality and resource measurements across the three benchmarks. Quality, FPS, peak memory, and Gaussian count are scene-equal means; training time is summed over 13 scenes. SkipGS and FactorGrad-GS use the matched experiment block; other methods use released training configurations. PSNR is in dB, SSIM and LPIPS are unitless, time is in seconds, FPS is in frames/s, peak memory is in MiB, and #G is in millions. Shading identifies FactorGrad-GS; boldface marks the lowest time, highest FPS, lowest peak memory, and smallest Gaussian count.

Method	PSNR↑	SSIM↑	LPIPS↓	Time↓	FPS↑	Peak↓	#G
3DGS (SIGGRAPH 2023)	27.28	0.833	0.213	18825	119.19	12945	2.498
Mini-Splatting (ECCV 2024)	27.07	0.835	0.222	15128	490.89	10783	0.429
Speedy-Splat (CVPR 2025)	26.73	0.806	0.280	11712	**644.26**	10352	**0.282**
Taming-3DGS (SA 2024)	27.17	0.817	0.254	4588	262.88	9384	0.554
DashGaussian (CVPR 2025)	27.41	0.833	0.221	4512	147.31	14447	2.036
LeGS (ICML 2026)	28.03	0.854	0.184	8303	339.27	15834	1.926
Shorter-GS (CVPR 2026)	27.00	0.828	0.220	**2445**	281.47	15134	3.003
FastGS (CVPR 2026)	27.38	0.820	0.255	2918	611.02	8183	0.349
SkipGS (ECCV 2026)	27.32	0.818	0.257	2623	615.82	8109	0.348
FactorGrad-GS	27.28	0.818	0.258	2520	616.41	**8049**	0.352

Rendering FPS is measured on the final representation. Peak memory is the scene-equal mean of device-level training peaks sampled every 0.2 s; PyTorch allocation counters are not mixed into this value.

The three profiles in Figure S1 use dataset-specific aggregations of the same scene records. Quality and summed wall time are reported in Table S2; the remaining two axes and their plotting ranges are provided below.

Table S4: Dataset-level rendering throughput and peak training memory underlying Figure S1. FPS is in frames/s and memory is in MiB. Both are scene-equal means; primary methods first average the three executions of each scene.

	Mip-NeRF 360		Tanks & Temples		Deep Blending	
Method	FPS	Peak	FPS	Peak	FPS	Peak
3DGS (SIGGRAPH 2023)	113.49	13799	149.45	8628	114.57	13422
Mini-Splatting (ECCV 2024)	392.51	11341	774.75	8734	649.77	10322
Speedy-Splat (CVPR 2025)	604.00	11265	753.22	6930	716.47	9666
Taming-3DGS (SA 2024)	229.46	10392	346.29	6390	329.87	7846
DashGaussian (CVPR 2025)	138.66	15295	187.99	9904	145.57	15172
FastGS (CVPR 2026)	571.13	9068	662.05	5344	739.49	7038
SkipGS (ECCV 2026)	574.07	8948	669.23	5408	750.31	7030
FactorGrad-GS	574.47	8881	672.40	5398	749.11	6957

The radar panels use independent linear ranges. The centre-to-rim endpoints are listed below; outward is favorable on every axis. Polygon area is not an aggregate score.

Axis	Mip-NeRF 360	Tanks & Temples	Deep Blending
PSNR (dB)	26.50→28.00	23.00→24.50	29.25→30.50
SSIM (-)	0.775→0.825	0.810→0.855	0.890→0.910
LPIPS (-)	0.310→0.200	0.260→0.160	0.290→0.230
FPS (frames/s)	0→700	0→800	0→800
Peak (MiB)	17000→7000	11000→4000	17000→5000
Training time (s)	14000→0	2000→0	3500→0

B.2 Complete Scene-Level Reconstruction Results

The primary matched comparison averages three independent fixed-seed executions for every scene (Table S5). Time uses the archived external training-wrapper wall clock: it includes process launch, data setup, final model saving, and monitor shutdown, with one-second timestamp resolution. Rendering, evaluation, and FPS are separate processes. All three methods use this same boundary. SkipGS and FactorGrad-GS use FastGS Base as their renderer and density-control host.

Table S5: Per-scene quality and training-process wall time. The first nine scenes are Mip-NeRF 360, Train and Truck are Tanks & Temples, and the last two are Deep Blending. Quality is the test-view mean within each scene. PSNR is in dB, SSIM and LPIPS are unitless, and wall time is in seconds.

Scene	Method	PSNR↑	SSIM↑	LPIPS↓	Wall↓
Bicycle	FastGS (CVPR 2026)	24.83	0.715	0.310	266.3
	SkipGS (ECCV 2026)	24.79	0.713	0.312	243.3
	FactorGrad-GS	24.75	0.712	0.313	**228.7**
Flowers	FastGS (CVPR 2026)	21.20	0.560	0.406	265.0
	SkipGS (ECCV 2026)	21.19	0.558	0.408	237.3
	FactorGrad-GS	21.18	0.558	0.407	**227.7**
Garden	FastGS (CVPR 2026)	27.13	0.835	0.174	331.3
	SkipGS (ECCV 2026)	27.13	0.834	0.176	298.7
	FactorGrad-GS	27.13	0.834	0.176	**290.3**
Stump	FastGS (CVPR 2026)	26.66	0.755	0.297	220.3
	SkipGS (ECCV 2026)	26.64	0.755	0.298	202.0
	FactorGrad-GS	26.68	0.756	0.298	**193.3**
Treehill	FastGS (CVPR 2026)	22.90	0.612	0.427	233.3
	SkipGS (ECCV 2026)	22.84	0.610	0.430	211.3
	FactorGrad-GS	22.87	0.610	0.431	**203.3**
Room	FastGS (CVPR 2026)	31.94	0.920	0.217	217.7
	SkipGS (ECCV 2026)	31.75	0.919	0.219	201.0
	FactorGrad-GS	31.71	0.917	0.220	**191.3**
Counter	FastGS (CVPR 2026)	29.16	0.907	0.203	243.0
	SkipGS (ECCV 2026)	29.06	0.906	0.205	215.7
	FactorGrad-GS	28.98	0.905	0.207	**207.3**
Kitchen	FastGS (CVPR 2026)	31.78	0.929	0.127	318.0
	SkipGS (ECCV 2026)	31.73	0.928	0.128	282.7
	FactorGrad-GS	31.60	0.927	0.129	**271.7**
Bonsai	FastGS (CVPR 2026)	32.23	0.942	0.191	245.3
	SkipGS (ECCV 2026)	32.07	0.941	0.192	220.7
	FactorGrad-GS	31.96	0.940	0.193	**212.3**
Train	FastGS (CVPR 2026)	22.51	0.805	0.242	143.3
	SkipGS (ECCV 2026)	22.31	0.802	0.245	130.3
	FactorGrad-GS	22.29	0.800	0.248	**125.7**

Continued on the next page.

Table S5 (continued).

Scene	Method	PSNR↑	SSIM↑	LPIPS↓	Wall↓
Truck	FastGS (CVPR 2026)	25.68	0.872	0.178	139.7
	SkipGS (ECCV 2026)	25.63	0.871	0.181	121.0
	FactorGrad-GS	25.55	0.870	0.183	**119.0**
Dr Johnson	FastGS (CVPR 2026)	29.54	0.898	0.275	156.7
	SkipGS (ECCV 2026)	29.45	0.897	0.278	137.0
	FactorGrad-GS	29.30	0.895	0.280	**130.0**
Playroom	FastGS (CVPR 2026)	30.44	0.905	0.265	138.3
	SkipGS (ECCV 2026)	30.64	0.904	0.267	121.7
	FactorGrad-GS	30.56	0.904	0.266	**119.3**

B.3 Scene-Level Resource Measurements

FPS measures final-model rendering throughput, not training iterations per second. It uses CUDA-event timing after ten warmup renders and covers three complete sweeps of the test cameras. Peak memory is the maximum device-level sample during training, expressed in GiB. It is distinct from PyTorch's allocated or reserved memory counters. The CUDA column sums the trainer's iteration timers and is provided alongside, not substituted for, the wall time in Table S5.

Table S6: Mean resources over the same three executions. FPS is in frames/s, peak device memory is in GiB, Gaussian count is in millions, and CUDA time is in seconds. All FPS values within a scene use the same camera and resolution settings.

Scene	Method	FPS↑	Peak	#G	CUDA time
Bicycle	FastGS (CVPR 2026)	558.2	7.95	0.538	214.1
	SkipGS (ECCV 2026)	556.0	7.71	0.538	188.7
	FactorGrad-GS	576.3	7.88	0.536	**177.2**
Flowers	FastGS (CVPR 2026)	578.1	7.73	0.487	215.4
	SkipGS (ECCV 2026)	573.7	7.75	0.485	187.7
	FactorGrad-GS	573.3	7.63	0.493	**178.4**
Garden	FastGS (CVPR 2026)	478.4	9.11	0.738	275.1
	SkipGS (ECCV 2026)	482.4	8.56	0.741	242.5
	FactorGrad-GS	476.9	8.69	0.758	**234.4**
Stump	FastGS (CVPR 2026)	605.9	6.28	0.385	184.2
	SkipGS (ECCV 2026)	605.0	6.27	0.387	165.7
	FactorGrad-GS	600.9	6.17	0.393	**157.4**
Treehill	FastGS (CVPR 2026)	602.1	6.93	0.395	192.5
	SkipGS (ECCV 2026)	594.8	6.63	0.391	170.3
	FactorGrad-GS	604.4	6.88	0.395	**162.6**
Room	FastGS (CVPR 2026)	660.0	10.33	0.208	179.8
	SkipGS (ECCV 2026)	671.8	10.20	0.206	162.9
	FactorGrad-GS	671.9	9.91	0.210	**153.3**
Counter	FastGS (CVPR 2026)	576.5	9.60	0.208	210.6
	SkipGS (ECCV 2026)	578.9	9.95	0.207	183.9
	FactorGrad-GS	583.4	9.67	0.208	**175.4**
Kitchen	FastGS (CVPR 2026)	473.5	12.04	0.378	280.4
	SkipGS (ECCV 2026)	484.4	11.25	0.378	245.1
	FactorGrad-GS	466.8	11.30	0.380	**234.5**
Bonsai	FastGS (CVPR 2026)	607.6	9.72	0.276	207.4
	SkipGS (ECCV 2026)	619.6	10.33	0.277	183.4
	FactorGrad-GS	616.4	9.92	0.278	**175.0**

Continued on the next page.

Table S6 (continued).

Scene	Method	FPS↑	Peak	#G	CUDA time
Train	FastGS (CVPR 2026)	621.5	5.42	0.231	132.7
	SkipGS (ECCV 2026)	628.4	5.63	0.230	119.6
	FactorGrad-GS	630.6	5.36	0.224	**115.6**
Truck	FastGS (CVPR 2026)	702.6	5.02	0.253	129.0
	SkipGS (ECCV 2026)	710.0	4.93	0.254	110.9
	FactorGrad-GS	714.2	5.18	0.255	**109.0**
Dr Johnson	FastGS (CVPR 2026)	715.4	7.39	0.251	144.7
	SkipGS (ECCV 2026)	718.5	7.38	0.248	125.1
	FactorGrad-GS	731.0	7.33	0.244	**118.7**
Playroom	FastGS (CVPR 2026)	763.6	6.35	0.186	127.4
	SkipGS (ECCV 2026)	782.1	6.36	0.185	111.4
	FactorGrad-GS	767.2	6.26	0.200	**108.6**

B.4 Repeated Executions and Paired Changes

Repeated executions hold the host seed and module seed fixed. Their spread measures execution-level variability, including nondeterministic GPU operations, rather than sensitivity to independently chosen training seeds. We compute dataset and overall statistics within each execution before calculating the sample standard deviation across the three executions.

Table S7: Mean $\pm$ sample standard deviation. Quality weights the 13 scenes equally; wall time sums their training times. PSNR is in dB, SSIM and LPIPS are unitless, and wall time is in seconds.

Method	PSNR	SSIM	LPIPS	Wall time
FastGS (CVPR 2026)	27.384 ± 0.025	0.8197 ± 0.0003	0.2548 ± 0.0002	2918.3 ± 37.4
SkipGS (ECCV 2026)	27.324 ± 0.015	0.8185 ± 0.0001	0.2569 ± 0.0001	2622.7 ± 17.7
FactorGrad-GS	27.275 ± 0.019	0.8176 ± 0.0003	0.2578 ± 0.0001	$\mathbf{2520.0} \pm 16.8$

For each scene and execution, we subtract the paired SkipGS result from FactorGrad-GS. This keeps the method comparison paired before aggregation. Positive PSNR/SSIM changes and negative LPIPS changes are favorable. Wall-time saving is $100(1 - T_{\mathrm{FG}}/T_{\mathrm{SkipGS}})$.

Table S8: Paired scene changes, mean $\pm$ sample standard deviation over three executions. PSNR changes are in dB, SSIM and LPIPS changes are unitless, and wall-time saving is a percentage (its standard deviation is in percentage points). Differences are computed before rounding the absolute values.

Scene	ΔPSNR	ΔSSIM	ΔLPIPS	Wall saving
Bicycle	-0.039 ± 0.027	-0.0009 ± 0.0008	$+0.0008 \pm 0.0012$	$+6.01 \pm 1.65$
Flowers	-0.005 ± 0.028	$+0.0003 \pm 0.0001$	-0.0016 ± 0.0004	$+4.06 \pm 1.55$
Garden	$+0.010 \pm 0.019$	-0.0003 ± 0.0004	-0.0000 ± 0.0003	$+2.79 \pm 0.84$
Stump	$+0.041 \pm 0.001$	$+0.0006 \pm 0.0010$	-0.0002 ± 0.0008	$+4.29 \pm 0.29$
Treehill	$+0.036 \pm 0.043$	-0.0002 ± 0.0006	$+0.0008 \pm 0.0004$	$+3.78 \pm 0.45$
Room	-0.042 ± 0.041	-0.0019 ± 0.0005	$+0.0018 \pm 0.0004$	$+4.81 \pm 0.31$
Counter	-0.073 ± 0.039	-0.0011 ± 0.0004	$+0.0016 \pm 0.0006$	$+3.86 \pm 0.25$
Kitchen	-0.132 ± 0.043	-0.0006 ± 0.0001	$+0.0010 \pm 0.0002$	$+3.87 \pm 1.46$
Bonsai	-0.108 ± 0.029	-0.0012 ± 0.0003	$+0.0009 \pm 0.0004$	$+3.77 \pm 0.67$
Train	-0.027 ± 0.129	-0.0021 ± 0.0007	$+0.0032 \pm 0.0002$	$+3.57 \pm 1.12$
Truck	-0.075 ± 0.030	-0.0010 ± 0.0005	$+0.0020 \pm 0.0006$	$+1.66 \pm 0.84$
Drjohnson	-0.146 ± 0.304	-0.0020 ± 0.0029	$+0.0020 \pm 0.0004$	$+5.11 \pm 0.69$
Playroom	-0.076 ± 0.135	-0.0002 ± 0.0020	-0.0007 ± 0.0004	$+1.92 \pm 0.48$

Runtime interpretation. The three total wall-time savings are 3.37%, 4.21%, and 4.16%. Their mean is 3.91% and sample standard deviation is 0.47 percentage points. A paired t interval with two degrees of freedom gives $[2.73, 5.09]\%$ for the mean saving. The corresponding absolute reduction is 102.7 ± 12.7 seconds. The repeat unit is an execution, not one of the heterogeneous scenes.

Quality interpretation. Mean PSNR improves on Stump, Treehill, and Garden, and decreases on the other ten scenes. Dr Johnson has the largest average PSNR decrease and the largest execution-level spread. Kitchen and Bonsai show smaller but more consistent decreases. The aggregate result is a time–quality trade-off, not a quality improvement on every scene. FPS and memory in Table S6 also depend on the representation produced by that optimization trajectory.

B.5 Component Ablation on Each Dataset

All variants in Table S9 use the FastGS Base plus SkipGS host and train independently from iteration 1. Removing channel sampling keeps full RGB structural derivatives on retained iterations. Removing temporal sampling sets $q = 1$. Removing the stage schedule starts joint time–channel sampling at 3K and omits the stabilization interval. The remaining host settings are unchanged.

Table S9: Three-execution means for each dataset. Quality, FPS, peak device memory, and Gaussian count average scenes; wall time sums scenes. Full is FactorGrad-GS with $q = 2/3$ and $\Delta = 768$. PSNR is in dB, SSIM and LPIPS are unitless, wall time is in seconds, FPS is in frames/s, peak memory is in GiB, and #G is in millions.

Variant	PSNR	SSIM	LPIPS	Wall	FPS	Peak	#G
			Mip-NeRF 360				
SkipGS (ECCV 2026)	27.46	0.796	0.263	2113	574.1	8.74	0.401
FactorGrad-GS	27.43	0.796	0.264	2026	574.5	8.67	0.406
w/o channel sampling	27.45	0.796	0.264	2085	576.2	8.66	0.401
w/o temporal sampling	27.43	0.796	0.263	2044	574.0	8.93	0.406
w/o stage schedule	27.41	0.795	0.265	2026	574.2	9.05	0.410
w/o IPW	27.05	0.762	0.307	1803	667.5	8.56	0.236
Unweighted BM-Full	26.99	0.759	0.310	**1755**	676.4	8.52	0.229
			Tanks & Temples				
SkipGS (ECCV 2026)	23.97	0.837	0.213	251	669.2	5.28	0.242
FactorGrad-GS	23.92	0.835	0.216	245	672.4	5.27	0.239
w/o channel sampling	23.94	0.836	0.214	250	668.6	5.16	0.242
w/o temporal sampling	23.94	0.836	0.215	246	673.3	5.29	0.238
w/o stage schedule	23.92	0.835	0.217	243	680.7	5.60	0.238
w/o IPW	23.71	0.817	0.239	225	723.3	5.15	0.181
Unweighted BM-Full	23.69	0.815	0.242	**219**	731.2	4.98	0.175
			Deep Blending				
SkipGS (ECCV 2026)	30.05	0.901	0.272	259	750.3	6.87	0.217
FactorGrad-GS	29.93	0.900	0.273	249	749.1	6.79	0.222
w/o channel sampling	30.00	0.900	0.273	257	751.1	7.02	0.216
w/o temporal sampling	29.69	0.899	0.273	252	747.0	7.00	0.222
w/o stage schedule	29.95	0.900	0.274	247	747.0	7.09	0.223
w/o IPW	29.66	0.893	0.290	228	813.8	6.74	0.152
Unweighted BM-Full	29.61	0.893	0.293	**220**	819.1	6.56	0.146

Module contributions. Channel sampling gives the largest incremental time reduction among the three module removals. Temporal sampling supplies an additional, smaller reduction. The stage schedule changes the balance between runtime and quality: removing it slightly reduces total time, but worsens mean SSIM and LPIPS on each dataset. It is not uniformly better on every quality metric; Deep Blending PSNR is slightly higher without this schedule.

Scaling and budget controls. The w/o IPW variant removes both channel and temporal compensation. The archived unweighted BM-Full control keeps full RGB structural backward at $q = 5/17$ after 3K, with neither stage protection nor compensation. This approximately matches the nominal channel-work budget, not measured CUDA cost or estimator variance. Both controls change density evolution and produce smaller representations, with lower quality and higher rendering FPS. Their time savings cannot be attributed solely to faster evaluation of the same structural update.

B.6 Schedule Sensitivity

The sensitivity study uses one complete matched campaign. Its Full row is from that same campaign, rather than the three-execution average. Late keeps full RGB derivatives until $T_d + \Delta$ and then uses the same late sampling rule. The $q = 0.5$ and $q = 0.8$ variants change the late time-retention ratio; the two Δ variants change the restoration interval. All other module and host parameters remain fixed.

Table S10: Within-campaign sensitivity on each dataset. Quality and FPS are scene means; wall time is summed over the dataset. Full uses $q = 2/3$ and $\Delta = 768$. Temporal sampling disabled, equivalent to late $q = 1$, is reported in Table S9. PSNR is in dB, SSIM and LPIPS are unitless, wall time is in seconds, and FPS is in frames/s.

Setting	PSNR	SSIM	LPIPS	Wall	FPS
		Mip-NeRF 360			
FactorGrad-GS	27.42	0.795	0.264	**2031**	574.4
Late	27.45	0.795	0.265	2100	563.1
$q = 0.5$	27.41	0.795	0.265	2057	567.7
$q = 0.8$	27.42	0.795	0.264	2063	566.3
$\Delta = 0$	27.43	0.795	0.264	2072	568.1
$\Delta = 1536$	27.43	0.796	0.264	2058	568.7
		Tanks & Temples			
FactorGrad-GS	23.92	0.835	0.215	**245**	673.8
Late	23.96	0.837	0.214	253	658.3
$q = 0.5$	23.91	0.834	0.217	247	660.7
$q = 0.8$	23.94	0.835	0.216	250	655.2
$\Delta = 0$	24.05	0.836	0.216	251	663.5
$\Delta = 1536$	23.94	0.835	0.216	248	658.2
		Deep Blending			
FactorGrad-GS	30.05	0.902	0.273	**250**	747.8
Late	30.10	0.901	0.273	258	731.9
$q = 0.5$	29.95	0.900	0.273	251	743.6
$q = 0.8$	29.78	0.899	0.272	254	732.9
$\Delta = 0$	29.93	0.900	0.272	255	726.0
$\Delta = 1536$	29.83	0.901	0.273	253	735.7

Starting the sampled path earlier. Full is faster than Late on all three datasets. Its Mip-NeRF 360 PSNR is slightly lower, while SSIM is similar and LPIPS is lower at the displayed precision. Late therefore remains a different operating point rather than an exact replacement for Full. The early channel-only stage accounts for the principal schedule difference.

Retention is not a monotonic end-to-end speed control. Reducing q from $2/3$ to 0.5 retains fewer structural backward operations, but does not yield lower measured total wall time in this campaign. Runtime also depends on density, host acceptance decisions, launch overhead, and execution variability. Conversely, a higher retention ratio does not uniformly improve final quality. The table supports the default operating point without establishing a globally optimal q.

Stabilization affects datasets differently. Removing the 768-step restoration interval improves Tanks & Temples PSNR in this execution, while Deep Blending PSNR decreases. Increasing it

to 1536 does not consistently improve quality either. These results motivate reporting the stage schedule explicitly, rather than describing it as a guaranteed response to every topology event.

B.7 Composition with Other Hosts

The principal study composes FactorGrad-GS with SkipGS. Table S11 removes that gate and evaluates the same Full schedule directly on FastGS Base. This pair uses complete 1–30K training and the same process-wall measurement as the primary study.

Table S11: Direct composition on FastGS Base, one matched execution over all three benchmarks. Quality averages scenes and wall time sums scenes. PSNR is in dB, SSIM and LPIPS are unitless, and wall time is in seconds.

Method	PSNR	SSIM	LPIPS	Wall time
FastGS (CVPR 2026)	27.36	0.819	0.255	2903
FastGS (CVPR 2026) + FactorGrad-GS	27.31	0.819	0.256	**2805**

Table S12: Additional host-composition studies, each from one matched campaign. Times are the archived native training timers and are compared within a host pair only. Vanilla uses Late; the other listed hosts use Full. These rows are not pooled with the primary repeated-execution statistics. PSNR is in dB, SSIM and LPIPS are unitless, and native training time is in seconds.

Host	Variant	PSNR	SSIM	LPIPS	Native time
Vanilla 3DGS (SIGGRAPH 2023)	Base	27.30	0.832	0.216	22098
	+ FG (Late)	27.25	0.831	0.218	**21071**
Taming-3DGS (SA 2024)	Base	27.21	0.817	0.255	4842
	+ FG (Full)	27.15	0.816	0.256	**4735**
DashGaussian (CVPR 2025)	Base	27.41	0.833	0.220	4445
	+ FG (Full)	27.34	0.832	0.222	**4416**
LeGS (ICML 2026)	Base	28.02	0.853	0.184	8256
	+ FG (Full)	28.00	0.853	0.186	**8026**

Compatibility and useful operating points. Vanilla 3DGS (Kerbl et al., 2023), Taming-3DGS (Mallick et al., 2024), DashGaussian (Chen et al., 2025), and LeGS (Ning et al., 2026) have different training cost distributions. The loss-side interface can be implemented on each host, but the resulting time reduction is not uniform. LeGS reduces its native training time while retaining nearly the same mean PSNR. DashGaussian shows a smaller reduction with a larger PSNR decrease. In the archived Vanilla study, Full was slower; the Late configuration shown here is the effective operating point. It is not evidence that one fixed Full schedule improves every host.

Shorter-GS. The separate Shorter-GS (Liu & Han, 2026) integration supplies its 26K densification endpoint to the module. Its native timer changes from 2375.1 to 2364.3 seconds, while external training-process wall time changes from 2547.1 to 2537.6 seconds. Mean PSNR changes from 27.00 to 26.95 dB. Both timers show less than 0.5% saving, consistent with a short post-densification interval and a small loss-side contribution to total training time. These measurements locate the module's useful scope more precisely than interface compatibility alone.

Resource accounting across historical hosts. Several early host-transfer runs did not record a comparable training-memory peak or an FPS benchmark. Such values are not inferred from checkpoint size or from later runs. The complete resource comparison in Table S6 uses the matched primary protocol.

C Backward-Path and Optimization Measurements

C.1 Direct Kernel Timing and Training Profiles

We measure the complete loss objective, the isolated D-SSIM path, and D-SSIM backward alone at two image resolutions. CUDA events bracket the measured region and are synchronized before elapsed time is read. Each configuration has 20 warmup repetitions and 100 measured repetitions. The image and target remain fixed during the microbenchmark; the complete objective includes the dense ℓ_1 branch with $\lambda_{\text{str}} = 0.2$.

Table S13: CUDA microbenchmark, mean $\pm$ sample standard deviation. The one-channel path still computes the exact RGB scalar. Times are in milliseconds; image resolutions are in pixels. Shading marks the one-channel path; bold marks the lower mean time in each row.

Scope	Resolution	Full RGB	One channel
Objective	800×533	0.502 ± 0.002	**0.419** ± 0.002
Objective	1600×1066	1.613 ± 0.007	**1.335** ± 0.021
D-SSIM fwd+bwd	800×533	0.380 ± 0.016	**0.277** ± 0.004
D-SSIM fwd+bwd	1600×1066	1.220 ± 0.007	**0.934** ± 0.007
D-SSIM backward	800×533	0.156 ± 0.001	**0.077** ± 0.001
D-SSIM backward	1600×1066	0.537 ± 0.006	**0.255** ± 0.001

Selected-channel backward roughly halves the isolated D-SSIM backward time. The reduction becomes smaller when full RGB forward statistics and dense ℓ_1 are included. At 1600×1066, the complete objective changes from 1.613 to 1.335 ms, whereas D-SSIM backward changes from 0.537 to 0.255 ms. On a structural-omission step, exact value-only D-SSIM evaluation costs 0.154 and 0.523 ms at the two resolutions; the value is still available to the view gate.

Table S14: Cumulative component CUDA times from complete training on Bicycle, Kitchen, Train, and Dr Johnson. Derivative-probe iterations are excluded. Times are in seconds. These instrumented regions are not the external process-wall timer. Shading marks FactorGrad-GS; bold marks the lower time in each row.

Component	SkipGS (ECCV 2026)	FactorGrad-GS
Rendering	160.05	**159.83**
Loss forward	92.78	**91.19**
Image-loss backward	73.66	**49.16**
Rasterizer backward	311.57	**309.37**
Density control	19.98	**19.88**
Optimizer	**32.96**	33.26

Where the measured saving occurs. Image-loss backward decreases from 73.66 to 49.16 seconds across the four scenes. Rasterizer backward changes much less, from 311.57 to 309.37 seconds. The direct reduction therefore occurs at the intended loss-side boundary. During active densification, an accepted FactorGrad-GS channel update takes 0.511 ms in image-loss backward, compared with 0.766 ms for SkipGS. Their rasterizer backward times are 3.375 and 3.378 ms.

Late paths and scheduling overhead. Accepted late channel updates use 0.515 ms of image-loss backward; accepted ℓ_1-only updates use 0.207 ms. SkipGS uses 0.770 ms for its retained full loss. The FactorGrad decision takes approximately 9–11 μs on the CPU for these applied paths. Component regions can include different stream and launch effects, so they are interpreted as a profile, not summed to replace the end-to-end wall-time comparison.

C.2 Gradient, Adam, and Density Evolution

Frozen-image probes compare sampled and complete RGB derivatives at the same state. The exact scalar matches the full path, and averaging a complete RGB cycle recovers its derivative up to floating-point error. Single-channel variance follows Equation S24.

Table S15: Image-gradient probes along FactorGrad-GS training. Cosine and norm ratio compare an inverse-weighted single channel with the full structural gradient. Relative variance includes the stage's time ratio; full-RGB stages are tested with a hypothetical channel estimator. Cycle error is the maximum relative discrepancy over a complete frozen RGB cycle.

Stage	Cosine	Norm ratio	Rel. variance	Cycle error
Warmup	0.577	1.730	2.00	5.84×10^{-6}
Active densification	0.576	1.729	2.00	1.07×10^{-5}
Stabilization	0.576	1.729	2.00	1.48×10^{-5}
Late	0.576	1.728	3.50	2.22×10^{-5}

Adaptive optimization. For an applied gradient $\widehat{\mathbf{g}}_t$, Adam updates its moments as

$$\mathbf{m}_t = \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\widehat{\mathbf{g}}_t, \tag{S1}$$

$$\mathbf{v}_t = \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\widehat{\mathbf{g}}_t^{\odot 2}. \tag{S2}$$

At a frozen state, derivative variance contributes to $\mathbb{E}[\widehat{\mathbf{g}}_t^{\odot 2}]$. Thus matching a first moment does not match the second-moment update. Table S16 measures this effect along the actual optimization trajectories, where the states also diverge.

Table S16: FactorGrad-GS/SkipGS ratios of parameter-group statistics in the four-scene profile. Second moment reports the square root of its mean. Values above and below one indicate changed optimizer state, not accuracy.

Stage	Group	Gradient RMS	First moment	Root second moment
Active densification	f_dc	1.01	0.99	1.04
Active densification	opacity	0.79	1.13	1.00
Active densification	rotation	0.91	1.11	1.17
Active densification	scaling	0.59	1.20	1.00
Active densification	xyz	1.01	0.99	1.00
Active densification	f_rest	0.97	1.06	1.04
Late	f_dc	1.11	1.07	1.00
Late	opacity	0.98	1.15	0.93
Late	rotation	0.98	1.10	1.02
Late	scaling	1.04	1.15	0.99
Late	xyz	1.25	1.21	1.11
Late	f_rest	1.11	1.07	1.03

Table S17: FactorGrad-GS minus SkipGS density-event counts in the same four-scene campaign. Pruned counts refer to applied pruning, not candidates.

Scene	Final Gaussians	Clone parents	Split parents	Pruned
Bicycle	-1087	+123	+7728	+8938
Kitchen	+2051	+475	+17452	+15876
Train	-7256	+2205	+2655	+12116
Dr Johnson	-2945	+6999	+4412	+14356

Gate acceptance and phase-specific density events. We additionally aggregate the archived per-iteration backward indicators and event traces (Table S18). Each run contains all 30,000 iteration records. Gate fractions count executed backward calls, not the nominal FactorGrad retention probability q. Final pruning remains active after densification ends and is included in the late-stage counts. The full-RGB warm-up already shows small between-run count differences; these paired trajectories therefore do not isolate sampling as the sole cause of every event change. Late acceptance can increase or decrease, consistent with the gate observing different current images under the same scalar-loss rule.

Table S18: Phase-wise event-count differences (FactorGrad-GS minus SkipGS) and gate acceptance fractions (S / FG). Pruned counts include both densification pruning and final pruning. The stabilization phase has no density events, but the host gate can still reject backward.

Scene	Phase	Clone parents	Split parents	Pruned	Gate S / FG
Bicycle	Warm-up	$+15$	$+463$	-2	1.000 / 1.000
Bicycle	Densification	$+108$	$+7265$	$+1642$	1.000 / 1.000
Bicycle	Stabilization	0	0	0	0.737 / 0.728
Bicycle	Late	0	0	$+7298$	0.613 / 0.617
Kitchen	Warm-up	-130	$+400$	-8	1.000 / 1.000
Kitchen	Densification	$+605$	$+17052$	$+7235$	1.000 / 1.000
Kitchen	Stabilization	0	0	0	0.788 / 0.781
Kitchen	Late	0	0	$+8649$	0.612 / 0.594
Train	Warm-up	-358	-175	$+28$	1.000 / 1.000
Train	Densification	$+2563$	$+2830$	$+8126$	1.000 / 1.000
Train	Stabilization	0	0	0	0.749 / 0.750
Train	Late	0	0	$+3962$	0.619 / 0.612
Dr Johnson	Warm-up	$+549$	$+28$	-5	1.000 / 1.000
Dr Johnson	Densification	$+6450$	$+4384$	$+7704$	1.000 / 1.000
Dr Johnson	Stabilization	0	0	0	0.721 / 0.717
Dr Johnson	Late	0	0	$+6657$	0.533 / 0.525

C.3 Training-Objective Trajectories

The archived exact reconstruction objective traces optimization progress in the four-scene mechanism campaign (Figure S2). The horizontal axis is training iteration, not elapsed time. The changing camera introduces substantial step-to-step variation; the curves summarize how the two methods progress through the same nominal stages.

Interpreting the trajectories. The plots show the rapid early decrease and the slower refinement regime. FactorGrad-GS and SkipGS follow similar broad trends, but their trajectories are not identical. The loss value is exact for each method's current image; the current images differ after sampled parameter updates. Exact-primal evaluation therefore does not require coincident curves.

Clone and split decisions also read gradient-dependent host statistics. Their counts change even though FactorGrad-GS retains the density controller (Table S17). The sampled derivative budget can therefore alter final Gaussian count and rendering FPS as well as loss-kernel cost.

Stage-averaged measurements. Across the late stage, the mean training objective on Bicycle is 0.08368 for SkipGS and 0.08488 for FactorGrad-GS. On Kitchen it is 0.02437 and 0.02488; on Train, 0.06647 and 0.06745; and on Dr Johnson, 0.02692 and 0.02762. These training objectives complement the held-out reconstruction metrics: they quantify the optimization signal that the gate observes, rather than serving as a substitute for test-image quality.

Timing and convergence answer different questions. A per-iteration objective curve assesses optimization progress under the chosen update rule. The wall-time tables measure how long that rule takes to execute. Together with the direct backward profile, they distinguish reduced structural computation from an unmeasured claim that every skipped derivative was redundant. The final quality differences are reported for every scene in Table S8.

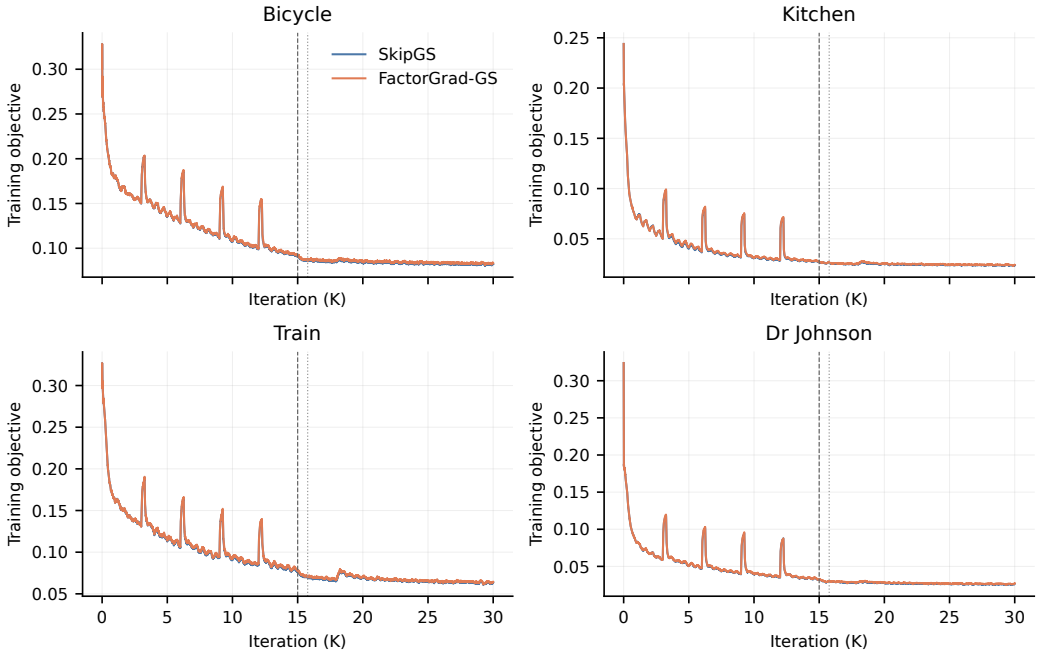

Figure S2: Training objectives for SkipGS and FactorGrad-GS on Bicycle, Kitchen, Train, and Dr Johnson, shown with a trailing mean over 256 logged samples. These are training-view losses, not held-out PSNR curves. Both methods evaluate the full RGB ℓ_1+D-SSIM scalar.

D Structural Objective and Derivative Estimator

D.1 Preliminary: 3D Gaussian Splatting

Representation and Rendering 3D Gaussian Splatting (3DGS) parameterizes a scene as a collection of N anisotropic Gaussian primitives $\mathcal{G} = \{\mathcal{G}_i\}_{i=1}^{N}$ (Kerbl et al., 2023). Each primitive $\mathcal{G}_i$ is characterized by a 3D center $\boldsymbol{\mu}_i \in \mathbb{R}^3$, a rotation matrix $\mathbf{R}_i \in \mathrm{SO}(3)$, a scale vector $\mathbf{s}_i \in \mathbb{R}^3_{>0}$, a scalar opacity $o_i \in [0,1]$, and a view-dependent color $\mathbf{c}_i(\mathbf{d}) \in \mathbb{R}^3$ represented by spherical harmonics along the viewing direction $\mathbf{d}$. To ensure valid geometric properties, the covariance matrix $\boldsymbol{\Sigma}_i$ is constructed using the standard factorization:

$$\boldsymbol{\Sigma}_i = \mathbf{R}_i \mathbf{S}_i \mathbf{S}_i^\top \mathbf{R}_i^\top. \tag{S3}$$

where $\mathbf{S}_i = \mathrm{diag}(\mathbf{s}_i)$ represents the diagonal scale matrix. Constraining $\mathbf{s}_i$ to be strictly positive guarantees that $\boldsymbol{\Sigma}_i$ remains positive definite. The spatial influence of each Gaussian at any 3D point $\mathbf{x}$ is subsequently defined by the spatial kernel $\phi_i(\mathbf{x})$ is:

$$\phi_i(\mathbf{x}) = \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_i)^\top \boldsymbol{\Sigma}_i^{-1}(\mathbf{x}-\boldsymbol{\mu}_i)\right]. \tag{S4}$$

Typically, the optimization process is initialized by seeding the primitive centers $\boldsymbol{\mu}_i$ with a sparse point cloud generated via Structure-from-Motion (SfM).

Let $\boldsymbol{\theta}$ encapsulate all trainable parameters of the Gaussian primitives. For a specific view v, the M projected Gaussians overlapping a given pixel $\mathbf{p}$ are depth-sorted in a front-to-back order, denoted by indices $i_1, \ldots, i_M$. Given the projected opacity $\alpha_{i,v}(\mathbf{p})$, the accumulated transmittance $T_{k,v}(\mathbf{p})$ evaluated immediately before the k-th primitive is formulated as:

$$T_{k,v}(\mathbf{p}) = \prod_{\ell<k} \left[1 - \alpha_{i_\ell,v}(\mathbf{p})\right]. \tag{S5}$$

The expected pixel color $\widehat{\mathbf{I}}_{\boldsymbol{\theta},v}(\mathbf{p})$ is then computed via front-to-back alpha compositing:

$$\widehat{\mathbf{I}}_{\boldsymbol{\theta},v}(\mathbf{p}) = \sum_{k=1}^{M} T_{k,v}(\mathbf{p})\alpha_{i_k,v}(\mathbf{p})\mathbf{c}_{i_k}(\mathbf{d}_{i_k,v}). \tag{S6}$$

where $\mathbf{d}_{i_k,v}$ denotes the viewing direction for primitive i_k in view v. Ultimately, the rendered image $\widehat{\mathbf{I}}_{\boldsymbol{\theta},v}$ acts as the essential computational bridge, enabling image-space reconstruction gradients to backpropagate and optimize the underlying 3D Gaussian parameters $\boldsymbol{\theta}$.

Training Objective To reconstruct the scene, the standard 3DGS framework optimizes the parameters $\boldsymbol{\theta}$ by minimizing a combination of ℓ_1 and D-SSIM losses against a target image $\mathbf{I}_v$, the scalar reconstruction loss $\mathcal{L}(\boldsymbol{\theta}; v)$ is:

$$\mathcal{L}(\boldsymbol{\theta}; v) = (1-\lambda_{\text{str}})\frac{\left\|\widehat{\mathbf{I}}_{\boldsymbol{\theta},v} - \mathbf{I}_v\right\|_1}{3|\Omega_v|} + \lambda_{\text{str}}\left[1 - \text{SSIM}(\widehat{\mathbf{I}}_{\boldsymbol{\theta},v}, \mathbf{I}_v)\right]. \tag{S7}$$

where Ω_v denotes the pixel domain and λ_{str} determines the weight of the structural dissimilarity term. Throughout training, Adam optimization (Kingma & Ba, 2015) is periodically interleaved with an adaptive density control mechanism. Within this mechanism, primitives accumulating large positional gradients are cloned or split, while transparent or oversized primitives are pruned. Because topology evolution is explicitly driven by these gradients, introducing stochastic gradient variations during active densification can permanently alter the subsequent Gaussian structure and its refinement trajectory.

D.2 SSIM Statistics and Fused Backward

Let $\widehat{\mathbf{I}}_t = \mathcal{R}(\boldsymbol{\theta}_t, v_t)$ be the rendered RGB image and $\mathbf{I}_t$ its target, with $P = HW$ pixels. Vectors are bold; losses, probabilities, and iteration indices are scalar. We use the host convention $1 - \text{SSIM}$ for D-SSIM, without an additional factor of $1/2$:

$$\mathcal{L}_t^{(1)} = \frac{\|\widehat{\mathbf{I}}_t - \mathbf{I}_t\|_1}{3P}, \qquad \mathcal{D}_{t,h} = \frac{1 - \text{SSIM}(\widehat{\mathbf{I}}_{t,h}, \mathbf{I}_{t,h})}{3}, \tag{S8}$$

$$\mathcal{D}_t = \sum_{h\in\{R,G,B\}} \mathcal{D}_{t,h}, \qquad \mathcal{L}_t = (1-\lambda_{\text{str}})\mathcal{L}_t^{(1)} + \lambda_{\text{str}}\mathcal{D}_t, \quad \lambda_{\text{str}} = 0.2. \tag{S9}$$

For one channel, write $\mathbf{x}, \mathbf{y} \in \mathbb{R}^P$ for prediction and target. Let K denote the separable 11×11 Gaussian-window operator, including the kernel's zero padding. The following products and divisions are element-wise:

$$\boldsymbol{\mu}_x = K\mathbf{x}, \qquad \boldsymbol{\mu}_y = K\mathbf{y}, \tag{S10}$$

$$\mathbf{v}_x = K(\mathbf{x}^2) - \boldsymbol{\mu}_x^2, \qquad \mathbf{v}_y = K(\mathbf{y}^2) - \boldsymbol{\mu}_y^2, \tag{S11}$$

$$\mathbf{c}_{xy} = K(\mathbf{x}\odot\mathbf{y}) - \boldsymbol{\mu}_x \odot \boldsymbol{\mu}_y. \tag{S12}$$

The SSIM map (Wang et al., 2004) and its spatial mean are

$$\mathbf{A} = \boldsymbol{\mu}_x^2 + \boldsymbol{\mu}_y^2 + C_1, \qquad \mathbf{B} = \mathbf{v}_x + \mathbf{v}_y + C_2, \tag{S13}$$

$$\mathbf{C} = 2\boldsymbol{\mu}_x \odot \boldsymbol{\mu}_y + C_1, \qquad \mathbf{E} = 2\mathbf{c}_{xy} + C_2, \tag{S14}$$

$$\mathbf{s} = \frac{\mathbf{C}\odot\mathbf{E}}{\mathbf{A}\odot\mathbf{B}}, \qquad \text{SSIM}(\mathbf{x},\mathbf{y}) = \frac{\mathbf{1}^\top\mathbf{s}}{P}, \tag{S15}$$

where $C_1 = 0.01^2$ and $C_2 = 0.03^2$. The letter $\mathbf{E}$ avoids overloading the structural loss $\mathcal{D}_t$ with a kernel intermediate.

Three cached derivative factors. The forward kernel computes the SSIM map and saves three image-sized factors. After collecting the dependence of variances and covariance on $\boldsymbol{\mu}_x$, these are

$$\mathbf{b}_\mu = \frac{2\boldsymbol{\mu}_y\odot\mathbf{E} - 2\boldsymbol{\mu}_y\odot\mathbf{C}}{\mathbf{A}\odot\mathbf{B}} - \frac{2\boldsymbol{\mu}_x\odot\mathbf{C}\odot\mathbf{E}}{\mathbf{A}^2\odot\mathbf{B}} + \frac{2\boldsymbol{\mu}_x\odot\mathbf{C}\odot\mathbf{E}}{\mathbf{A}\odot\mathbf{B}^2}, \tag{S16}$$

$$\mathbf{b}_v = -\frac{\mathbf{C}\odot\mathbf{E}}{\mathbf{A}\odot\mathbf{B}^2}, \qquad \mathbf{b}_c = \frac{2\mathbf{C}}{\mathbf{A}\odot\mathbf{B}}. \tag{S17}$$

For an incoming map gradient $\mathbf{w}$, the backward kernel evaluates

$$\mathcal{B}_h(\mathbf{w}) = K^\top(\mathbf{w} \odot \mathbf{b}_\mu) + 2\mathbf{x} \odot K^\top(\mathbf{w} \odot \mathbf{b}_v) + \mathbf{y} \odot K^\top(\mathbf{w} \odot \mathbf{b}_c). \tag{S18}$$

This is the vector–Jacobian product implemented by the three convolution branches in the fused kernel. For the normalized channel contribution $\mathcal{D}_{t,h}$, the incoming gradient is $-\mathbf{1}/(3P)$. Equation S18 therefore makes the computational boundary explicit: channel selection reduces these backward convolutions, while the RGB SSIM statistics remain complete.

D.3 Exact Value, Sampling, and Variance

Let $Z_t \in \{0, 1\}$ retain the structural derivative with marginal probability q_t, and let H_t select one channel with marginal $p_h = 1/3$. A stop-gradient construction expresses the implemented scalar/derivative separation:

$$\widetilde{\mathcal{D}}_t = \mathrm{sg}(\mathcal{D}_t) + \frac{Z_t}{q_t p_{H_t}} \big[\mathcal{D}_{t,H_t} - \mathrm{sg}(\mathcal{D}_{t,H_t})\big]. \tag{S19}$$

Its value is $\mathcal{D}_t$ for every draw. Define $\mathbf{r}_{t,h} = \nabla_{\mathrm{vec}(\widehat{\mathbf{I}}_t)} \mathcal{D}_{t,h} \in \mathbb{R}^{3P}$ by zero-extending the selected channel, and $\mathbf{r}_t = \sum_h \mathbf{r}_{t,h}$. The image derivative is

$$\begin{cases} \mathbf{u}_t^{(1)} = \nabla_{\mathrm{vec}(\widehat{\mathbf{I}}_t)} \mathcal{L}_t^{(1)}, \\ \widehat{\mathbf{r}}_t = \dfrac{Z_t \mathbf{r}_{t,H_t}}{q_t p_{H_t}}, \\ \widehat{\mathbf{u}}_t = (1 - \lambda_{\mathrm{str}}) \mathbf{u}_t^{(1)} + \lambda_{\mathrm{str}} \widehat{\mathbf{r}}_t. \end{cases} \tag{S20}$$

With renderer Jacobian $\mathbf{J}_t = \partial \,\mathrm{vec}(\widehat{\mathbf{I}}_t)/\partial \boldsymbol{\theta}_t \in \mathbb{R}^{3P \times d}$ and host acceptance indicator a_t, the parameter derivative supplied to the host is

$$\widehat{\mathbf{g}}_t = a_t \mathbf{J}_t^\top \widehat{\mathbf{u}}_t. \tag{S21}$$

The renderer still receives a dense RGB tensor, since $\mathbf{u}_t^{(1)}$ is dense. A sampled structural channel does not imply a channel-sparse rasterizer.

Frozen-state expectation. Fix the image, target, renderer Jacobian, and host decision. For a fresh sampling assignment with $\Pr(Z_t = 1, H_t = h) = q_t p_h$, averaging over that assignment gives

$$\mathbb{E}[\widehat{\mathbf{r}}_t] = \sum_h q_t p_h \frac{\mathbf{r}_{t,h}}{q_t p_h} = \mathbf{r}_t. \tag{S22}$$

This statement concerns the derivative at a fixed state. The implementation uses fixed-count permutations across time and advances its channel counter after applied updates. Conditioning on the realized history reveals part of the remaining permutation and changes its conditional probabilities; thus Equation S22 is not a claim of conditional unbiasedness along every adaptive training trajectory.

Variance and retained computation. At that same frozen state,

$$\mathbb{E}\|\widehat{\mathbf{r}}_t - \mathbf{r}_t\|^2 = \frac{1}{q_t} \sum_h \frac{\|\mathbf{r}_{t,h}\|^2}{p_h} - \|\mathbf{r}_t\|^2, \tag{S23}$$

$$\frac{\mathbb{E}\|\widehat{\mathbf{r}}_t - \mathbf{r}_t\|^2}{\|\mathbf{r}_t\|^2} = \frac{3}{q_t} - 1 \quad \text{for uniform channels and } \mathbf{r}_t \neq \mathbf{0}. \tag{S24}$$

The second line follows because channel supports are disjoint in image space. It equals 2 for channel-only sampling and 3.5 for $q_t = 2/3$. The corresponding fractions of retained channel derivatives are $1/3$ and $2/9$. These are operator budgets, not end-to-end speedups. Multiplication by $\mathbf{J}_t^\top$ mixes channel directions, so the same relative variance formula need not hold for Gaussian parameters.

Why uniform channels? For fixed positive channel norms, minimizing $\sum_h \|\mathbf{r}_{t,h}\|^2/p_h$ over probabilities gives

$$p_h^* = \frac{\|\mathbf{r}_{t,h}\|}{\sum_j \|\mathbf{r}_{t,j}\|}. \tag{S25}$$

This follows from Cauchy–Schwarz, with equality when probabilities are proportional to the norms. Computing all current norms would require the derivatives being avoided. Uniform sampling instead uses equal-cost kernels and balanced coverage without this measurement. Equation (S25) motivates an importance-sampling extension; it is not an evaluated policy in the present experiments.

Balanced assignments and optimizer state. A complete frozen RGB cycle has average $\frac{1}{3}\sum_h 3\mathbf{r}_{t,h} = \mathbf{r}_t$. For a uniformly permuted time block with k of m entries selected, $\Pr(Z_i = 1) = k/m$ and $\mathrm{Cov}(Z_i, Z_j) = -q(1-q)/(m-1)$ for $i \neq j$. The implementation realizes reproducible orderings through locally seeded permutation keys. Even with correct marginal scaling, Adam's squared-gradient state (Kingma & Ba, 2015) responds to the extra variance. Identical objective values therefore do not imply identical parameter or density trajectories.

E Implementation and Hyperparameters

Normalization in the selective autograd path. With $P = HW$ pixels, the channel contribution $\mathcal{D}_{t,h}$ includes the RGB averaging factor $1/(3P)$. For the SSIM-map backward operator $\mathcal{B}_h$ in Eq. (S18), the selected seed is

$$\mathbf{w}_{t,H_t} = -\frac{\lambda_{\mathrm{str}}}{q_t P}\mathbf{1} = \frac{1}{q_t p_{H_t}}\left(-\frac{\lambda_{\mathrm{str}}}{3P}\mathbf{1}\right), \qquad p_{H_t} = 1/3. \tag{S26}$$

Thus, normalizing by the selected channel's element count supplies the factor of three; the temporal weight supplies $1/q_t$. The output is scattered into an otherwise zero RGB tensor before adding dense ℓ_1. This is Eq. (8), not a full RGB backward followed by masking. The full forward uses the host's 11×11 separable Gaussian window (standard deviation 1.5), zero padding, $C_1 = 0.01^2$, and $C_2 = 0.03^2$.

Notation and acronyms. RGB denotes red, green, and blue; VJP denotes vector–Jacobian product; IPW denotes inverse-probability weighting. SSIM is structural similarity, and D-SSIM uses the host convention $1 - \mathrm{SSIM}$. An EMA is an exponential moving average, RMS is root mean square, and SH denotes spherical harmonics. RAD and VCAS denote randomized automatic differentiation and variance-controlled adaptive sampling. MS-SSIM denotes multi-scale SSIM. #G counts Gaussian primitives; it is not a quality score. The metric definitions are in Section 4. LR in setting tables denotes learning rate.

E.1 Execution Order and Tensor Shapes

The host renders one camera per iteration. FactorGrad-GS prepares its decision and exact-value objective from that render; SkipGS then reads the detached loss and decides whether to invoke backward. The module introduces neither a second render nor a second Gaussian optimizer. Table S19 tracks the boundary between the complete scalar calculation and the sampled derivative path.

In Figure 2, U_{host} denotes the host's optimizer and density-control operations, not an additional optimizer; $\widehat{\mathbf{u}}_t$ is the combined image adjoint in Eq. (S20).

One iteration. The following ordering matches the training loop. Host density and optimizer operations retain their original placement after the backward decision.

1. Sample a host camera and render $\widehat{\mathbf{I}}_t$ once.
2. Obtain the stage, temporal mask, channel, and weight from the controller.
3. Evaluate dense ℓ_1 and exact RGB D-SSIM. Build the selected derivative path, or detach D-SSIM on a structural-omission iteration.

Table S19: Tensor flow for a selected-channel update. $N = 1$ in the scene trainer.

Object	Shape	Operation
Render / target	$3 \times H \times W$	Host camera and rasterizer
Batched image pair	$N \times 3 \times H \times W$	Add batch axis
SSIM map and each cached factor	$N \times 3 \times H \times W$	Exact RGB forward
Exact loss	Scalar	RGB/spatial mean
Selected image and cached factors	$N \times 1 \times H \times W$	Contiguous channel slices
Selected structural image derivative	$N \times 1 \times H \times W$	Fused SSIM backward
Structural derivative buffer	$N \times 3 \times H \times W$	Scatter into selected channel
Total image derivative	$3 \times H \times W$	Add dense ℓ_1 derivative
Gaussian parameter derivatives	Host parameter shapes	Existing rasterizer backward

4. Supply $\mathrm{sg}(\mathcal{L}_t)$ to SkipGS. If accepted, call backward on the constructed objective; otherwise no reverse pass is executed.
5. Record whether the selected channel was applied. Advance the channel cycle only after an applied selected-channel backward.
6. Execute the host's density-control, parameter-update, and logging logic.

E.2 Stage and Sampling Definitions

Let $t_0 = 3000$, $T_d = 15000$, and $\Delta = 768$. The Full schedule is

$$\mathrm{mode}(t) = \begin{cases} \text{full RGB}, & 1 \le t < t_0, \\ \text{one channel}, q = 1, & t_0 \le t < T_d, \\ \text{full RGB}, & T_d \le t < T_d + \Delta, \\ \text{time–channel}, q = 2/3, & T_d + \Delta \le t \le 30000. \end{cases} \tag{S27}$$

Time blocks retain two of three global iteration positions in shuffled order. Channel blocks visit each RGB channel once over three applied selected-channel updates. The two orderings use separate deterministic keys and do not consume the host's camera-sampling random stream. The late stage has 14,233 positions, of which 9,489 retain structural derivatives in the archived schedule. A SkipGS rejection can still prevent a retained derivative from being applied.

Where runtime can decrease. Writing C_{scalar} for complete loss evaluation and $C_{s,1}$ for a selected-channel backward, the sampled stage has the schematic cost

$$C_t \simeq C_{\mathrm{render}} + C_{\mathrm{scalar}} + C_{\mathrm{schedule}} + a_t\big(C_{1,\mathrm{bwd}} + Z_t C_{s,1} + C_{\mathrm{raster,bwd}} + C_{\mathrm{update}}\big). \tag{S28}$$

The channel ratio does not scale every term. In particular, the renderer's backward remains active on an ℓ_1-only accepted update.

E.3 Resolved Host and Module Settings

The primary environment uses Python 3.7.13, PyTorch 1.12.1, and CUDA runtime 11.6; the hardware is specified in Section 4. Training starts at iteration 1 and ends at 30K; it does not resume a shared 15K checkpoint. The main comparison uses the Base densification interval of 500.

Data and evaluation. We use Mip-NeRF 360 (Barron et al., 2022), Tanks & Temples (Knapitsch et al., 2017), and Deep Blending (Hedman et al., 2018). The COLMAP camera loader sorts image names and holds out every eighth camera. The primary Base runs use the `images` directory and the host's automatic resolution setting, which downsizes images wider than 1600 pixels. PSNR and SSIM are higher-is-better; LPIPS (Zhang et al., 2018) is lower-is-better. Each scene first averages its test views; dataset quality then averages scenes.

Saved state. The controller records stage counts, selected/applied channel counts, and backward outcomes. Its resumable state includes the channel-cycle counter. The final model, command, evaluation JSON, rendering benchmark, training log, and TensorBoard events are retained per run.

Table S20: Shared settings of the primary comparison and core ablation.

Setting	Value
Iterations / images per iteration	30,000 / 1
Maximum SH degree / background	3 / black, no random background
Position learning rate	1.6×10^{-4} to 1.6×10^{-6}, scene-scale adjusted
DC SH / remaining SH learning rates	0.0025 / high-feature LR divided by 20
Opacity / scale / rotation learning rates	0.025 / 0.005 / 0.001
Adam $\beta_1, \beta_2, \epsilon$	$0.9, 0.999, 10^{-15}$
Densification start / end / interval	500 / 15,000 / 500
Opacity-reset interval	3,000
D-SSIM coefficient / constants	0.2 / $C_1 = 10^{-4}, C_2 = 9\times10^{-4}$
SSIM window / standard deviation	11×11 / 1.5
FactorGrad early start / stabilization	3,000 / 768 iterations
FactorGrad late q / channel marginals	$2/3$ / $1/3$ each
FactorGrad seed / host seed	3407 / 0
SkipGS start / warmup	15,000 / 500 iterations
SkipGS threshold / EMA decay	0 / 0.95
SkipGS minimum backward ratio	Automatic calibration from warmup, floor 0.5
SkipGS optimizer accumulation mode	Off
Checkpoint for evaluation	Final 30K representation
FPS benchmark	10 warmup frames; 3 complete test-view sweeps
Device-memory sampling period	0.2 s

Table S21: Resolved scene-specific FastGS settings shared by FastGS, SkipGS, FactorGrad-GS, and the ablations. High SH LR is the command-line value before division by 20. "Dense" is the host density-size parameter; "box mult." is the rasterizer compact-box multiplier.

Scene	High SH LR	Abs. threshold	Dense	Box mult.	Loss threshold
Bicycle	0.005	0.0012	0.001	0.5	0.1
Flowers	0.005	0.0015	0.005	0.5	0.1
Garden	0.02	0.0008	0.001	0.5	0.06
Stump	0.005	0.0015	0.004	0.5	0.1
Treehill	0.005	0.002	0.01	0.5	0.1
Room	0.02	0.0008	0.001	0.5	0.1
Counter	0.02	0.0008	0.001	0.5	0.1
Kitchen	0.02	0.0006	0.001	0.5	0.1
Bonsai	0.02	0.0006	0.001	0.5	0.1
Train	0.042	0.0015	0.01	0.7	0.1
Truck	0.04	0.0009	0.001	0.7	0.1
Drjohnson	0.0025	0.0012	0.013	0.7	0.1
Playroom	0.0015	0.0012	0.003	0.7	0.1

F Additional Qualitative Results

F.1 Mip-NeRF 360: Outdoor Scenes

The held-out view comparisons use the final models of the third primary execution (Figures S3–S5). For each scene, the selected index is $\lfloor n_{\text{test}}/3 \rfloor$ in its sorted test-render list. Selection is independent of image quality. Ground truth and prediction use the same camera, resolution, and crop; no novel-view ground truth is synthesized.

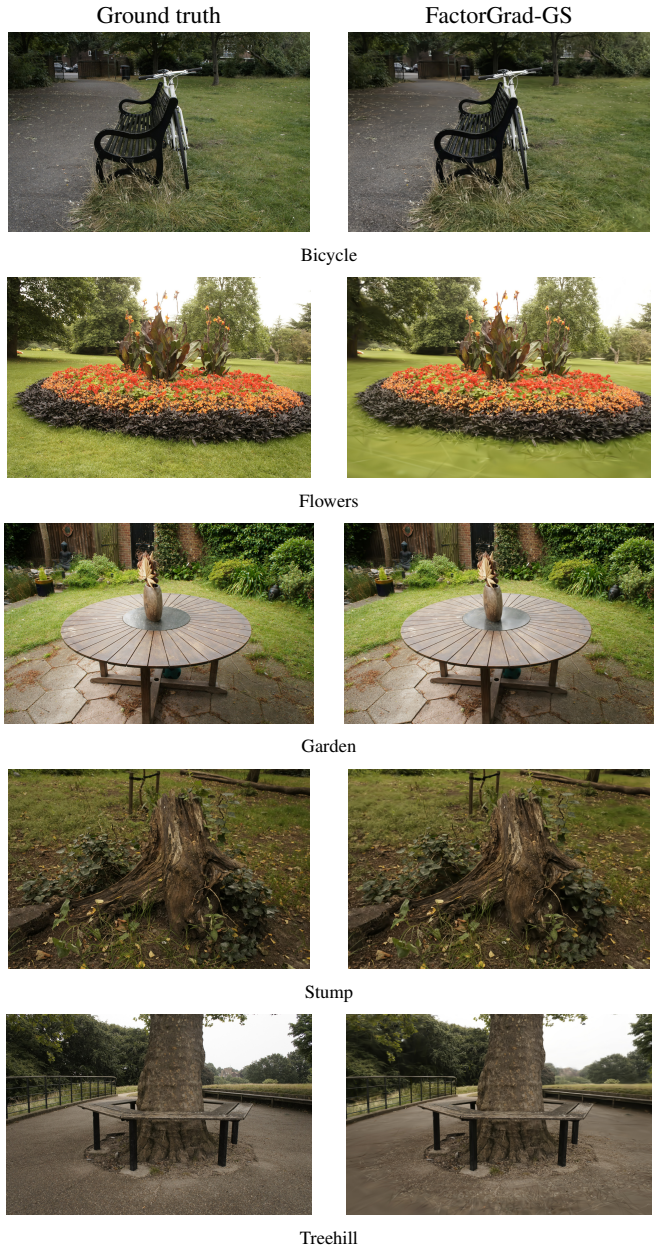

Figure S3: Outdoor Mip-NeRF 360 scenes: Bicycle, Flowers, Garden, Stump, and Treehill. Ground truth is on the left and FactorGrad-GS on the right. The images are copied from the saved test outputs without enhancement.

F.2 MIP-NERF 360: INDOOR SCENES

The same view-selection rule is applied to Room, Counter, Kitchen, and Bonsai. The displayed views include repeated textures, reflective surfaces, and thin structures. Full-scene panels retain context that can be lost when only favorable local crops are displayed.

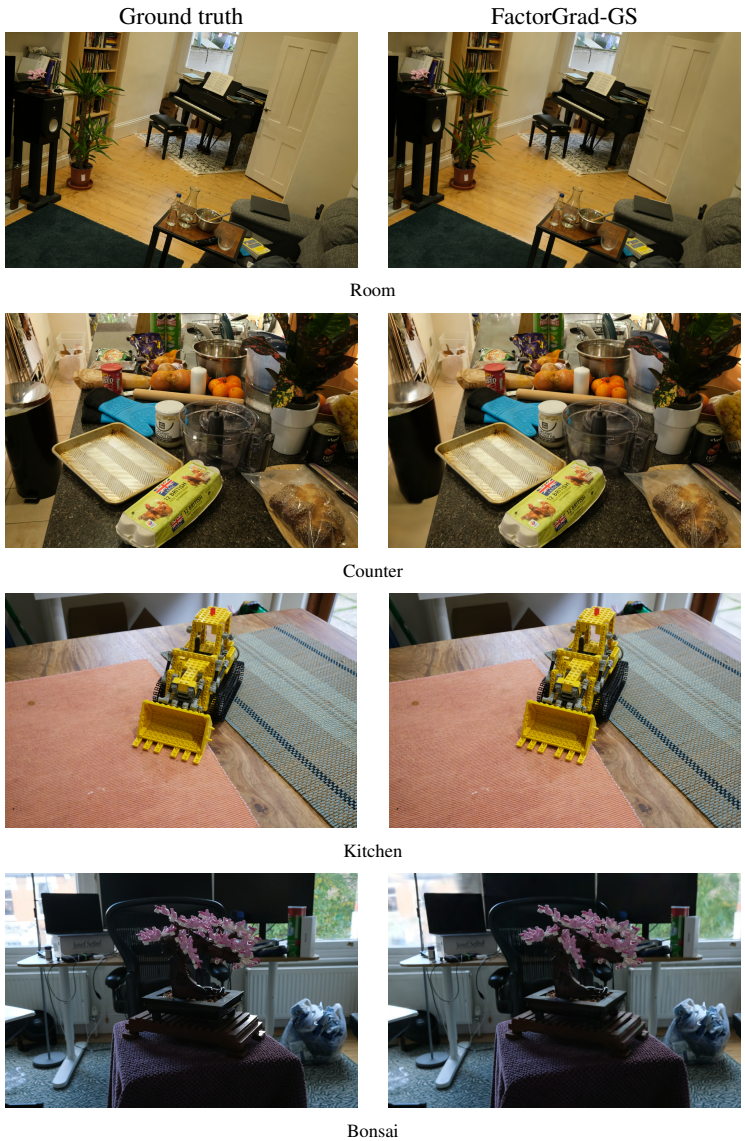

Figure S4: Indoor Mip-NeRF 360 scenes. Every pair is a held-out view from the primary Base protocol, with ground truth on the left and FactorGrad-GS on the right. Original aspect ratios are preserved.

F.3 TANKS & TEMPLES AND DEEP BLENDING

Train and Truck are shown first, followed by Dr Johnson and Playroom. These panels use the same final iteration and automatic selection rule as the Mip-NeRF 360 examples. Their corresponding numerical results are included in Tables S5 and S6.



Figure S5: Tanks & Temples and Deep Blending held-out views. Ground truth is on the left and FactorGrad-GS on the right. These are captured-camera pairs; orbit animations at interpolated camera poses do not provide additional GT.

F.4 Selected Multi-Method Local Details

The existing multi-baseline gallery supplies the local comparisons in Figure S6. Each row uses identical captured-image pixels and crop coordinates for all methods. These examples were selected to inspect local differences and are separate from the quality-independent view selection above. Their Mip-NeRF 360 renders use a different resolution configuration from the primary Base campaign; their crop appearance is not pooled into the primary numerical results.

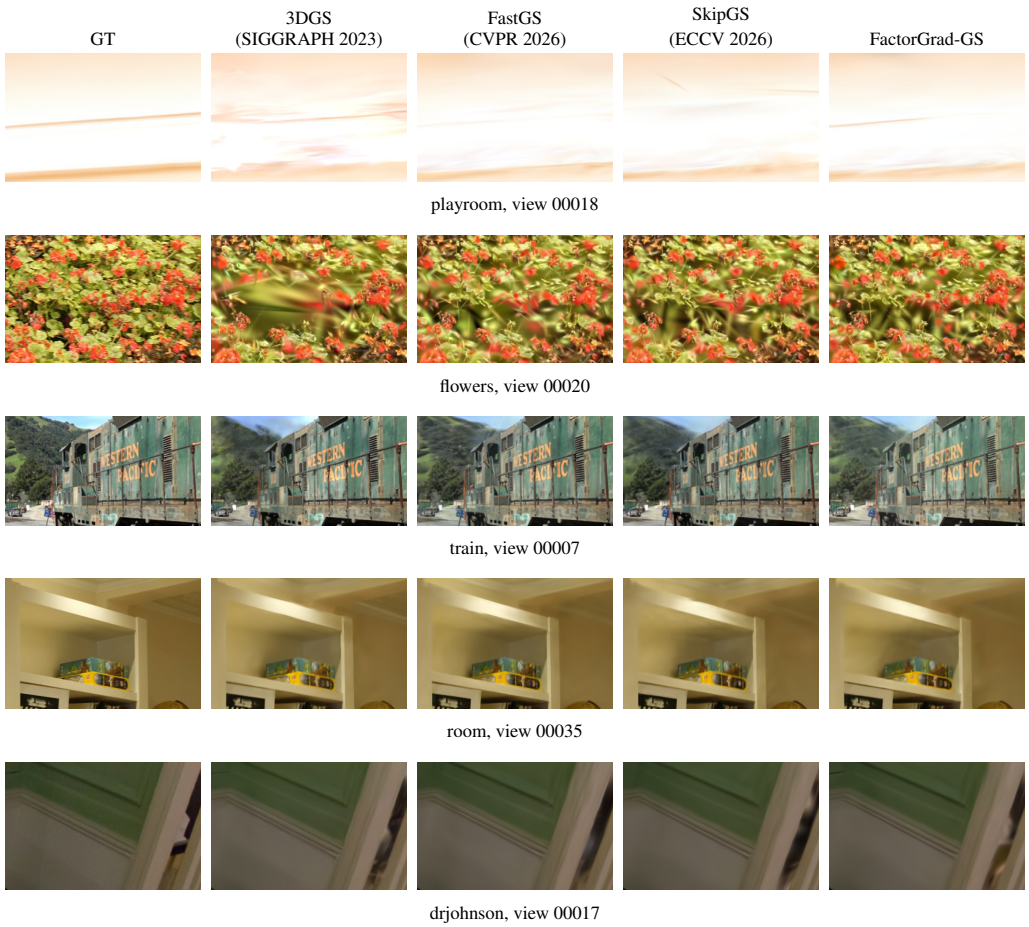

Figure S6: Saved local crops from the aligned multi-baseline gallery. Columns show GT, vanilla 3DGS, FastGS, SkipGS, and FactorGrad-GS. Rows show Playroom, Flowers, Train, Room, and Dr Johnson. A shared crop and display size are used within each row; images are not independently sharpened or rescaled in color.

Scope of the visual comparison. These panels illustrate local texture and edge differences, not a uniform ranking of the methods. The full-image measurements across all held-out views remain the basis for quantitative comparison. In particular, a locally cleaner crop does not imply a higher scene-average PSNR or lower LPIPS.

View provenance. The outdoor scenes have 25, 22, 24, 16, and 18 test cameras, respectively. The indoor scenes have 39, 30, 35, and 37; Train and Truck contain 38 and 32; Dr Johnson and Playroom contain 33 and 29. The saved camera indices and GT/render paths are retained with the figure assets. This distinguishes camera-aligned evidence from free-camera renderings used for demonstration.

F.5 Extended Multi-Method Comparison

The two scenes in Figure 4 are complemented by a Room example in Figure S7. Unlike the selected Train and Playroom details, this Room crop has higher local reconstruction error for FactorGrad-GS than for SkipGS. The visualization assets come from a separate aligned campaign with identical ground-truth pixels and test-view indices across methods. Its Mip-NeRF 360 image-resolution setting differs from the primary FastGS Base benchmark. These examples illustrate local appearance changes; their per-crop errors are not pooled with the primary three-execution results.

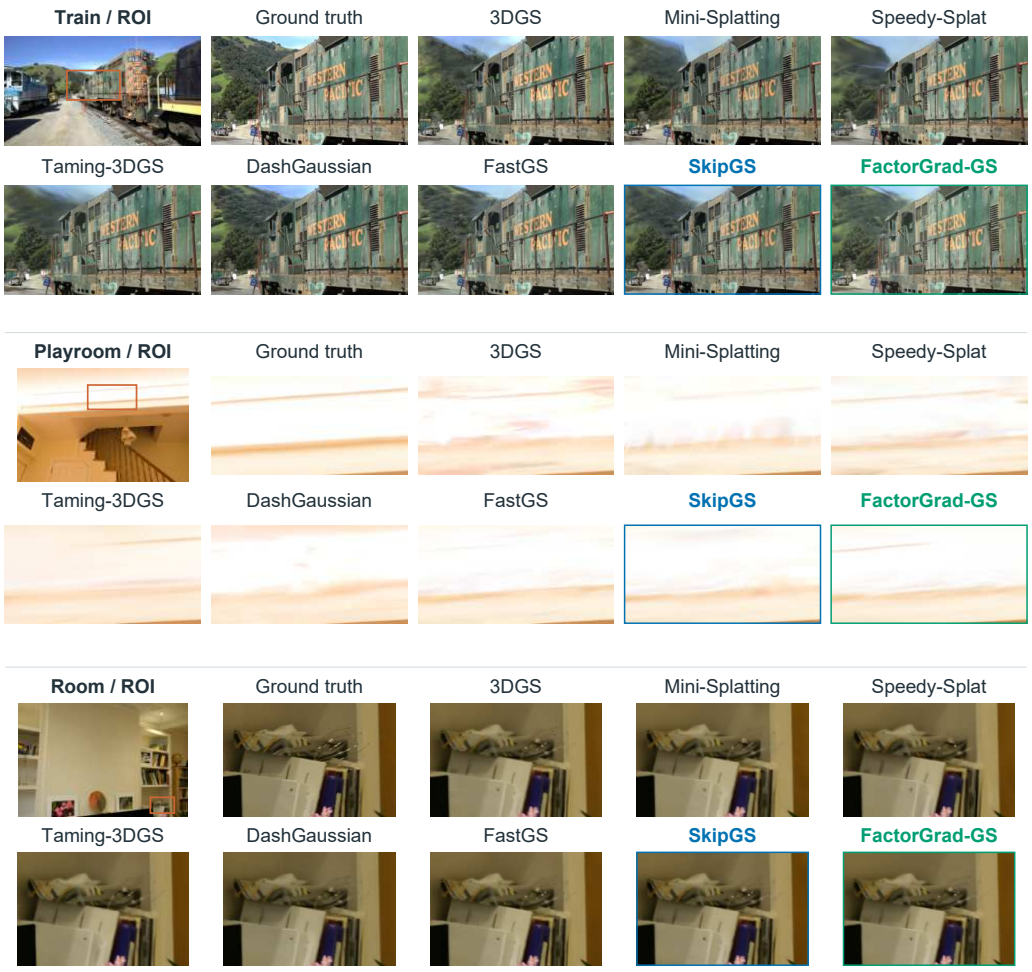

Figure S7: Extended eight-method comparison using identical native crops. Train and Playroom repeat the scenes in Figure 4; Room adds a local loss relative to SkipGS. The ground-truth context views locate each crop. All rendered pixels are unchanged.

F.6 Selected Local Error Maps

RGB error maps show the spatial distribution of reconstruction error for all three scenes in Figure S7. Crop MAE is the mean absolute RGB difference after dividing the stored 8-bit values by 255, without resizing or additional color conversion.

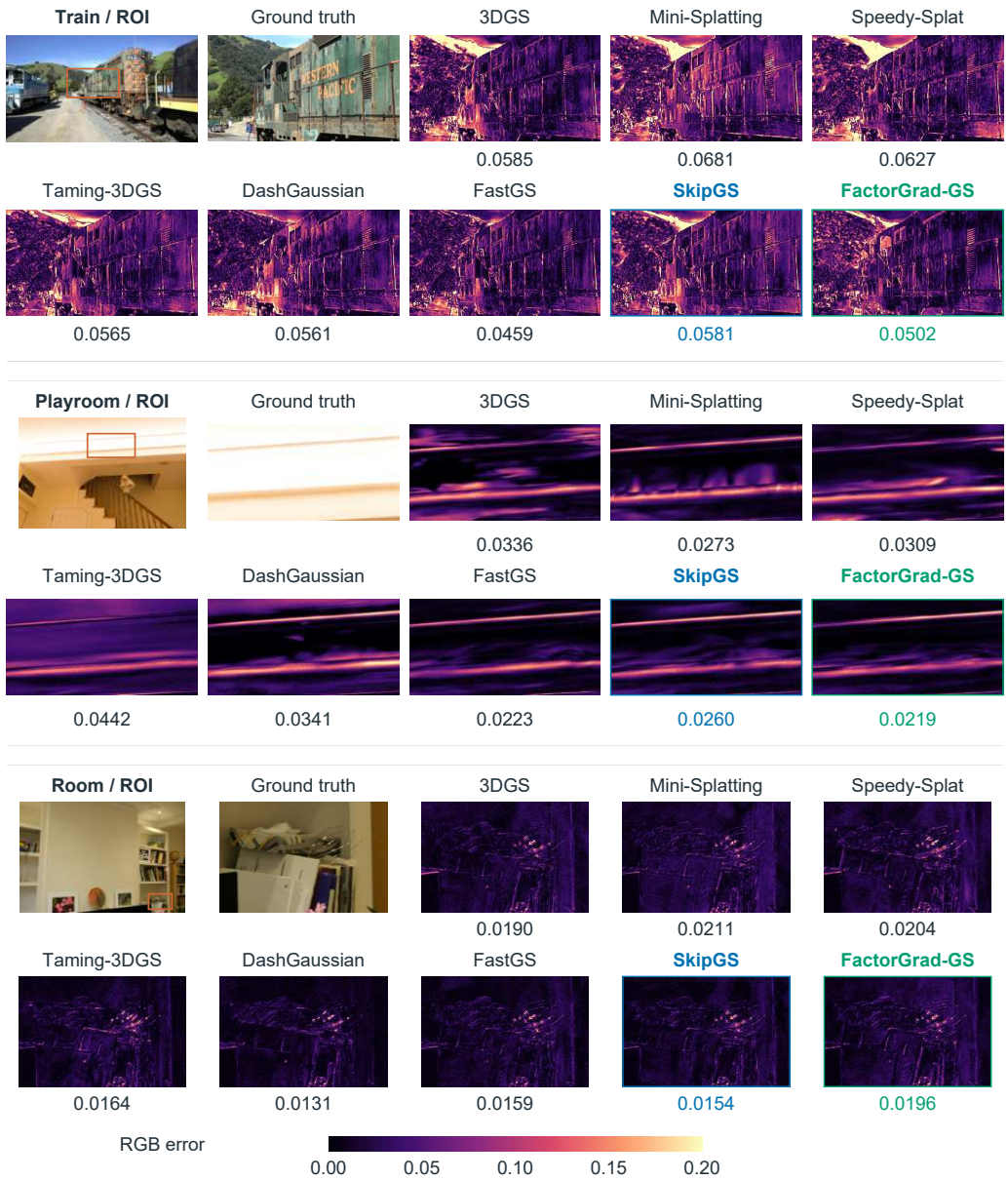

Figure S8: RGB error maps for Train, Playroom, and Room in Figure S7. The first two scenes correspond to Figure 4. Brighter pixels indicate larger mean absolute RGB error. All scenes and methods share the numeric color scale $[0, 0.20]$, with larger errors clipped for display only. Crop MAE labels use the unclipped errors.