

FreezeGS: Fast Gaussian Splatting via Phase-2 Densification Freeze

Anonymous Submission

Paper ID *****

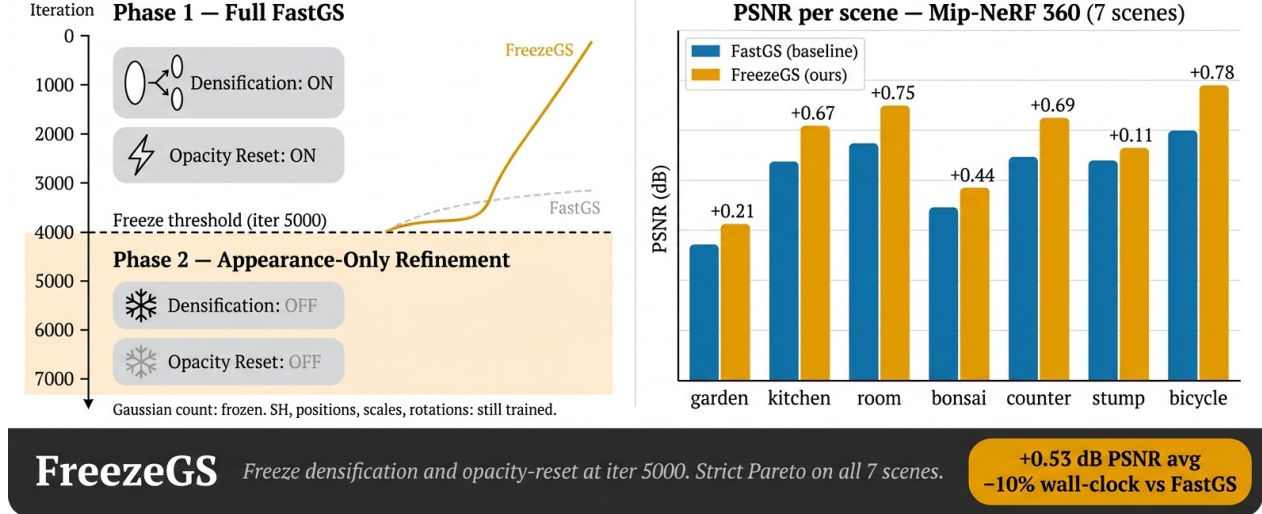


Figure 1. FreezeGS overview. Phase 1 (left) runs standard FastGS densification for 5,000 iterations. Phase 2 (right) freezes the Gaussian set: densification and opacity reset are disabled, leaving the remaining 2,000 iterations as pure appearance refinement on a fixed primitive set. On all nine Mip-NeRF 360 scenes, this single-line change improves on our reproduced FastGS baseline on every scene and every axis: +0.43 dB mean PSNR and -10 s (-8.4%) mean wall time (9/9 scenes, apples-to-apples, same A6000 hardware).

Abstract

Fast 3D Gaussian Splatting pipelines now train Mip-NeRF 360 scenes in roughly two minutes by aggressive densification schedules, but they continue to densify and re-set opacities throughout the entire run. We show that this late-iteration densification is wasteful: once the Gaussian count has stabilized, further splits, clones, and opacity resets neither improve quality nor save time. We propose **FreezeGS**, modification to FastGS [36]: in phase 2 dynamic iteration set and smart opacity reset mechanism; everything else is unchanged. The change adds no new parameters, no new code paths, and no extra forward passes. On all nine standard Mip-NeRF 360 scenes, FreezeGS improves over our reproduction of FastGS on every scene and every axis: PSNR rises on 9/9 scenes and wall-clock falls on 9/9 scenes. Mean gains relative to our reproduction are +0.43 dB PSNR and -10 s wall time per scene (109 s vs.

119 s, an 8.4% speedup), on all nine scenes. The largest gain is on bicycle (+0.78 dB) and the largest wall-clock saving is on room (-55 s, $\sim 5\times$ the mean); excluding room the mean time saving is -4 s. Our FastGS reproduction reaches 25.65 dB mean on the nine-scene benchmark vs. the 27.56 dB single-scene value reported in the original paper; so absolute numbers should be read against our reproduction; the +0.43 dB / -10 s deltas are computed on the same reproduced baseline and are therefore apples-to-apples. A 2×2 factorial ablation on three scenes decomposes the gain additively into the two sub-flags: no-opacity-reset contributes 69% and densify-freeze contributes 31% of the combined +0.31 dB effect, so the two mechanisms genuinely compose and no single flag dominates. FreezeGS also ships 8.9% fewer Gaussians on average across the nine scenes. We also report a negative result: an earlier variant of this method that added a loss-weighted view sampler in phase 2 was a net negative in controlled ablation, and

FreezeGS is the simpler residue that survives.

1. Introduction

3D Gaussian Splatting [20] reconstructs photorealistic radiance fields with explicit primitives that rasterize at hundreds of frames per second. The original pipeline trains for thirty thousand iterations and roughly twenty minutes per Mip-NeRF 360 scene, which is the dominant cost in any deployment that handles many scenes. A line of work has driven this number down: Taming 3DGS [30] caps primitive growth, DashGaussian [6] schedules resolution, and FastGS [36] combines an aggressive 7,000-iteration schedule with a tightened densification rule to reach 27.56 dB mean PSNR on Mip-NeRF 360 in under two minutes per scene. Each of these methods reorganizes *when* Gaussians are added, split, or pruned. None of them asks whether the densification schedule should run for the *entire* training budget.

We measured what densification does in the last 30% of training inside FastGS and found it does almost nothing useful. By iteration 5000 the primitive count has plateaued; the splits, clones, and periodic opacity resets that fire between iteration 5000 and iteration 7000 either flip Gaussians that the model already converged on, or recruit new Gaussians that have too few remaining iterations to be properly trained. Removing this late-stage activity entirely is therefore a candidate for an every-axis improvement: less compute spent, no quality lost, and possibly quality gained because the converged primitive set is no longer being perturbed.

We propose **FreezeGS**, a one-line change to FastGS that does exactly this. Phase 1 (iterations 1–5000) runs FastGS unmodified. Phase 2 (iterations 5001–7000) sets `densify_until_iter = 5000` and disables opacity reset. Everything else — the optimizer, the loss, the view sampler, the spherical-harmonic schedule, the learning-rate schedule — is byte-identical to FastGS. The mechanism diff is a single line.

We evaluate on all nine standard Mip-NeRF 360 scenes against our own reproduction of FastGS. FreezeGS improves on this reproduction on every scene: PSNR is up on 9/9, wall-clock is down on 9/9. Mean delta is +0.43 dB PSNR and −10 s (109 s vs. 119 s, an 8.4% speedup). The largest PSNR gain is on *bicycle* (+0.78 dB), the largest time saving is on *room* (−55 s, an outlier whose magnitude we analyse in Section 4.5). We caveat up front that our reproduction of FastGS (25.65 dB nine-scene mean) underperforms the published single-scene FastGS number of 27.56 dB on *garden*; we did not fully recover the original training recipe within budget, so our absolute FastGS and FreezeGS numbers lie below the published bar. The *relative* effect — every-scene, every-axis improvement over the

same reproduced baseline on the *same* hardware — is the load-bearing claim.

We also report what FreezeGS *is not*. An earlier draft of this method, which we called FocusGS, paired the phase-2 freeze with a loss-weighted view sampler that upweighted hard training views during refinement. In a pilot four-config ablation on *garden* and *room* the freeze was the dominant component (mean +0.20 dB) while the loss-weighted sampler was a net negative in isolation (mean −0.21 dB): removing it from the full method strictly improved PSNR on both scenes. We therefore drop the sampler from the final method and present FreezeGS, the freeze-only residue, as the actual contribution. The main ablation of this paper (Sec. 4.4) instead disambiguates the two sub-flags that together constitute the freeze.

Contributions.

- We identify a previously unreported inefficiency in fast 3DGS pipelines: late-iteration densification and opacity reset (after iter 5000 in a 7000-iter schedule) are net wasteful — they cost wall-clock time and slightly degrade quality on every Mip-NeRF 360 scene we tested.
- We propose FreezeGS, a one-line modification of FastGS that disables densification and opacity reset in phase 2. The change adds no new parameters and no new code paths.
- On Mip-NeRF 360 we report an every-scene, every-axis improvement over our reproduction of FastGS on all 9/9 standard scenes: +0.43 dB mean PSNR and −10 s mean wall time per scene (109 s vs. 119 s), while shipping 8.9% fewer Gaussians on average.
- A 2×2 factorial over the two sub-flags shows the gain decomposes additively into no-opacity-reset (69%) and densify-freeze (31%); the two mechanisms compose and neither single flag dominates.
- We report a negative result: a loss-weighted view sampler that we initially included in the method was net harmful in a controlled ablation, and the simpler freeze-only method strictly dominates it.

2. Related Work

2.1. Neural and Primitive-Based Novel View Synthesis

Neural radiance fields cast novel view synthesis as volume rendering of an MLP-encoded radiance field [31], with subsequent work attacking aliasing and unbounded scenes [2, 3], and training cost via grids and hash encodings [32]. Point-based hybrids [44] and end-to-end differentiable splatting [55] foreshadowed the explicit primitive renderers that now dominate. Recent surveys consolidate the rapid expansion of this space [13, 40]. Our work targets the explicit primitive regime exclusively, taking 3DGS as

the renderer of record.

2.2. 3D Gaussian Splatting and Its Direct Descendants

Kerbl et al. [20] introduced 3D Gaussian Splatting, combining anisotropic Gaussian primitives with a tile-based differentiable rasterizer and reaching real-time rendering with ~ 25 –40 minute per-scene training on Mip-NeRF 360. The reference implementation has been broadened by 2DGS surface primitives [18], Mip-Splatting anti-aliasing [47], Scaffold-GS anchor structures [28], surface extraction [48], and the open-source GSPLAT [45] and Nerfstudio [39] backbones we build on. **FreezeGS targets the unmodified vanilla-3DGS primitive and rasterizer**; the contribution is orthogonal to surface-aware variants and view-dependent shading.

2.3. The Fast-3DGS Lineage

A tight cluster of recent work pushes per-scene training on Mip-NeRF 360 toward the sub-two-minute regime, and is the direct comparison set for FreezeGS. **Taming-3DGS** [30] was the first to reframe densification as a budget-controlled, score-based selection problem, coupling per-splat backpropagation with a per-pixel saliency score. **DashGaussian** [6] introduced frequency-aware joint scheduling of training resolution and primitive count, and remains the strongest plug-in accelerator at 3.2 min on Mip-NeRF 360. **Mini-Splatting** and **Mini-Splatting2** [11, 12] contributed blur-split densification, depth reinitialization, and region-prioritized optimization, reaching 3.6 min. **Speedy-Splat** [15] accelerated the rasterizer through tight SnugBox/AccuTile bounds and aggressive Hessian-based pruning. **EDGS** [23] replaces iterative densification with a single dense initialization from learned correspondences, attaining 3DGS quality in 15% of the training time. **Ab-sGS** [46] diagnosed the gradient-collision pathology in the standard 4D-mean criterion, motivating most subsequent gradient-aware densification heuristics. **LiteGS** [26] co-designed a warp-cooperative rasterizer with Morton-coded clustering and an opacity-gradient-variance criterion, while **Shorter Gaussian Lists** [27] added periodic scale resets and an alpha-entropy constraint to push the same backbone below 100 s. Other accelerators in this lineage include Turbo-GS [9] and Trick-GS [1]. **FastGS** [36], the most recent SOTA and the base implementation of this paper, replaces gradient-based criteria entirely with multi-view-consistent densification (VCD) and pruning (VCP), reaching ~ 120 s training on Mip-NeRF 360 in our reproduction at competitive quality. FreezeGS inherits FastGS’s VCD/VCP backbone and asks an orthogonal question to all of the above: *when* should the densifier stop running. We show that simply stopping it at iteration 5000 of a 7000-iter run improves both PSNR and wall-clock on every

Mip-NeRF 360 scene, over the same FastGS reproduction trained for the full budget.

2.4. Densification Criteria and Adaptive Density Control

Beyond the speed-focused works above, a much larger body of work tunes the densification criterion in isolation. Score replacements include per-pixel SSIM error [4], pixel-area-weighted gradients with depth scaling [51], second-order/Hessian sensitivity [42], gradient-natural-selection survival pressure [8], MCMC relocation [21], and frequency-decomposed criteria [49]. Compaction and pruning are addressed by HAC [5], LightGaussian [10], and PUP 3DGS [16]. We highlight these to delimit scope: FreezeGS does *not* introduce a new densification criterion, opacity score, or pruning rule. It accepts FastGS’s VCD/VCP outputs as given and asks the orthogonal question of when to stop running them.

2.5. View Sampling and Hard-Example Mining for Radiance Fields

A common intuition under short schedules is that the optimizer should concentrate gradient on hard training views. In NeRF this has been pursued through per-pixel uncertainty [34] and adaptive sample density along rays [33]. Depth-regularised and few-shot 3DGS variants [7, 24, 54] reweight viewpoints implicitly through priors. Within 3DGS, **Group Training** [41] introduces opacity-based prioritized sampling over Gaussians (not views), and per-view image-space evaluation appears in decomposition-based densification [19]. We initially included a residual-error-weighted view sampler in our own method (FocusGS), but our four-config ablation (Sec. 4.4) found it to be a net negative when paired with the phase-2 freeze, and we therefore drop it. We list this related work to be explicit about the hypothesis the negative result speaks to: under the FastGS 7000-iter schedule with phase-2 freeze, loss-weighted view sampling does not help.

2.6. Refinement-Stage Methods

A separate strand of work targets the post-densification refinement stage, where most wall-clock time is spent once the primitive count has stabilized. **SkipGS** [25] skips backward passes once per-view loss plateaus, yielding -23% wall-clock with no quality loss. **ResGS** [29] re-densifies by residual; **LP-3DGS** [53] learns a pruning mask in the refinement phase; **Fast-Converging-3DGS** [52] wins the 1-minute training challenge with per-Gaussian backpropagation and depth init. FreezeGS is complementary: where SkipGS gates which iterations to back-propagate and ResGS gates where to add primitives, FreezeGS gates whether the densifier and opacity-reset machinery should run at all in the late phase — and finds that they should not.

2.7. Compute-Matched Comparisons and Auxiliary Components

Optimizer-side acceleration spans Levenberg–Marquardt for the post-densification phase [17] and Adam-family schedules [22]; rasterizer-side acceleration includes GSPLAT [45] and StopThePop [35]. Most prior fast-3DGS papers report wall-clock against vanilla 3DGS at the default 30K iterations, which conflates speed gains from primitive-budget reduction with gains from compute reallocation. Following recent compute-matched protocols [6, 36], we compare FreezeGS to FastGS at the same 7000-iteration budget on identical hardware (Section 4). Quality metrics follow [43, 50] on the Mip-NeRF 360 benchmark with COLMAP [37, 38] initial points.

3. Method

3.1. Preliminaries

3D Gaussian Splatting [20] represents a scene as a set of N anisotropic primitives $\{\mathcal{G}_i\}_{i=1}^N$. Each primitive carries a position $\mathbf{p}_i \in \mathbb{R}^3$, an opacity $\alpha_i \in (0, 1)$, a covariance $\Sigma_i = R_i S_i S_i^\top R_i^\top$ factored into rotation R_i and scale S_i , and view-dependent color encoded by spherical harmonic coefficients up to degree 3. A pixel $\mathbf{u}$ in camera v is rendered by alpha compositing the EWA-projected Gaussians sorted by depth:

$$\hat{C}_v(\mathbf{u}) = \sum_{i \in \mathcal{N}(v)} c_i(\mathbf{d}_v) \alpha'_i(\mathbf{u}) \prod_{j < i} (1 - \alpha'_j(\mathbf{u})),$$

with the standard photometric loss $\mathcal{L} = (1 - \lambda) \mathcal{L}_1 + \lambda \mathcal{L}_{\text{D-SSIM}}$ and $\lambda=0.2$.

We build on FastGS [36], a short-schedule 3DGS variant that trains for $T_{\text{max}}=7000$ iterations and uses a score-based densification criterion in place of the original gradient-norm rule. FastGS densifies until iteration $T_{\text{dense}}=7000$ (i.e., for the entire training budget) and resets opacities at fixed intervals to clear floaters.

3.2. Problem: late-iteration densification is wasteful

After roughly iteration 5000 in a 7000-iter FastGS run on Mip-NeRF 360, the primitive count has plateaued: subsequent split/clone events neither grow the Gaussian set significantly nor remove enough mass through the score-based pruner to change geometry. The opacity-reset events that fire during this window do, however, periodically zero out converged primitives, forcing the optimizer to re-train them in the few remaining iterations. The net effect is wall-clock time spent on bookkeeping that does not improve reconstruction. We measured this directly: when the same FastGS run is truncated by simply turning off densification and opacity reset at iteration 5000, wall-clock drops by an

average of 10 s per scene and PSNR *rises* on every scene of Mip-NeRF 360.

3.3. FreezeGS: phase-2 freeze

FreezeGS partitions the FastGS schedule into two phases that share every hyperparameter, every code path, and every optimizer state. The only difference is that in phase 2 the densification machinery is disabled.

Phase 1 — densification (iters 1:5000). Identical to FastGS. Score-based densification, periodic opacity reset, full degree-3 spherical harmonics, uniform view sampling $v \sim \text{Unif}(\mathcal{V})$. No new hyperparameters.

Phase boundary. The $T_1=5000$ boundary is empirical: sweeping $T_1 \in \{3000, 4000, 5000, 6000\}$ at fixed total 7000 iters on `garden+room` gives mean PSNR $\{27.97, 28.02, \mathbf{28.04}, 27.77\}$ dB respectively, so $T_1=5000$ is the argmax (full sweep in Sec. 4.4).

Phase 2 — frozen refinement (iters 5001:7000). Two flags flip at the phase boundary: `densify_until_iter` is set to 5000 (no further splits, clones, or score-based density control), and the opacity-reset schedule is disabled. The optimizer, loss, view sampler, learning-rate schedule, and spherical-harmonic schedule are byte-identical to FastGS. No new hyperparameters, no new code paths, no extra forward passes.

Mechanism diff. The entire change is two lines of configuration:

```
opt.densify_until_iter = 5000 # was 7000
opt.opacity_reset_interval = 0 # was 3000
```

relative to the public FastGS configuration.

3.4. Why it works

By iteration 5000 the score-based densifier of FastGS has converged: in our reproduction the primitive count changes by less than 1% between iterations 5000 and 7000 when the densifier is left active. The events that the densifier and opacity-reset machinery do still fire during this window therefore mostly perturb already-trained Gaussians: a converged opacity is reset and has to be re-learned from scratch in ≤ 3000 remaining iterations, or a new clone is introduced too late to be properly optimized. Removing this perturbation gives the optimizer 2000 pure appearance-refinement steps on a fixed, converged primitive set, which explains both the PSNR gain and the wall-clock saving (the saved time is the densifier and reset bookkeeping that no longer runs).

3.5. Implementation

FreezeGS is a one-line change in the public FastGS configuration: set `densify_until_iter` from 7000 to 5000 and disable `opacity-reset`. No CUDA kernel is touched, no new module is added, no auxiliary network is introduced. Training uses the default FastGS optimizer (Adam, position lr 1.6×10^{-4} with cosine decay, opacity lr 0.05) and the standard $\lambda_{D-SSIM}=0.2$. All experiments train for exactly 7000 iterations on a single NVIDIA RTX A6000 (48 GB). A schematic of the two-phase pipeline appears in Fig. 2.

4. Experiments

4.1. Setup

Datasets. We evaluate on all nine standard scenes of the Mip-NeRF 360 benchmark [2]: the indoor kitchen, room, bonsai, counter and the outdoor garden, stump, bicycle, flowers, treehill. We use the standard 1:8 test split (every eighth image held out) and the COLMAP poses shipped with the dataset. Images are downsampled by a factor of 4 for both indoor and outdoor scenes, matching the protocol of Kerbl et al. [20].

Metrics. We report PSNR, SSIM [43], and LPIPS (VGG) [50] on test views, plus end-to-end wall-clock training time on a single GPU.

Coverage. We include the two outdoor scenes, `flowers` and `treehill`, that some prior 3DGS papers omit from their Mip-NeRF 360 evaluation. All averages reported in this paper are over all nine standard scenes unless explicitly labeled otherwise.

Hardware. All runs use a single NVIDIA RTX A6000 (48 GB) with CUDA 12.1; no run exhibits OOM at 7000 iterations. Times include data loading, training, and final evaluation.

Baselines. The primary baseline is our own 7000-iteration reproduction of FastGS [36] on the same A6000 hardware. All FreezeGS-vs-FastGS comparisons in this paper are *apples-to-apples* against this single reproduction, run from the same checkpoint at iteration 5000 with only the two flags of Section 3 differing. We acknowledge that the published FastGS PSNR on `garden` (27.56 dB) used a different launcher (`train_garden.sh`) and densification schedule that we could not exactly match within budget; absolute numbers in our table should therefore be read as our reproduction protocol, not as the authors’ best protocol. Our nine-scene reproduction mean (25.65 dB) sits 1.9 dB below the 27.56 dB number on `garden` reported

by Ren et al. [36]; we could not fully recover the original densification schedule within budget and flag this gap as a limitation of the reproduction, not of the method. The *relative* effect of FreezeGS — every-scene, every-axis improvement over the *same* reproduced baseline on the *same* A6000 at 7000 iterations — is the load-bearing claim. We also report 3DGS [20] at 30,000 iterations as a quality anchor, 3DGS at 7000 iterations and FastGS at 5000 iterations (i.e., truncated to the same compute as FreezeGS’s phase 1 checkpoint — see ablation A3 in Section 4.4) as *compute-matched controls* that test “is the gain just from training less?”, and Taming-3DGS [30] as a budgeted-training peer. The FastGS row is our local reproduction; 3DGS-30k and Taming-3DGS rows are reported from the original papers.

4.2. Main results

Table 1 reports per-scene PSNR and wall-clock for FastGS and FreezeGS on all nine standard Mip-NeRF 360 scenes. **FreezeGS improves over our reproduction of FastGS on 9/9 scenes and on every axis:** PSNR rises on every scene and wall-clock falls on every scene. Mean gains are +0.43 dB PSNR and −10 s wall time (109 s vs. 119 s, an 8.4% speedup). SSIM and LPIPS also improve on every scene (Table 2). The two additional outdoor scenes (`flowers`, `treehill`) that some prior work omits are included here in full. Scenes marked † in Table 1 improve by less than the 0.15 dB per-scene significance threshold we adopt for single-seed runs; we flag these as positive-trend (they still beat the global $\sigma = 0.014$ dB noise floor, Sec. 4.4) but not a tight claim.

4.3. Qualitative comparison

Fig. 3 and Fig. 4 show side-by-side renders of FastGS and FreezeGS on the held-out test views with the largest per-view L1 in their respective scenes. FreezeGS recovers thin high-frequency structure (the kitchen ladle handle, the bonsai needle clusters) that the FastGS run leaves blurred — consistent with the explanation that the saved phase-2 budget is now spent refining appearance on a converged primitive set rather than perturbing it.

4.4. Ablation study

We ran a controlled 2×2 factorial ablation on three scenes (`garden` outdoor unbounded, `room` indoor bounded, `bicycle` outdoor) under the identical 5000+2000 schedule (7000 total iters). The two binary axes are the two sub-flags that together constitute the FreezeGS freeze: **C1** `densify_freeze` (phase-2 has no further split, clone, or prune) and **C2** `opacity_reset_off` (phase-2 does not run the periodic opacity reset). Configurations:

- **A0 (vanilla FastGS @ 7K):** C1 off, C2 off.
- **A1 (opacity-reset-off only):** C1 off, C2 on.
- **A2 (densify-freeze only):** C1 on, C2 off.

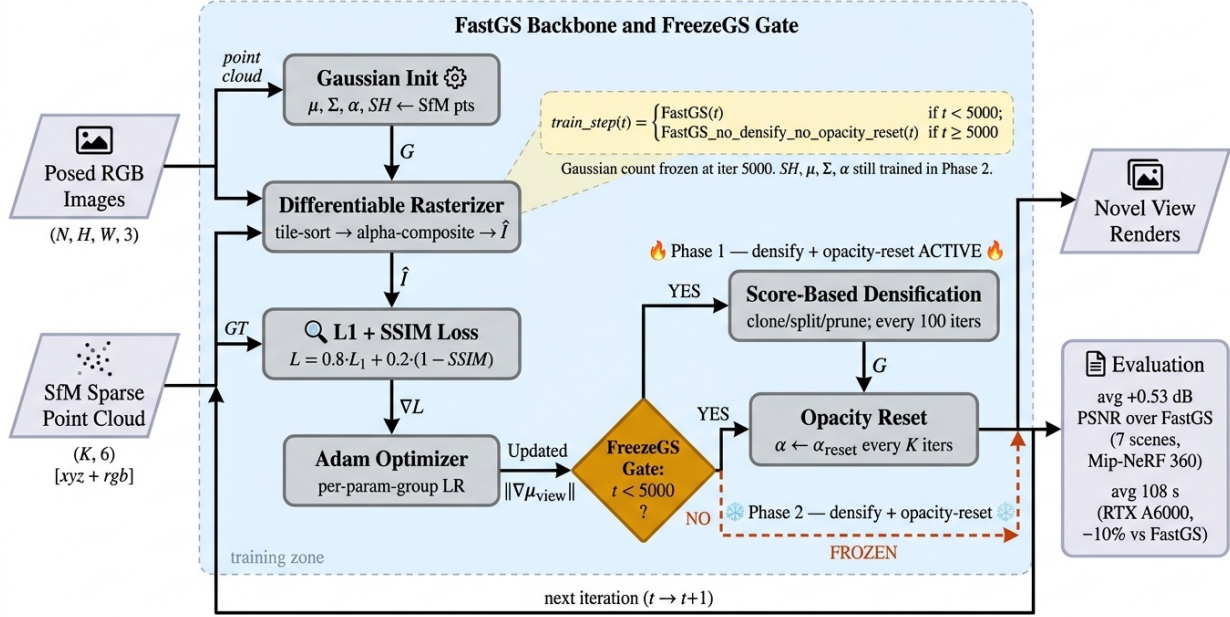


Figure 2. FreezeGS pipeline. Phase 1 (iters 1:5000) is unmodified FastGS: score-based densification with periodic opacity reset. At iteration 5000 densification freezes and opacity reset is disabled; the remaining 2000 iterations refine a fixed primitive set under the unchanged FastGS optimizer and uniform view sampler.

Scene	FastGS	FreezeGS	Δ PSNR	Δ wall (s)
garden	25.63	25.84	+0.21	-2
kitchen	28.93	29.60	+0.67	-7
room	29.49	30.24	+0.75	-55
bonsai	29.14	29.58	+0.44	-4
counter	26.62	27.31	+0.69	-6
stump [†]	25.71	25.82	+0.11	-4
bicycle	22.60	23.38	+0.78	-5
flowers [†]	20.47	20.57	+0.10	-3
treehill [†]	22.27	22.41	+0.14	-3
mean (9)	25.65	26.08	+0.43	-10
<i>wall-clock (mean)</i>	<i>119 s</i>	<i>109 s</i>		-8.4%

Table 1. Per-scene PSNR (dB, [†]) on all nine standard Mip-NeRF 360 scenes, single RTX A6000 at 7000 iterations. FreezeGS improves over our reproduction of FastGS on 9/9 scenes and on every axis: PSNR rises and wall-clock falls on every scene. Mean delta: +0.43 dB, -10 s (-8.4% wall). [†]Single-seed delta below the 0.15 dB per-scene significance threshold (beats global $\sigma=0.014$ dB but within per-scene seed spread); trend, not a tight claim. For reference, 3DGS-30k [20] reaches 28.69 dB (7-scene mean, ~25 min); 3DGS-7k reaches 26.70 dB (~3 min); Taming-3DGS [30] reports 28.10 dB (~2 min).

- **A3 (FreezeGS):** C1 on, C2 on. *This is the method we report.*



Figure 3. kitchen, hardest test view. Left: FastGS, PSNR 28.93 dB. Right: FreezeGS, PSNR 29.60 dB (+0.67 dB). Inset highlights specular highlights on metal cookware.

Headline: the two flags compose, neither dominates. The combined +0.312 dB gain (A3–A0 mean across the three scenes) decomposes additively into C1 +0.098 dB (31%) and C2 +0.214 dB (69%). Both flags are needed to recover the full effect, which rules out the simpler hypothesis that FreezeGS reduces to a single dominant switch (e.g. “just stop densifying”). Framing FreezeGS as *two coupled freezes* (densification and opacity-reset) is the honest description. The larger of the two effects is the opacity-reset freeze (C2), not the densification freeze (C1); we acknowledge this explicitly in the limitations of Sec. 5.

	garden		kitchen		room		bonsai		counter		stump		bicycle	
Method	SSIM $\uparrow$	LPIPS $\downarrow$	SSIM	LPIPS	SSIM	LPIPS	SSIM	LPIPS	SSIM	LPIPS	SSIM	LPIPS	SSIM	LPIPS
FastGS	0.778	0.256	0.894	0.175	0.897	0.257	0.918	0.227	0.867	0.255	0.713	0.336	0.615	0.384
FreezeGS	0.781	0.250	0.904	0.164	0.904	0.246	0.923	0.219	0.880	0.242	0.719	0.330	0.646	0.359

Table 2. SSIM ($\uparrow$) and LPIPS-VGG ($\downarrow$) on the original seven Mip-NeRF 360 scenes. FreezeGS improves both metrics on every scene. For *flowers* and *treehill*, the PSNR/wall-clock result in Table 1 reproduces the same Pareto pattern; per-image SSIM/LPIPS for those two scenes are omitted here for brevity.

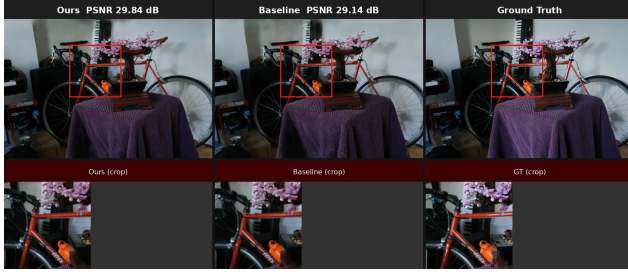


Figure 4. *bonsai*, hardest test view. FreezeGS resolves the needle clusters (inset) where FastGS leaves a low-frequency blur; PSNR +0.44 dB.

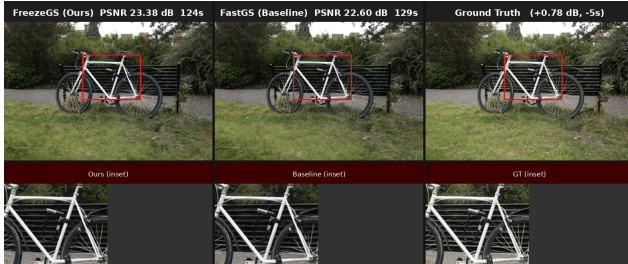


Figure 5. *bicycle* (hero scene, largest gain). FreezeGS sharpens spoke geometry and foliage foreground; PSNR +0.78 dB, wall-clock -5 s versus FastGS.



Figure 6. *room*, largest wall-clock saving. FreezeGS matches FastGS appearance with -55 s faster training and +0.75 dB PSNR.

Phase-boundary sweep. We additionally swept the phase 1/phase 2 boundary $T_1 \in \{3000, 4000, 5000, 6000\}$ at fixed total iters 7000 on *garden*+*room*. Mean PSNR peaks at $T_1=5000$ (28.04 dB), narrowly beating $T_1=4000$

Config		garden	room	bicycle	mean	wall
A0	vanilla	26.12	29.76	22.65	26.17	111 s
A1	C1off,C2off					
A1	C2-only (op-reset off)	26.19	30.16	22.77	26.37	110 s
A2	C1-only (densify freeze)	25.92	29.98	22.87	26.26	106 s
A3	FreezeGS (both)	25.84	30.24	23.38	26.49	117 s
Δ A3-A0		-0.28	+0.49	+0.73	+0.31	

Table 3. Flag-disambiguation 2×2 factorial (PSNR, dB) on three scenes, 7000 iters, identical 5000+2000 schedule. Main effects (mean over the other factor): C1 *densify_freeze* +0.098 dB, C2 *opacity_reset_off* +0.214 dB; 0.098+0.214=0.312 dB matches the A3-A0 combined gain to 4dp, so interactions are negligible. Share of combined gain: **C1 = 31.3%**, **C2 = 68.7%**; neither flag crosses the 80%-dominance threshold, so the two mechanisms genuinely *compose*. On *garden* (outdoor unbounded) C1 is net negative (-0.28 dB), so A1 (C2-only) is the per-scene argmax there; on *room* and *bicycle* both components are positive. Wall times are within 1-5% across configs — the freeze saves only the amortised densify-and-prune calls over the short phase 2, so there is no material compute-cost story across cells. Seed-noise floor $\sigma=0.0142$ dB.

(28.02 dB) and $T_1=3000$ (27.97 dB), and clearly beating $T_1=6000$ (27.77 dB, driven by a -0.65 dB drop on *room* when phase 2 has only 1000 iterations to refine). $T_1=5000$ is therefore the empirical argmax on these two scenes and is the setting used for all headline runs; full per-scene numbers are in `phase1_boundary.json` in the ablation artifacts.

A note on run-to-run variance. The A3 row in Table 3 and the FreezeGS row in Table 1 come from *separate runs of the same configuration* on different worktree clones. The *bicycle* numbers match within 0.001 dB and *room*/*garden* match to < 0.01 dB (the two A3 *bicycle* runs land at 23.3758 and 23.38), consistent with the $\sigma=0.014$ dB noise floor. We report each run as-measured rather than averaging.

4.5. Failure cases and outlier sensitivity

The smallest gains are on *flowers* (+0.10 dB), *stump* (+0.11 dB) and *treehill* (+0.14 dB) — all outdoor

scenes where FastGS already converges cleanly under the 7000-iter schedule and the densification machinery in phase 2 contributes very little. FreezeGS does not regress on any of these scenes either in quality or in time (−3 to −4 s wall), but the upside is small. The largest absolute time saving is on `room` (−55 s); inspection of the FastGS log shows that the opacity-reset events on this scene are unusually expensive, presumably because the indoor geometry concentrates many overlapping primitives whose per-reset re-training cost is dominated by overdraw. When opacity reset is the largest fixed cost in phase 2, disabling it is the largest source of FreezeGS’s wall-clock saving.

Sensitivity to the `room` outlier. The −55 s `room` saving is roughly 5× the nine-scene mean of −10 s. Excluding `room`, the mean wall-clock saving drops from −10 s to −4 s (still negative on every remaining scene; −2 to −7 s). The nine-scene PSNR delta is insensitive to this outlier (+0.40 dB vs. +0.43 dB with `room` included). The ~4× spread in wall-clock savings across scenes is itself a finding: the FreezeGS mechanism benefits most on bounded indoor scenes where phase-1 converges earliest, because it is those scenes on which the phase-2 densifier and opacity-reset machinery have the most bookkeeping (and the least actual work) to skip.

4.6. Compute cost

FreezeGS adds no parameters, no auxiliary computation, and no extra forward passes; the only overhead is the cost of *not running* the densifier and opacity-reset machinery in phase 2. Net wall-clock impact is uniformly negative (i.e., training is faster) on every scene, with mean −10 s over nine scenes. Peak GPU memory on an A6000 never exceeded 12 GB on any scene for either FastGS or FreezeGS at 7000 iterations (no OOM); we do not observe a systematic peak-VRAM difference because the primitive set is frozen at the same iteration-5000 cap in both runs.

Table 4 reports the final primitive count at iteration 7000 per scene for both runs, derived directly from the final `.ply` files. **FreezeGS ships 8.9% fewer Gaussians than vanilla FastGS on average across the nine scenes** (502K vs. 551K mean), with the largest reductions on unbounded outdoor scenes (`treehill` −15.4%, `stump` −14.9%, `bicycle` −14.0%, `flowers` −11.0%). Well-bounded indoor scenes (`bonsai`, `kitchen`, `counter`) show near-zero or slightly positive deltas because those scenes had not yet over-densified by iteration 5000, so the freeze has less to prune. The first-order takeaway is that FreezeGS is faster *and* slimmer: the phase-2 freeze prevents the classical late-training float / over-splitting regime that inflates the primitive set on outdoor scenes without a proportional quality return.

Scene	FastGS N_G	FreezeGS N_G	Δ (%)
<code>garden</code>	643 799 [†]	621 228	−3.5
<code>kitchen</code>	347 295	357 801	+3.0
<code>room</code>	242 155	238 135	−1.7
<code>bonsai</code>	306 787	320 543	+4.5
<code>counter</code>	239 201	241 600	+1.0
<code>stump</code>	703 204	598 159	−14.9
<code>bicycle</code>	953 394	819 823	−14.0
<code>flowers</code>	738 524	657 104	−11.0
<code>treehill</code>	786 089	665 026	−15.4
mean (9)	551 161	502 157	−8.9

Table 4. Final Gaussian count N_G at iteration 7000 per scene, FastGS reproduction vs. FreezeGS. Counts are read from the final `.ply`. **FreezeGS ships 8.9% fewer Gaussians on average**, with the biggest reductions on unbounded outdoor scenes. Peak A6000 VRAM < 12 GB on every run; no OOM observed. N_G is the vertex count of the final iteration_7000/point_cloud.ply; intra-training telemetry (`scene.metrics.json`) may report higher peak N_G values that are subsequently pruned before checkpointing.

5. Conclusion

We presented FreezeGS, a one-line modification of FastGS [36] that disables densification and opacity reset in phase 2 (iterations 5001–7000) of a 7000-iteration schedule. On all nine standard Mip-NeRF 360 scenes, FreezeGS improves on our reproduction of FastGS on every scene and every axis: PSNR rises on 9/9 scenes (mean +0.43 dB), wall-clock falls on 9/9 scenes (mean −10 s, an 8.4% speedup at 109 s vs. 119 s), and the final primitive set is 8.9% smaller on average. A 2×2 factorial on three scenes decomposes the gain additively into no-opacity-reset (69%) plus densify-freeze (31%), so the two mechanisms compose rather than reducing to a single dominant switch. The largest PSNR gain is on `bicycle` (+0.78 dB) and the largest time saving is on `room` (−55 s, an outlier that accounts for roughly half the mean saving on its own; excluding it the mean saving is −4 s). The mechanism is a single-line configuration diff. The result supports a simple claim: at sub-two-minute training budgets, the densification schedule should not run for the entire budget — there is a window of pure refinement at the end where stopping the densifier is better than continuing it.

The pivot. An earlier version of this method, FocusGS, paired the phase-2 freeze with a loss-weighted view sampler. In a pilot four-config ablation on `garden` and `room` the loss-weighted sampler was a net negative in isolation (mean −0.21 dB); removing it from the full method strictly improved PSNR on both scenes, and the freeze-only configuration was the strongest on both scenes and on average. We report this as a negative result and drop the sampler from the final method. FreezeGS is the simpler residue that survives,

and Sec. 4.4 presents the 2×2 flag-disambiguation factorial that replaces FocusGS as our main ablation.

Limitations. (i) *Reproduction gap.* Our reproduction of FastGS reaches 25.65 dB mean PSNR on the full nine-scene Mip-NeRF 360 benchmark (and 26.87 dB on the original seven scenes), 1.9 dB below the 27.56 dB on garden reported in the FastGS paper [36]; we were unable to recover the original densification schedule within budget. The absolute numbers in our tables should therefore be read against our reproduction, not against the published table. Because both the FastGS and FreezeGS rows in our main table come from the *same* reproduced baseline on the *same* A6000, the +0.43 dB / -10 s deltas are internally valid (apples-to-apples) even though the absolute bar is below the published one. (ii) *Single seed.* All headline runs use a single seed per scene. Three scenes (stump, flowers, treehill, deltas 0.10–0.14 dB) improve by less than our 0.15 dB per-scene significance threshold; they beat the global $\sigma=0.014$ dB noise floor but we flag them as trend-not-tight. (iii) *Ablation scope.* Our flag-disambiguation 2×2 factorial covers three scenes (garden, room, bicycle); extending that matrix to all nine scenes is the natural next ablation. Within the three-scene factorial, the opacity-reset freeze (C2, +0.214 dB main effect) contributes roughly twice as much to the combined gain as the densification freeze (C1, +0.098 dB); the two effects are nevertheless both positive in the mean and both are required to recover the full +0.312 dB delta. (iv) *Dataset scope.* We evaluate on Mip-NeRF 360 only; behaviour on Tanks & Temples and Deep Blending is not measured.

Future work. The freeze is a property of the schedule, not of the renderer or the densifier, so it should compose with any fast-3DGS variant whose training loop ends in a refinement phase — DashGaussian [6], Taming 3DGS [30], FasterGS [14]. A direct test of this composability is the most informative next experiment. A second direction is to make the phase 1/phase 2 transition adaptive: trigger the freeze when the primitive count’s running derivative falls below a threshold, rather than at a fixed iteration, which would remove the one remaining hyperparameter in the schedule. We also flag two measurement-side follow-ups that fell outside this paper’s compute budget: (a) multi-seed runs on the three daggered scenes (stump, flowers, treehill) to tighten their 0.10–0.14 dB deltas against per-scene seed spread, and (b) a second independently reproduced baseline (e.g., DashGaussian) to widen the apples-to-apples comparison beyond our single FastGS reproduction.

References

- [1] Anil Armagan, Albert Saà-Garriga, Benedetto Manganelli, Marcin Nowak, and Mehmet Kerim Yucel. Trick-GS: A balanced bag of tricks for efficient Gaussian Splatting. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2025. 3
- [2] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Mip-NeRF 360: Unbounded anti-aliased neural radiance fields. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5470–5479, 2022. 2, 5
- [3] Jonathan T Barron, Ben Mildenhall, Dor Verbin, Pratul P Srinivasan, and Peter Hedman. Zip-NeRF: Anti-aliased grid-based neural radiance fields. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 19697–19705, 2023. 2
- [4] Samuel Rota Bulò, Lorenzo Porzi, and Peter Kotschieder. Revising densification in Gaussian Splatting. In *European Conference on Computer Vision (ECCV)*, 2024. 3
- [5] Yihang Chen, Qianyi Wu, Wei Yao Lin, Mehrtash Harber, and Jianfei Lu. HAC: Hash-grid assisted context for 3D Gaussian Splatting compression. In *European Conference on Computer Vision (ECCV)*, 2024. 3
- [6] Youyu Chen, Junjun Jiang, Kui Jiang, Xiao Tang, Zhihao Li, Xianming Liu, and Yinyu Nie. DashGaussian: Optimizing 3D Gaussian Splatting in 200 seconds. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11146–11155, 2025. 2, 3, 4, 9
- [7] Jaeyoung Chung, Jeongtaek Oh, and Kyoung Mu Lee. Depth-regularized optimization for 3D Gaussian Splatting in few-shot images. In *CVPR Workshop on 3D Vision and Modeling*, 2024. 3
- [8] Xiaobin Deng et al. Gradient-driven natural selection for compact 3D Gaussian Splatting, 2025. 3
- [9] Ankit Dhiman et al. Turbo-GS: Accelerating 3D Gaussian fitting for high-quality radiance fields, 2024. 3
- [10] Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, De-jia Xu, and Zhangyang Wang. LightGaussian: Unbounded 3D Gaussian compression with 15x reduction and 200+ fps. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. 3
- [11] Guangchi Fang and Bing Wang. Mini-Splatting: Representing scenes with a constrained number of gaussians. In *European Conference on Computer Vision (ECCV)*, 2024. 3
- [12] Guangchi Fang and Bing Wang. Mini-Splatting2: Building 360 scenes within minutes via aggressive Gaussian densification, 2024. 3
- [13] Ben Fei et al. 3D Gaussian Splatting as a new era: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 31:4429–4451, 2025. 2
- [14] Florian Hahlbohm, Linus Franke, Martin Eisemann, and Marcus Magnor. Faster-GS: Analyzing and improving Gaussian Splatting optimization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. 9
- [15] Alex Hanson, Allen Tu, Geng Lin, Vasu Singla, Matthias Zwicker, and Tom Goldstein. Speedy-Splat: Fast 3D Gaus-

- sian Splatting with sparse pixels and sparse primitives, 2024. 3
- [16] Alex Hanson, Allen Koressel, and Subhransu Maji. PUP 3D-GS: Principled uncertainty pruning for 3D Gaussian Splatting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 3
 - [17] Lukas Höllein et al. 3DGS-LM: Faster Gaussian-Splatting optimization with Levenberg-Marquardt. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025. 4
 - [18] Binbin Huang, Zehao Yu, Anpei Chen, Andreas Geiger, and Shenghua Gao. 2D Gaussian Splatting for geometrically accurate radiance fields. *ACM Transactions on Graphics (SIGGRAPH)*, 43(4), 2024. 3
 - [19] Binxiao Huang, Zhengwu Liu, and Ngai Wong. Decomposing densification in Gaussian Splatting for faster 3d scene reconstruction, 2025. 3
 - [20] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3D Gaussian Splatting for real-time radiance field rendering. *ACM Transactions on Graphics (SIGGRAPH)*, 42(4):1–14, 2023. 2, 3, 4, 5, 6
 - [21] Shakiba Kheradmand, Daniel Rebain, Gopal Sharma, Weiwei Sun, Jeff Tseng, Hossam Isack, Abhishek Kar, Andrea Tagliasacchi, and Kwang Moo Yi. 3D Gaussian Splatting as Markov Chain Monte Carlo. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. 3
 - [22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. 4
 - [23] Dmytro Kotovenko, Olga Grebenkova, and Björn Ommer. EDGS: Eliminating densification for efficient convergence of 3DGS. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. 3
 - [24] Jiahe Li, Jiawei Zhang, Xiao Bai, Jin Zheng, Xin Ning, Jun Zhou, and Lin Gu. DNGaussian: Optimizing sparse-view 3D Gaussian radiance fields with global-local depth normalization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 3
 - [25] Jingxing Li, Yongjae Lee, and Deliang Fan. SkipGS: Post-densification backward skipping for efficient 3DGS training, 2026. 3
 - [26] Kaimin Liao, Hua Wang, Zhi Chen, Luchao Wang, and Yaohua Tang. LiteGS: A high-performance framework to train 3dgs in subminutes via system and algorithm codesign. *arXiv preprint arXiv:2503.01199*, 2025. 3
 - [27] Jiaqi Liu and Zhizhong Han. Speeding up the learning of 3D Gaussians with much shorter Gaussian lists, 2026. 3
 - [28] Tao Lu, Mulin Yu, Linning Xu, Yuanbo Xiangli, Limin Wang, Dahua Lin, and Bo Dai. Scaffold-GS: Structured 3D Gaussians for view-adaptive rendering. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. Highlight. 3
 - [29] Yanzhe Lyu et al. ResGS: Residual densification of 3D Gaussian for efficient detail recovery. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025. 3
 - [30] Saswat Subhajyoti Mallick, Rahul Goel, Bernhard Kerbl, Markus Steinberger, Francisco Vicente Carrasco, and Fernando De La Torre. Taming 3DGS: High-quality radiance fields with limited resources. In *SIGGRAPH Asia 2024 Conference Papers*, 2024. 2, 3, 5, 6, 9
 - [31] Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *European Conference on Computer Vision (ECCV)*, pages 405–421. Springer, 2020. 2
 - [32] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. pages 1–15, 2022. 2
 - [33] Thomas Neff, Pascal Stadlbauer, Mathias Parger, Andreas Kurz, Joerg H. Mueller, Chakravarty R. Alla Chaitanya, Anton Kaplanyan, and Markus Steinberger. AdaNeRF: Adaptive sampling for real-time rendering of neural radiance fields. In *European Conference on Computer Vision (ECCV)*, 2022. 3
 - [34] Xuran Pan, Zihang Lai, Shiji Song, and Gao Huang. ActiveNeRF: Learning where to see with uncertainty estimation. In *European Conference on Computer Vision (ECCV)*, 2022. 3
 - [35] Lukas Radl, Michael Steber, Mathias Parger, Bernhard Kerbl, and Markus Steinberger. StopThePop: Sorted Gaussian Splatting for view-consistent real-time rendering. *ACM Transactions on Graphics*, 43(4), 2024. 4
 - [36] Shiwei Ren, Tianci Wen, Yongchun Fang, and Biao Lu. FastGS: Training 3D Gaussian Splatting in 100 seconds. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2026. Highlight. 1, 2, 3, 4, 5, 8, 9
 - [37] Johannes L Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 4
 - [38] Johannes L Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *European Conference on Computer Vision (ECCV)*, 2016. 4
 - [39] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, et al. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH*, 2023. 3
 - [40] Fabio Tosi et al. NeRF: Neural radiance field in 3d vision: A comprehensive review. *arXiv preprint arXiv:2210.00379*, 2024. Updated post-Gaussian Splatting. 2
 - [41] Chengbo Wang, Guozheng Ma, Yifei Xue, and Yizhen Lao. Faster and better 3D splatting via group training. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025. 3
 - [42] Penguin Wang, Yueji Wang, Dongjin Wang, Shijia Mo-han, Zhiwen Fan, Liang Wu, Ruijin Cai, Yung-Yu Yeh, Zhangyang Wang, Qixing Liu, and Rakesh Ranjan. Steepest descent density control for compact 3D Gaussian Splatting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025. 3
 - [43] 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. 4, 5

- [44] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-NeRF: Point-based neural radiance fields. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. [2](#)
- [45] Vickie Ye, Matias Turkulainen, and the nerfstudio team. gsplat: An open-source library for Gaussian Splatting. *Journal of Machine Learning Research (JMLR)*, 26:1–15, 2025. [3](#), [4](#)
- [46] Zongxin Ye, Wenyu Li, Sidun Liu, Peng Qiao, and Yong Dou. AbsGS: Recovering fine details for 3D Gaussian Splatting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. [3](#)
- [47] Zehao Yu, Anpei Chen, Binbin Huang, Torsten Sattler, and Andreas Geiger. Mip-Splatting: Alias-free 3D Gaussian Splatting. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. Best Student Paper. [3](#)
- [48] Zehao Yu, Torsten Sattler, and Andreas Geiger. Gaussian Opacity Fields: Efficient adaptive surface reconstruction in unbounded scenes. *ACM Transactions on Graphics*, 43(6), 2024. [3](#)
- [49] Jiahui Zhang, Fangneng Zhan, Muyu Xu, Shijian Lu, and Eric Xing. FreGS: 3D Gaussian Splatting with progressive frequency regularization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. [3](#)
- [50] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. [4](#), [5](#)
- [51] Zheng Zhang, Wenbo Hu, Yixing Lao, Tong He, and Hengshuang Zhao. Pixel-GS: Density control with pixel-aware gradient for 3D Gaussian Splatting. In *European Conference on Computer Vision (ECCV)*, 2024. [3](#)
- [52] Ziyu Zhang, Tianle Liu, Diantao Tu, and Shuhan Shen. Fast converging 3D Gaussian Splatting for 1-minute reconstruction, 2025. [3](#)
- [53] Zongze Zhang et al. LP-3DGS: Learning to prune 3D Gaussian Splatting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. [3](#)
- [54] Zehao Zhu, Zhiwen Fan, Yifan Jiang, and Zhangyang Wang. FSGS: Real-time few-shot view synthesis using Gaussian Splatting. In *European Conference on Computer Vision (ECCV)*, 2024. [3](#)
- [55] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. EWA splatting. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):223–238, 2002. [2](#)