EVERY GAUSSIAN COUNTS: RECAP-GS FOR RESOURCE-CONSTRAINED 3D GAUSSIAN SPLATTING

Anonymous authors
Paper under double-blind review

ABSTRACT

3D Gaussian Splatting achieves high-quality novel-view synthesis, but its aggressive densification can make training and deployment costly under tight memory and representation-size budgets. We formulate budget-constrained 3D Gaussian Splatting as a capacity-allocation problem and introduce ReCap-GS (Recovering Capacity for Gaussian Splatting), a fixed-budget framework centered on stale-capacity recovery. Through a controlled opacity perturbation and recovery phase, ReCap-GS identifies primitives that fail to regain responsibility and reclaims their slots for subsequent demand-driven refinement. The method combines early multi-view residual-consensus growth, budget-preserving recycling, and transmittance-aware opacity redistribution while enforcing a fixed upper bound on the Gaussian population throughout training. After topology optimization has ended, we additionally adapt the Multi-view Update strategy introduced in Improved-GS. We evaluate ReCap-GS on the seven public Mip-NeRF360 scenes against a broad suite of budget-aware, pruning-based, relocation-based, and adaptive-densification baselines. Under strict budgets of 50k, 100k, and 200k Gaussians, ReCap-GS improves PSNR over Improved-GS by 0.43, 0.36, and 0.24 dB, respectively, while matching or improving SSIM and LPIPS. Compared with Improved-GS, peak VRAM is reduced by approximately 76%, 73%, and 65% at these budgets, respectively. ReCap-GS remains competitive as capacity increases, suggesting that under tight budgets, recovering stale capacity is an effective complement to demand-driven representation growth.

1 INTRODUCTION

3D Gaussian Splatting (3DGS) has become a widely used representation for novel-view synthesis, combining high rendering quality with real-time rasterization (Kerbl et al., 2023). Its fast renderer, however, does not imply an efficient training process or a compact representation. Starting from a sparse Structure-from-Motion initialization, standard 3DGS repeatedly densifies the scene and can grow to millions of primitives. This increases memory consumption, model size, and rendering cost, and makes the resources required by a reconstruction difficult to control.

Recent work has addressed this problem through budget-aware densification, pruning, relocation, and redistribution of Gaussian primitives (Fang & Wang, 2024; Mallick et al., 2024; Kheradmand et al., 2024; Deng et al., 2026). These approaches improve compactness by controlling where capacity is added, removed, or relocated. Under a strict population budget, however, the problem is more general: once all available slots are occupied, improving one region requires deciding which capacity elsewhere has become dispensable. A primitive that is weak at a given instant is not necessarily obsolete; conversely, a primitive that was previously important may have become stale if it can no longer regain responsibility after the representation is perturbed.

We therefore view resource-constrained 3DGS as a capacity-allocation problem. Rather than asking only where the representation should grow, we ask which existing primitives have become stale. We introduce ReCap-GS (Recovering Capacity for Gaussian Splatting), a fixed-budget framework whose central mechanism is stale-capacity recovery. ReCap-GS probes the current allocation by applying a controlled opacity perturbation and allowing the representation to recover. Primitives that previously carried substantial responsibility but fail to regain it after recovery are identified through a drop in their relative opacity rank, which we use as a proxy for loss of responsibility.

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.

Their slots are reclaimed and returned to the available capacity pool, where subsequent absolute-gradient refinement reuses them according to local splitting demand. Unlike selection based solely on current opacity, the stale score captures how a primitive's relative role changes during recovery.

ReCap-GS progressively grows the representation toward a user-defined budget and then reuses capacity in place, so that the Gaussian population never exceeds the prescribed upper bound. Structural updates redistribute opacity using a rule designed to limit abrupt changes in accumulated transmittance during replacement. Once topology optimization is complete, we additionally adapt the Multi-view Update strategy introduced in Improved-GS (Deng et al., 2026) as a late-stage refinement, averaging gradients across multiple views while keeping the total number of processed views fixed. This separation is deliberate: stale-capacity recovery determines where occupied capacity is recovered, whereas early residual consensus and subsequent absolute-gradient refinement determine where available capacity is allocated. Late multi-view refinement then optimizes the resulting frozen topology.

We evaluate ReCap-GS on the seven public Mip-NeRF360 (Barron et al., 2022) scenes over Gaussian budgets ranging from 50k to 2M, against a broad set of compact, pruning-based, relocation-based, and adaptive-densification baselines. We distinguish two evaluation protocols that are often conflated: a native/final-budget setting, where a method may transiently exceed its final target size, and a strict setting enforcing $N(t) \leq B$ throughout training, including initialization. Under strict budgets of 50k, 100k, and 200k Gaussians, our final configuration improves PSNR over Improved-GS by 0.43, 0.36, and 0.24 dB, respectively, while matching or improving its SSIM and LPIPS. Compared with Improved-GS, peak VRAM is reduced by approximately 76%, 73%, and 65% at these budgets. The advantage is strongest in the constrained regime, while reconstruction quality remains competitive as capacity increases.

Controlled ablations support the proposed mechanism rather than generic recycling alone. The recovery probe is beneficial by itself, but history-dependent stale selection provides additional gains over probe-only training, random reclamation, and selection based solely on post-recovery opacity. Together with the resource-aware benchmark, these results establish ReCap-GS as a framework for treating a fixed Gaussian population as reusable capacity rather than as a one-way densification process.

2 Related Work

3D Gaussian Splatting and densification. 3D Gaussian Splatting (3DGS) represents a scene with anisotropic Gaussian primitives optimized directly from posed images and rendered through differentiable splatting (Kerbl et al., 2023). Its adaptive density control progressively clones, splits, and removes primitives, enabling high-fidelity reconstruction but often producing large and scene-dependent Gaussian populations. Several works have therefore revisited the densification process itself. Improved-GS (Deng et al., 2026) studies when and how densification should occur and introduces additional refinement mechanisms, including Multi-view Update, to improve reconstruction fidelity. ReCap-GS instead targets capacity that has already been allocated and asks when it should be recovered for reuse.

Compact and budget-constrained Gaussian representations. A growing body of work aims to reduce the number of Gaussians or explicitly control representation size. Mini-Splatting (Fang & Wang, 2024) reorganizes the spatial distribution of primitives through specialized densification and simplification operations under a constrained number of Gaussians. Taming-3DGS (Mallick et al., 2024) uses guided constructive densification to grow toward a prescribed target budget while improving training efficiency. 3DGS-MCMC (Kheradmand et al., 2024) interprets Gaussian relocation and densification through a Markov-chain Monte Carlo formulation, enabling continual redistribution of the representation. More recent methods further emphasize spatial redistribution and pruning: GS^2 (Yang et al., 2026) combines adaptive densification, progressive pruning, and graph-guided spatial relocation, while Prune Wisely (Wang et al., 2026) introduces reconstruction-aware adaptive pruning together with 3D Difference-of-Gaussians primitives. SalientGS (Xiong et al., 2026) uses multi-view residual signals to guide Gaussian birth and relocation toward underfit regions. These approaches demonstrate that compactness depends not only on how many primitives are retained, but also on how capacity is distributed spatially. Orthogonal work targets storage rather than primitive

allocation through attribute compression, quantization, and compact parameterizations (Lee et al., 2024; Fan et al., 2024; Chen et al., 2024). Our focus is instead the allocation of a strictly bounded primitive population during training.

Capacity allocation under a finite budget. SPARE-GS (Chen et al., 2026) is particularly related to our capacity-allocation perspective: it regulates structural evolution through demand-driven regional quotas under a global resource budget. Whereas SPARE-GS controls capacity at the regional level, ReCap-GS uses a controlled recovery trajectory to identify stale capacity at the primitive level. This history-dependent victim criterion determines which occupied slots to reclaim; capacity allocation itself follows early residual consensus and subsequent absolute-gradient splitting demand. ReCap-GS is therefore complementary to pruning-, relocation-, and quota-based approaches: its focus is not merely on reducing the representation, but on detecting when previously allocated capacity should be recovered and reused.

3 METHOD

3.1 PROBLEM FORMULATION

Let $\mathcal{G}_t = \{G_i\}_{i=1}^{N_t}$ denote the Gaussian representation at training iteration t, where each primitive contains a center, covariance, opacity, and view-dependent color parameters. Given calibrated training views $\{(I_v, \Pi_v)\}$, standard 3DGS minimizes the photometric objective

$$\mathcal{L}_v(\mathcal{G}) = (1-\lambda)\|\hat{I}_v - I_v\|_1 + \lambda\big(1 - \mathrm{SSIM}(\hat{I}_v, I_v)\big), \qquad \hat{I}_v = \mathcal{R}(\mathcal{G}; \Pi_v). \tag{1}$$

We set $\lambda = 0.2$ for all ReCap-GS experiments.

Our setting additionally constrains the entire optimization trajectory by a user-specified primitive budget B:

$$N_t \le B \qquad \text{for every training iteration } t. \tag{2}$$

The constraint applies from the Structure-from-Motion initialization onward, not only to the final checkpoint. Once $N_t = B$, adding capacity in one region therefore requires recovering it elsewhere. ReCap-GS treats this as a capacity-allocation problem with two complementary questions: where is additional capacity demanded, and which occupied slots have become stale? Figure 1 summarizes the training schedule and the separation between capacity recovery and capacity allocation.

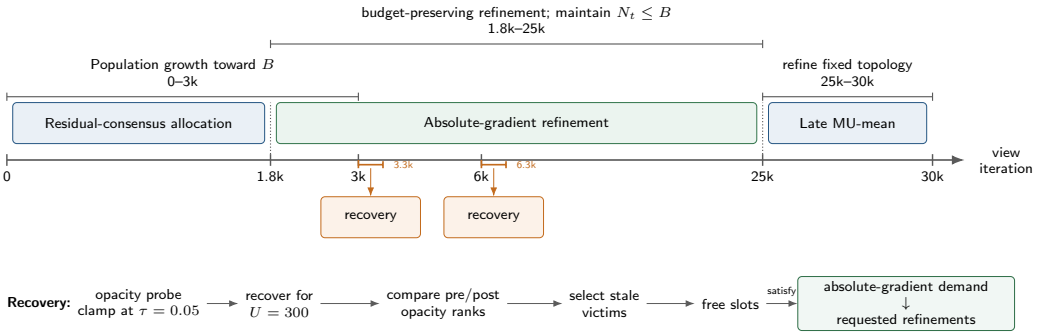

Figure 1: Overview of ReCap-GS. Residual-consensus allocation guides early growth toward persistent multi-view residuals until iteration 1.8k, when allocation switches to absolute-gradient refinement. Progressive population growth toward the budget continues through the early refinement phase. The Gaussian population remains bounded by B throughout training. Two recovery cycles begin at 3k and 6k: each applies an opacity probe, allows $U = 300$ view updates for recovery, and reclaims slots according to the loss of pre-to-post opacity rank. Topology refinement ends at 25k, followed by late MU-mean optimization of the fixed topology for the final 5k processed views. Recovery history determines which capacity is reclaimed, while absolute-gradient demand determines where the freed slots are used.

3.2 BUDGET-PRESERVING CAPACITY ALLOCATION

We first cap the SfM initialization at B and progressively grow the representation toward the budget. During the early growth phase, allocation is driven by multi-view residual consensus. High positive-residual image samples propose spatial targets, which are grouped using a voxelized multi-view consistency test. Only targets supported by multiple views and satisfying a color-consistency criterion are retained. Full construction of the residual-consensus targets is provided in the appendix.

At iteration 1.8k, the structural policy transitions to gradient-based refinement. We adopt the absolute screen-space gradient accumulation of Improved-GS (Deng et al., 2026) to identify primitives with strong splitting demand. This refinement is evaluated every 100 iterations until topology optimization ends at iteration 25k.

When free slots are available, structural demand can be satisfied directly. When insufficient capacity remains, requested splits are handled through budget-preserving slot exchanges. To select ordinary exchange victims, we rank non-parent primitives using a support heuristic that combines opacity, accumulated view visibility, and maximum screen-space radius, and reclaim primitives with the lowest support. The exact support score is provided in the appendix.

Victim slots are reused in place for replacement children, so the hard population constraint is respected throughout the structural update. Stale-capacity recovery augments this backbone with a history-dependent criterion for deciding which occupied slots should be released; the active structural demand then determines how available capacity is reused.

3.3 STALE-CAPACITY RECOVERY

Instantaneous opacity alone is an ambiguous measure of usefulness: a primitive may currently be weak because it supports a difficult or rarely observed region. We instead test whether a primitive can recover its relative role after a controlled perturbation. At recovery cycle c, beginning at iteration t_c, we store a persistent identity and the pre-perturbation opacity α_i^{pre} of every current primitive. We then clamp opacity to a maximum value τ,

$$\widetilde{\alpha}_i = \min(\alpha_i^{\mathrm{pre}}, \tau)\,, \tag{3}$$

and continue standard optimization for U view updates. The perturbation reduces established opacity advantages without changing geometry or appearance parameters, allowing image evidence to redistribute responsibility. We denote the resulting recovered opacity by α_i^{rec}.

Let $\mathcal{E}_c$ contain the primitives whose identities remain valid throughout the recovery window. For a value vector a over this set, let $\mathrm{rank}_i(a) \in [0,1]$ denote the normalized ascending rank of primitive i. We define the stale score as

$$s_i = \mathrm{rank}_i(\alpha^{\mathrm{pre}}) - \mathrm{rank}_i(\alpha^{\mathrm{rec}})\,. \tag{4}$$

A large positive score identifies a primitive that ranked highly before the probe but failed to recover the same relative responsibility afterward. Because the score depends only on ordering, it is invariant to strictly increasing transformations of the opacity values. It therefore distinguishes a historical loss of relative responsibility from merely having low current opacity. We reclaim the K_c primitives with the largest stale scores,

$$K_c = \max(1, \lfloor \rho|\mathcal{E}_c| \rfloor)\,, \qquad \mathcal{V}_c = \mathrm{TopK}_{K_c}\left(\{i \in \mathcal{E}_c\};\, s_i\right), \tag{5}$$

where ρ is the recovery fraction. The released slots are returned to the available capacity pool and can be consumed by subsequent structural updates. At the recovery cycles used in our final schedule, these updates are driven by absolute-gradient splitting demand. Thus, recovery history determines where capacity is taken from, while the active allocation policy determines where that capacity is subsequently used.

3.4 TRANSMITTANCE-AWARE OPACITY REDISTRIBUTION

When a victim slot is recycled, the original parent is retained and a replacement child is introduced. Copying the parent opacity α to both would increase their initial attenuation. We instead assign

$$\alpha' = 1 - \sqrt{1-\alpha} \tag{6}$$

to both primitives. This preserves the original transmittance in the limiting case of coincident, unit-response contributions and otherwise provides a transmittance-aware initialization rather than an exact conservation law. Full details are given in Appendix A.3.

3.5 Late multi-view refinement

After topology optimization ends at T_{top}, we freeze the primitive population and adapt the Multi-view Update strategy of Improved-GS (Deng et al., 2026). For a group $\mathcal{B}_k$ of m_k successive training views, we accumulate the mean gradient

$$g_k = \frac{1}{m_k} \sum_{v \in \mathcal{B}_k} \nabla_\Theta \mathcal{L}_v(\mathcal{G}; \Theta), \tag{7}$$

and perform one Adam update. Averaging, rather than summing, prevents the gradient magnitude from scaling directly with group size. We retain the same camera sequence and total number of processed views as ordinary continuation; only the frequency and multi-view support of optimizer updates change. This stage refines the parameters of a fixed topology and is conceptually separate from stale-capacity recovery.

4 Experiments

4.1 Experimental setup

Dataset and metrics. We evaluate on the seven public Mip-NeRF360 scenes (Barron et al., 2022) at Gaussian budgets of 50k, 100k, 200k, 300k, 500k, 1M, and 2M, using the standard train/test split and resolution factors. Reconstruction quality is measured using PSNR, SSIM, and LPIPS-VGG. To ensure consistent comparison across implementations, all metrics are recomputed from rendered test-view PNG images using a common evaluation pipeline. Unless stated otherwise, results are first averaged over the seven scenes for each training seed and reported as mean $\pm$ standard deviation across seeds 1 and 2.

Budget protocols and baselines. We distinguish a native/final-budget protocol, which permits temporary population growth beyond the final target size, from a strict protocol requiring $N_t \leq B$ throughout the complete trajectory, including initialization. When required, strict-adapted methods use a deterministic initialization cap; a run is included in the strict comparison only when the trajectory constraint is guaranteed or fully instrumented. We compare against Vanilla capped 3DGS (Kerbl et al., 2023), Mini-Splatting (Fang & Wang, 2024), Taming-3DGS (Mallick et al., 2024), 3DGS-MCMC (Kheradmand et al., 2024), Improved-GS (Deng et al., 2026), GS^2 (Yang et al., 2026), Prune Wisely (Wang et al., 2026), SalientGS (Xiong et al., 2026), and SPARE-GS (Chen et al., 2026). For the controlled budget sweeps, we integrate the SPARE-GS regional structural-regulation principle into the Taming-3DGS backbone and report this configuration as Taming+SPARE (our adaptation). Separately, Fig. 3 includes the official 3DGS+SPARE implementation at its evaluated natural representation size. Complete baseline-adaptation details are provided in Appendix B.3.

Implementation and resource measurement. ReCap-GS is trained for 30k processed views. Early residual-consensus allocation is followed by absolute-gradient refinement until topology is frozen at iteration 25k. We use two stale-recovery cycles beginning at 3k and 6k, with $U = 300$, $\tau = 0.05$, and $\rho = 0.2$; the final 5k views use late MU-mean refinement. The complete schedule and structural hyperparameters are provided in Appendix A.4.

ReCap-GS keeps training images in CPU memory and transfers the active view to the GPU on a dedicated CUDA stream. Peak VRAM is measured as the maximum `nvidia-smi` memory usage over the complete method process tree, and wall time covers the same end-to-end pipeline. Resource measurements therefore reflect the complete implementations, including their data-staging strategies. All resource comparisons use runs from the same NVIDIA RTX A6000 cluster campaign.

4.2 Comparison with prior work

Strict-budget comparison. Table 1 compares methods whose complete training trajectory satisfies the requested population budget. ReCap-GS achieves the highest PSNR at 50k, 100k, and 200k,

Table 1: Strict-budget comparison on the seven Mip-NeRF360 scenes. Values are mean ± sample standard deviation across training seeds 1 and 2 after first averaging each seed over the seven scenes. All reported trajectories satisfy $N_t \leq B$ throughout training. Whenever the native SfM initialization exceeds the requested budget, strict-adapted baselines use the deterministic initialization cap described in Appendix B.2; complete baseline adaptations are listed in Appendix B.3.

Method	PSNR ↑	SSIM ↑	LPIPS ↓
50k Gaussians			
Vanilla	21.807±.014	.7034±.0005	.3970±.0000
Taming	25.573±.008	.7427±.0000	.3676±.0002
Taming+SPARE (our adaptation)	25.571±.011	.7426±.0003	.3675±.0003
SalientGS	25.810±.009	.7628±.0001	.3430±.0002
3DGS-MCMC	26.085±.022	**.7694±.0001**	**.3385±.0000**
Improved-GS	26.200±.005	.7566±.0001	.3496±.0001
ReCap-GS (ours)	**26.625±.006**	.7664±.0004	.3392±.0004
100k Gaussians			
Vanilla	23.366±.048	.7550±.0003	.3419±.0001
Taming	26.718±.015	.7856±.0001	.3140±.0000
Taming+SPARE (our adaptation)	26.673±.056	.7846±.0003	.3148±.0003
SalientGS	26.825±.012	.8015±.0004	.2906±.0003
3DGS-MCMC	27.229±.008	**.8117±.0001**	.2845±.0002
Improved-GS	27.485±.001	.8046±.0000	.2899±.0000
ReCap-GS (ours)	**27.846±.004**	.8101±.0000	**.2836±.0002**
200k Gaussians			
Vanilla	25.105±.035	.7971±.0011	.2908±.0002
Taming	27.893±.032	.8215±.0001	.2655±.0000
Taming+SPARE (our adaptation)	27.869±.012	.8208±.0003	.2663±.0001
SalientGS	27.720±.008	.8310±.0001	.2443±.0000
3DGS-MCMC	28.212±.030	**.8445±.0001**	.2380±.0001
Improved-GS	28.567±.003	.8417±.0002	**.2374±.0004**
ReCap-GS (ours)	**28.806±.005**	.8436±.0000	**.2374±.0000**

improving over Improved-GS by 0.43, 0.36, and 0.24 dB, respectively. The comparison does not imply uniform metric dominance: 3DGS-MCMC achieves the highest SSIM throughout this range and slightly lower LPIPS at 50k, while ReCap-GS obtains the lowest LPIPS at 100k and matches Improved-GS at 200k. The effect of initialization capping on individual baselines is reported in Appendix C.2.

Across-budget behavior. ReCap-GS achieves the highest PSNR in the strict comparison at 50k, 100k, and 200k Gaussians. Its advantage over Improved-GS narrows to 0.135 dB at 300k, and the absolute PSNR gap remains below 0.06 dB at larger budgets. Improved-GS achieves higher SSIM at all three larger budgets and lower LPIPS at 500k and 1M, whereas ReCap-GS achieves lower LPIPS at 2M. These results position ReCap-GS primarily as a constrained-capacity method. The complete budget sweep, including native/final-budget configurations, is reported in Appendix C.

Qualitative comparison. Figure 2 compares representative held-out views at a strict 100k-Gaussian budget. Additional fixed-view comparisons over all seven scenes at 50k–200k are provided in Appendix F.

4.3 Quality–resource trade-offs

At 50k, 100k, and 200k Gaussians, ReCap-GS reduces peak VRAM relative to Improved-GS by approximately 76%, 73%, and 65%, respectively, while achieving higher PSNR. The memory advantage decreases with capacity, reaching 12% at 2M. Wall time follows a different trend: ReCap-GS is slightly faster at 50k and 100k, but not from 200k onward. We therefore interpret the primary

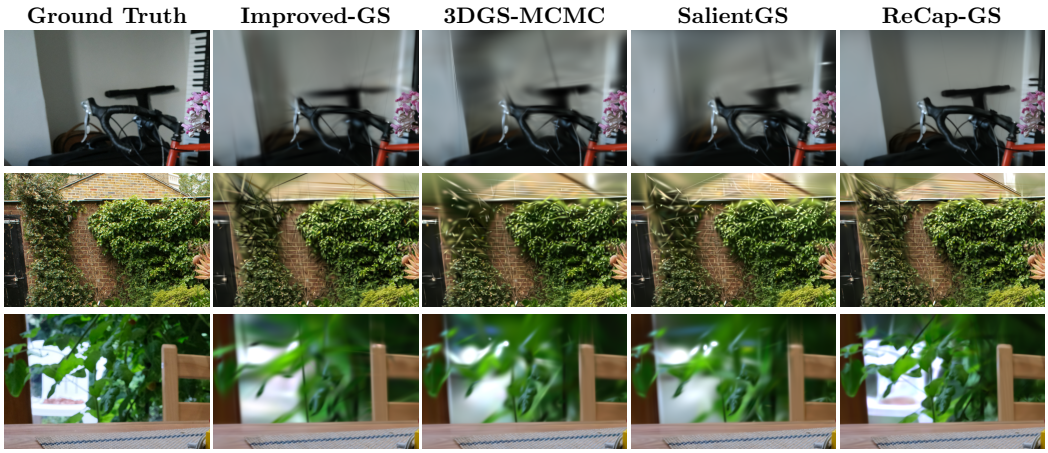

Figure 2: Qualitative comparison on held-out Mip-NeRF360 views under a strict 100k-Gaussian budget. All methods use their certified strict configuration from Table 1, with identical views and crop coordinates. The examples illustrate localized reconstruction differences in the capacity-constrained regime; systematic comparisons across all seven scenes are provided in Appendix F.

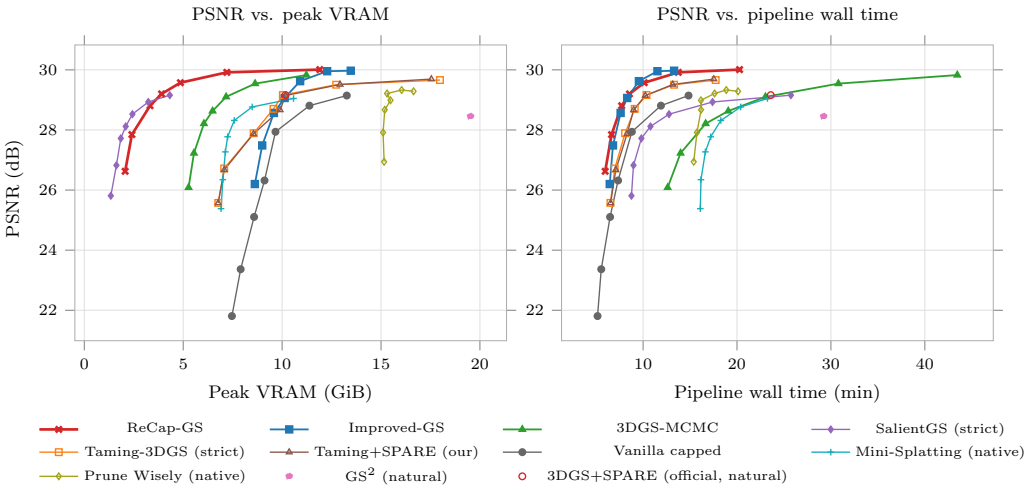

Figure 3: Quality–resource trade-offs across measured operating points. **Left:** PSNR versus peak VRAM. **Right:** PSNR versus end-to-end pipeline wall time. For methods evaluated over a budget sweep, connected points correspond to increasing Gaussian budgets from 50k to 2M when available; lines are shown only as visual guides and should not be interpreted as Pareto frontiers. For strict-adapted methods, the 50k–200k points use initialization-capped runs, while at larger budgets the native initialization already satisfies the strict constraint. Mini-Splatting and Prune Wisely are shown under their native protocols. GS^2 and the official 3DGS+SPARE configuration are shown only at their natural representation sizes and therefore appear as isolated operating points. Resource measurements follow Sec. 4.1.

resource benefit as an improved quality–memory trade-off under constrained capacity rather than as a universal runtime advantage. Exact measurements are reported in Appendix D.

Table 2: Ablation at a 300k-Gaussian budget. Values are mean ± standard deviation across training seeds 1 and 2 after averaging over all seven scenes. Higher PSNR/SSIM and lower LPIPS are better.

Variant	PSNR ↑	SSIM ↑	LPIPS ↓
Backbone	28.994±.022	.8528±.0001	.2206±.0004
Backbone + late MU-mean	29.164±.006	.8543±.0001	.2194±.0001
Backbone + stale	29.063±.009	.8575±.0000	.2153±.0000
Backbone + stale (no opacity redistribution)	29.008±.010	.8563±.0001	.2173±.0001
ReCap-GS (stale + late MU-mean)	**29.200±.007**	**.8588±.0001**	**.2143±.0003**

Table 3: Recovery-policy diagnostic at 300k Gaussians, averaged over *bicycle*, *stump*, and *kitchen* and training seeds 1 and 2. All variants use the same budgeted backbone; they differ in whether and how the recovery trajectory is used for victim selection.

Policy	PSNR ↑	SSIM ↑	LPIPS ↓
Backbone	27.558	.7917	.2529
Probe only	27.628	.7932	.2515
Random recovery	27.603	.7923	.2538
Post-recovery opacity	27.679	.7974	.2493
Stale recovery	**27.713**	**.8005**	**.2440**

4.4 Ablation studies

We ablate ReCap-GS at a fixed budget of 300k Gaussians. Unless stated otherwise, results use all seven Mip-NeRF360 scenes and training seeds 1 and 2, with the same budgeted backbone and training schedule.

Component ablations. Table 2 separates stale-capacity recovery from late MU-mean. Stale recovery improves the backbone by 0.070 dB PSNR and 0.0047 SSIM while reducing LPIPS by 0.0053, and late MU-mean is also beneficial independently. Their combination gives the strongest overall result. Removing transmittance-aware opacity redistribution degrades all three quality metrics, indicating that stable structural replacement contributes in addition to the stale victim-selection policy.

Recovery signal. Table 3 isolates the information used by stale recovery. The perturbation itself is beneficial, but stale rank change provides a further $+0.085$ dB PSNR over probe-only and outperforms both random reclamation and selection using post-recovery opacity. This indicates that the recovery trajectory carries useful information beyond either the perturbation alone or the primitive's instantaneous post-recovery state. Matched recovery-count statistics and the MU-mean versus MU-sum control are reported in Appendix E.

5 Conclusion

We introduced ReCap-GS, a resource-constrained 3D Gaussian Splatting framework that treats a fixed Gaussian population as reusable representation capacity. Its central mechanism, stale-capacity recovery, probes the current allocation through a controlled opacity perturbation and identifies primitives that fail to recover their relative responsibility. Their slots are then reclaimed and returned to the capacity pool and reassigned according to the active structural demand, while the Gaussian population remains bounded throughout training.

On the seven Mip-NeRF360 scenes, ReCap-GS provides its strongest gains in the capacity-constrained regime. Under strict budgets of 50k, 100k, and 200k Gaussians, it improves PSNR over Improved-GS by 0.43, 0.36, and 0.24 dB, respectively, while reducing peak VRAM by approximately 76%, 73%, and 65%. Controlled ablations further show that the recovery probe is beneficial on its own, while history-dependent stale selection provides additional gains over random reclamation and selection based only on post-recovery opacity.

The advantage diminishes as capacity increases, and ReCap-GS does not provide a universal runtime improvement at large budgets. Nevertheless, the results show that under tight resource constraints, deciding which existing capacity should be recovered is an important complement to deciding where available capacity should be allocated. This capacity-reuse perspective offers a practical direction for controlling representation growth in 3D Gaussian Splatting without relying on unconstrained densification.

AI USE STATEMENT

Generative AI tools were used to assist with language editing and paper organization. The authors reviewed and verified all AI-assisted text against the implementation and experiment logs. Generative AI was not used to generate code, analysis, experimental measurements or determine reported results. The authors take responsibility for the final content of the work, including text, claims, and artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

All main experiments use fixed public Mip-NeRF360 splits, explicit Gaussian budgets, and a common matched-PNG evaluation pipeline. The paper distinguishes native/final-budget and strict-capacity protocols and reports the resource measurement definitions used for peak VRAM and wall-clock time. The supplementary material provides the complete budget-matched benchmark, strict-budget certification details, baseline adaptations, the complete ReCap-GS training schedule, additional mechanistic controls, and the exact resource operating points underlying the quality–resource comparison.

The current research implementation is not included with the initial submission, as it is undergoing code cleanup, additional unit testing, and reproducibility checks. An anonymized implementation can be made available to reviewers during the discussion period if needed.

REFERENCES

Jonathan T. Barron, Ben Mildenhall, Dor Verbin, Pratul P. Srinivasan, and Peter Hedman. Mip-nerf 360: Unbounded anti-aliased neural radiance fields. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5470–5479, 2022. doi: 10.1109/CVPR52688.2022.00539.

Yihang Chen, Qianyi Wu, Weiyao Lin, Mehrtash Harandi, and Jianfei Cai. HAC: Hash-grid assisted context for 3d gaussian splatting compression. In *European Conference on Computer Vision (ECCV)*, pp. 422–438, 2024. doi: 10.1007/978-3-031-72667-5_24.

Zhang Chen, Shuai Wan, Fuzheng Yang, Jiazhi Xia, Weiyao Lin, and Junhui Hou. SPARE-GS: Structural parsimony and resource efficiency for 3d gaussian splatting, 2026.

Xiaobin Deng, Changyu Diao, Min Li, Ruohan Yu, and Duanqing Xu. Improving densification in 3d gaussian splatting for high-fidelity rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Findings*, pp. 223–232, 2026.

Zhiwen Fan, Kevin Wang, Kairun Wen, Zehao Zhu, Dejia Xu, and Zhangyang Wang. Lightgaussian: Unbounded 3d gaussian compression with 15x reduction and 200+ fps. In *Advances in Neural Information Processing Systems*, volume 37, pp. 140138–140158, 2024.

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.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42(4):1–14, 2023. doi: 10.1145/3592433.

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

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

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

Haoran Wang, Guoxi Huang, Fan Zhang, David Bull, and Nantheera Anantrasirichai. Prune wisely, reconstruct sharply: Compact 3d gaussian splatting via adaptive pruning and difference-of-gaussian primitives. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11716–11725, 2026.

Tianyu Xiong, Rui Li, Suning Ge, and Jiaqi Yang. Salientgs: Unified sfm-to-3dgs with importance-guided mcmc gaussian allocation, 2026.

Xianben Yang, Tao Wang, Yuxuan Li, Yi Jin, and Haibin Ling. GS^2: Graph-based spatial distribution optimization for compact 3d gaussian splatting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 33259–33268, 2026.

A Additional Implementation Details

This appendix provides the implementation details omitted from the main paper, including the ordinary recycling criterion, residual-consensus allocation, the complete training schedule, and pseudocode for ReCap-GS. Unless stated otherwise, all settings are shared across scenes and Gaussian budgets.

A.1 Ordinary Recycling

During absolute-gradient refinement, parent candidates are identified using the absolute screen-space gradient accumulation adopted from Improved-GS (Deng et al., 2026). When splitting demand exceeds the number of free slots, non-parent primitives are ranked using

$$u_i = \alpha_i \sqrt{1 + v_i} \left(1 + \frac{q_i}{q_{\mathrm{med}}} \right), \tag{8}$$

where α_i is opacity, v_i accumulated visibility, and q_i the maximum observed screen-space radius. q_{med} is the median of strictly positive radii over active primitives and is clamped from below by one. If no positive radius is available, q_{med} is set to one.

If D_t denotes splitting demand not satisfied by free slots, the number of recycled slots is

$$K_t = \min(D_t,\, \max(1, \lfloor 0.01B \rfloor),\, |\mathcal{C}_t|)\,, \tag{9}$$

where $\mathcal{C}_t$ is the eligible non-parent victim pool. The K_t lowest-support victim slots are reused in place for replacement children, ensuring $N_t \leq B$ throughout the structural update.

A.2 Residual-Consensus Allocation

During early allocation, positive image-space residuals are defined as

$$r(p) = \frac{1}{3} \sum_{c \in \{R,G,B\}} \max\Big(I_c(p) - \hat{I}_c(p), 0 \Big)\,. \tag{10}$$

We retain local maxima within a 9×9 neighborhood satisfying $r(p) \geq 0.01$, accumulated opacity ≥ 0.05, and valid rendered depth, keeping at most 4096 proposals per sampled view.

Valid pixels are lifted to 3D using rendered accumulated opacity $A(p)$ and inverse depth $D_{\text{inv}}(p)$,

$$z(p) = \left(\frac{D_{\text{inv}}(p)}{\max(A(p), 10^{-4})} \right)^{-1}, \tag{11}$$

and transformed to world coordinates. Proposals are voxelized with cell size $h = 0.02E$, where E is the scene extent. A voxel is retained when it contains proposals from at least three distinct training views and satisfies $\sigma_{\text{RGB}} \leq 0.15$, where

$$\sigma_{\text{RGB}} = \sqrt{\frac{\text{Var}(R) + \text{Var}(G) + \text{Var}(B)}{3}}. \tag{12}$$

Target position and color are residual-weighted averages of the proposals within each accepted voxel. If accepted targets exceed the available capacity, voxels are ranked by mean residual. Proposals are collected every five processed views and residual-consensus insertion is evaluated every 600 views until the transition to absolute-gradient refinement at iteration 1.8k. The proposal buffer is cleared after each residual-consensus insertion event.

A.3 Transmittance-Aware Opacity Redistribution

When a victim slot is reused, the original parent is retained and a replacement child is introduced. Copying the parent opacity α to both would increase their initial attenuation. We instead assign

$$\alpha' = 1 - \sqrt{1 - \alpha} \tag{13}$$

to both the retained parent and recycled child.

For two coincident contributions with unit kernel response,

$$(1 - \alpha')^2 = 1 - \alpha, \tag{14}$$

so their composed transmittance matches that of the original primitive. In the implementation the transformation is applied to the stored opacity parameter, whereas projected per-pixel opacity is additionally modulated by the Gaussian kernel. Exact equality therefore holds only in the coincident, unit-response limit. The rule should be interpreted as a transmittance-aware initialization, not as conservation of opacity mass or exact per-ray attenuation after the child is displaced.

A.4 Complete Training Schedule

Table 4 summarizes the final ReCap-GS schedule. Iterations denote processed training views, and all structural operations respect $N_t \leq B$.

Recovery-window identities. Ordinary structural updates remain active during recovery windows. Primitives replaced or created during a window receive invalid recovery identities and are excluded from stale ranking; only identities that remain valid throughout the complete recovery trajectory belong to $\mathcal{E}_c$.

Execution order. At iterations 3k and 6k, the ordinary structural event is processed before recovery identities are assigned and opacities are perturbed. At 3.3k and 6.3k, the ordinary structural event similarly precedes stale reclamation. Slots released by stale recovery can therefore first be reused at the next structural event.

Late refinement. During late MU-mean, the learning-rate schedule follows processed-view count rather than optimizer-update count. The final 5k views correspond to 500 updates over five-view groups followed by 125 updates over twenty-view groups.

Table 4: Complete training schedule of the final ReCap-GS configuration. Iterations refer to processed training views. Structural operations always respect the hard population constraint $N_t \leq B$.

Stage / event	Iteration range	Frequency	Operation
SfM initialization	$t = 0$	Once	Initialize from the SfM point cloud and cap the initial population at B whenever the native initialization exceeds the prescribed budget.
Opacity cleanup	$t = 300$	Once	Remove primitives whose opacity is below 0.02.
Residual proposal collection	$t < 1.8$k	Every 5 views	Collect high positive-residual image-space proposals satisfying the local-maximum, opacity, and valid-depth criteria described in Sec. A.2.
Residual-consensus allocation	$t < 1.8$k	Every 600 views	Aggregate proposals in 3D, apply multi-view and color-consistency tests, and instantiate accepted targets while free capacity remains.
Absolute-gradient refinement	$1.8\text{k} \leq t < 25\text{k}$	Every 100 views	Use absolute screen-space gradient accumulation to identify parent primitives with splitting demand. When free capacity is insufficient, ordinary recycling reuses low-support victim slots for replacement children.
Ordinary recycling	$1.8\text{k} \leq t < 25\text{k}$	At structural events	Reclaim at most $\max(1, \lfloor 0.01B \rfloor)$ low-support non-parent primitives, additionally bounded by unmet splitting demand and the eligible victim-pool size. Apply transmittance-aware opacity redistribution when forming parent–child replacements.
Stale recovery cycle 1	3.0k–3.3k	Once	Store pre-probe opacities, clamp them with $\tau = 0.05$, optimize for $U = 300$ processed views, and reclaim the fraction $\rho = 0.2$ with the largest stale scores at the end of the recovery window.
Stale recovery cycle 2	6.0k–6.3k	Once	Repeat the same controlled recovery procedure with $\tau = 0.05$, $U = 300$, and $\rho = 0.2$.
Topology freeze	$t = 25$k	Once	Disable all topology-changing operations. The Gaussian population and primitive identities remain fixed thereafter.
Late MU-mean	25,001–27,500	5 views/update	Accumulate gradients over groups of five views and apply their mean in one optimizer update.
Late MU-mean	27,501–30,000	20 views/update	Accumulate gradients over groups of twenty views and apply their mean in one optimizer update.

A.5 Complete ReCap-GS Algorithm

Algorithm 1 summarizes the complete ReCap-GS training procedure used in the main benchmark. The procedure separates capacity demand from capacity recovery: early residual consensus and subsequent absolute-gradient refinement determine where capacity is allocated, whereas ordinary and stale victim selection determine which occupied slots are made available. Ordinary structural updates remain active during stale-recovery windows, and all topology-changing operations respect the hard population constraint $N_t \leq B$.

Algorithm 1: ReCap-GS training with a hard Gaussian budget

Require: posed training images $\{(I_v, \Pi_v)\}$
Require: SfM initialization $\mathcal{G}_0$ and Gaussian budget B

Require: recovery starts $\mathcal{T}_{\text{rec}} = \{3000, 6000\}$
Require: recovery duration $U = 300$, opacity threshold $\tau = 0.05$
Require: stale recovery fraction $\rho = 0.2$
$\mathcal{G} \leftarrow \textsc{CapSfM}(\mathcal{G}_0, B)$
$\mathcal{P} \leftarrow \varnothing$ ▷ residual-proposal buffer
initialize all recovery identities as invalid
Single-view optimization and structural adaptation
for $t = 1, \ldots, 25000$ **do**
  update the learning rate using processed-view index t
  sample training view v
  render $\hat{I}_v$, compute the photometric loss, and backpropagate
  if $t = 300$ **then**
    remove primitives with opacity < 0.02
  end if
  ▷ Early residual-consensus allocation
  if $t < 1800$ **then**
    if $t \bmod 5 = 0$ **then**
      extract positive-residual local maxima from view v
      lift valid proposals to 3D and append them to $\mathcal{P}$
    end if
    if $t \bmod 600 = 0$ **then**
      $\mathcal{Q} \leftarrow \textsc{ResidualConsensus}(\mathcal{P})$
      rank accepted voxel targets by mean residual
      $F \leftarrow B - |\mathcal{G}|$
      instantiate at most F highest-ranked targets
      $\mathcal{P} \leftarrow \varnothing$
      assert $|\mathcal{G}| \le B$
    end if
    ▷ Absolute-gradient refinement
  else if $t < 25000$ **and** $t \bmod 100 = 0$ **then**
    identify parent candidates using accumulated absolute screen-space gradients
    let D be the requested splitting demand
    $F \leftarrow B - |\mathcal{G}|$
    if $D > F$ **then**
      $D_t \leftarrow D - F$
      form the eligible non-parent victim pool $\mathcal{C}_t$
      compute support scores u_i using Equation 8
      $K_t \leftarrow \min(D_t, \max(1, \lfloor 0.01B \rfloor), |\mathcal{C}_t|)$
      select the K_t primitives with lowest support
      reuse each selected victim slot for a replacement child
      invalidate the recovery identity of each reused slot
      redistribute each recycled parent's opacity between the retained parent and its replacement child using Equation 13
    end if
    satisfy the remaining feasible splitting demand using free slots with the standard densification initialization
    assign invalid recovery identities to newly instantiated primitives
    assert $|\mathcal{G}| \le B$
  end if
  ▷ Start a stale-recovery window after the structural event
  if $t \in \mathcal{T}_{\text{rec}}$ **then**
    assign persistent recovery identities to all current primitives
    store their pre-probe opacities $\{\alpha_i^{\text{pre}}\}$
    $\alpha_i \leftarrow \min(\alpha_i^{\text{pre}}, \tau)$ for all current primitives
  end if
  ▷ End a stale-recovery window after the structural event
  if $t \in \{3300, 6300\}$ **then**
    let $t_c = t - U$

$\mathcal{E}_c \leftarrow$ primitives whose recovery identities remain valid since t_c
if $|\mathcal{E}_c| > 0$ **then**
read recovered opacities $\{\alpha_i^{\text{rec}}\}_{i \in \mathcal{E}_c}$
compute stale scores s_i using Equation 4
$K_c \leftarrow \max(1, \lfloor \rho |\mathcal{E}_c| \rfloor)$
reclaim the K_c primitives with largest stale scores
released slots remain free until a subsequent structural event
end if
invalidate all recovery identities
assert $|\mathcal{G}| \leq B$
end if
apply the ordinary single-view optimizer step
assert $|\mathcal{G}| \leq B$
end for
freeze topology after processing view iteration $t = 25000$
Late multi-view refinement: 5-view groups
reset the gradient accumulator
for $t = 25001, \ldots, 27500$ **do**
update the learning rate using processed-view index t
sample one training view, render, compute the loss, and accumulate its gradient
if $t \bmod 5 = 0$ **then**
divide the accumulated gradients by 5
apply one optimizer update
reset the gradient accumulator
end if
end for
Late multi-view refinement: 20-view groups
reset the gradient accumulator
for $t = 27501, \ldots, 30000$ **do**
update the learning rate using processed-view index t
sample one training view, render, compute the loss, and accumulate its gradient
if $t \bmod 20 = 0$ **then**
divide the accumulated gradients by 20
apply one optimizer update
reset the gradient accumulator
end if
end for
return $\mathcal{G}$

B Evaluation Protocol

B.1 Common Image-Quality Evaluation

We use the standard Mip-NeRF360 train/test split, with outdoor scenes evaluated at $4\times$ and indoor scenes at $2\times$ downsampling. All PSNR, SSIM, and LPIPS-VGG values are recomputed from rendered test-view PNG images using one common evaluator rather than mixing method-native metrics.

For each training seed, a metric is first averaged over held-out views within each scene and then averaged equally over the seven scenes. Reported mean $\pm$ standard deviation values are computed across training seeds 1 and 2. Consequently, the reported standard deviation reflects training-seed variation rather than variation across scenes or test images.

B.2 Strict-Budget Certification

Under the strict protocol, the active Gaussian population must satisfy

$$N_{\text{peak}} = \max_t N_t \leq B, \tag{15}$$

including initialization. For instrumented runs, method-side counters are evaluated immediately after initialization and after every operation capable of increasing the population. Because the population is constant between structural mutations, this gives the exact trajectory peak. We distinguish this *certified peak* from an *observed peak* obtained from partial logs, which is only a lower bound, and from the final Gaussian count, which does not constrain transient overshoot.

When the native SfM initialization contains more than B points, the strict adaptation selects exactly B indices uniformly without replacement using a fixed NumPy generator with seed zero. Selected indices are sorted before the corresponding positions, colors, and, when available, normals are extracted. The resulting initialization is therefore deterministic and identical across training seeds. When $N_0 \leq B$, the native initialization is left unchanged.

Initialization capping alone does not establish strict eligibility: the subsequent trajectory must also satisfy the budget. Methods without a validated trajectory-bounded adaptation are therefore reported only under the native/final-budget protocol.

B.3 Baseline Adaptations

Table 5 summarizes the configurations used for the strict comparison.

Table 5: Baseline adaptations used in the strict-budget benchmark.

Method	Initialization	Population control
Vanilla capped	Deterministic cap if $N_0 > B$	Clone/split demand truncated to available slots; defensive low-opacity pruning prevents residual overshoot.
Taming-3DGS	Deterministic cap if $N_0 > B$	Native target-population growth.
Taming+SPARE (our adaptation)	Deterministic cap if $N_0 > B$	Our adaptation of SPARE-GS regional structural regulation within the Taming-3DGS backbone.
SalientGS	Deterministic cap if $N_0 > B$	Native allocation procedure after initialization.
3DGS-MCMC	Deterministic cap if $N_0 > B$	Growth targets $\min(B, \lfloor 1.05 N_t \rfloor)$; relocation is population-preserving once the cap is reached.
Improved-GS	Deterministic cap if $N_0 > B$	At most $B - N_t$ parents are selected for long-axis splitting, each giving a net population increase of one.
Mini-Splatting	Native	Native/final-budget only; no validated strict adaptation.
Prune Wisely	Native	Native/final-budget only; no validated strict adaptation.

An official end-to-end SPARE-GS implementation was not available for the Taming-3DGS backbone used in our controlled budget sweeps. We therefore report Taming+SPARE (our adaptation) as our adaptation of the SPARE-GS regional structural-regulation principle rather than as an official reproduction.

For the resource analysis, we additionally include a 3DGS+SPARE operating point obtained with the official SPARE-GS implementation. This configuration, together with GS^2, is evaluated only at its natural representation size and is therefore shown as an isolated operating point rather than as a controlled budget sweep.

C Additional Benchmark Results

C.1 Full Budget Sweep

Table 6 reports the complete quality sweep across the evaluated Gaussian budgets. Results are averaged over the seven Mip-NeRF360 scenes and reported as mean $\pm$ standard deviation across

training seeds 1 and 2. All image-quality metrics are recomputed using the common matched-PNG evaluation pipeline described in Sec. B.1.

Table 6: Full budget-matched benchmark under the evaluated configurations. Values are mean ± sample standard deviation across training seeds 1 and 2 after averaging each seed over all seven scenes. 3DGS-MCMC and Improved-GS use our deterministic initialization cap whenever the native COLMAP cloud exceeds the requested budget; ReCap-GS is strict by design; all other entries follow their native/final-budget protocol.

Budget	Method	PSNR ↑	SSIM ↑	LPIPS ↓
50k	Vanilla capped	22.089±0.065	0.7076±0.0008	0.3950±0.0006
	Taming-3DGS	23.816±0.045	0.7189±0.0007	0.3837±0.0007
	Taming+SPARE (our adaptation)	23.858±0.083	0.7197±0.0014	0.3829±0.0015
	SalientGS	27.215±0.042	0.7995±0.0004	0.2844±0.0003
	3DGS-MCMC	26.085±0.022	0.7694±0.0001	0.3385±0.0000
	Improved-GS	26.200±0.005	0.7566±0.0001	0.3496±0.0001
	Mini-Splatting	25.382±0.001	0.7204±0.0001	0.3869±0.0001
	Prune Wisely	26.942±0.004	0.7493±0.0004	0.3554±0.0004
	ReCap-GS	26.625±0.006	0.7664±0.0004	0.3392±0.0004
100k	Vanilla capped	23.363±0.013	0.7547±0.0008	0.3425±0.0011
	Taming-3DGS	25.091±0.011	0.7675±0.0004	0.3308±0.0009
	Taming+SPARE (our adaptation)	25.068±0.001	0.7672±0.0007	0.3308±0.0014
	SalientGS	27.382±0.013	0.8134±0.0001	0.2690±0.0003
	3DGS-MCMC	27.229±0.008	0.8117±0.0001	0.2845±0.0002
	Improved-GS	27.485±0.001	0.8046±0.0000	0.2899±0.0000
	Mini-Splatting	26.346±0.006	0.7665±0.0001	0.3354±0.0003
	Prune Wisely	27.916±0.037	0.7957±0.0000	0.2999±0.0003
	ReCap-GS	27.846±0.004	0.8101±0.0000	0.2836±0.0002
200k	Vanilla capped	25.080±0.005	0.7971±0.0003	0.2909±0.0003
	Taming-3DGS	27.429±0.042	0.8186±0.0000	0.2693±0.0002
	Taming+SPARE (our adaptation)	27.437±0.053	0.8183±0.0003	0.2696±0.0001
	SalientGS	27.761±0.010	0.8315±0.0000	0.2431±0.0002
	3DGS-MCMC	28.212±0.030	0.8445±0.0001	0.2380±0.0001
	Improved-GS	28.567±0.003	0.8417±0.0002	0.2374±0.0004
	Mini-Splatting	27.271±0.008	0.8085±0.0001	0.2840±0.0002
	Prune Wisely	28.671±0.053	0.8355±0.0003	0.2481±0.0001
	ReCap-GS	28.806±0.005	0.8436±0.0000	0.2374±0.0000
300k	Vanilla capped	26.319±0.014	0.8179±0.0005	0.2642±0.0005
	Taming-3DGS	28.703±0.007	0.8399±0.0002	0.2386±0.0004
	Taming+SPARE (our adaptation)	28.666±0.004	0.8390±0.0003	0.2394±0.0005
	SalientGS	28.117±0.013	0.8434±0.0002	0.2215±0.0001
	3DGS-MCMC	28.630±0.013	0.8587±0.0001	0.2153±0.0000
	Improved-GS	29.065±0.045	0.8592±0.0002	0.2100±0.0001
	Mini-Splatting	27.773±0.015	0.8298±0.0001	0.2551±0.0003
	Prune Wisely	28.987±0.013	0.8534±0.0001	0.2223±0.0002
	ReCap-GS	29.200±0.007	0.8588±0.0001	0.2143±0.0003
500k	Vanilla capped	27.939±0.001	0.8383±0.0001	0.2346±0.0001
	Taming-3DGS	29.162±0.027	0.8557±0.0004	0.2142±0.0004
	Taming+SPARE (our adaptation)	29.145±0.006	0.8551±0.0001	0.2148±0.0003
	SalientGS	28.525±0.005	0.8553±0.0002	0.1978±0.0002
	3DGS-MCMC	29.102±0.022	0.8726±0.0001	0.1922±0.0002
	Improved-GS	29.624±0.003	0.8768±0.0000	0.1809±0.0001
	Mini-Splatting	28.317±0.010	0.8515±0.0002	0.2229±0.0002
	Prune Wisely	29.215±0.025	0.8694±0.0000	0.1963±0.0002

Continued on next page

Table 6 – continued

Budget	Method	PSNR ↑	SSIM ↑	LPIPS ↓
	ReCap-GS	29.573±0.017	0.8738±0.0001	0.1890±0.0002
1M	Vanilla capped	28.808±0.047	0.8562±0.0001	0.2054±0.0002
	Taming-3DGS	29.506±0.008	0.8709±0.0001	0.1881±0.0005
	Taming+SPARE (our adaptation)	29.513±0.017	0.8703±0.0002	0.1891±0.0004
	SalientGS	28.931±0.014	0.8659±0.0002	0.1742±0.0003
	3DGS-MCMC	29.542±0.008	0.8854±0.0001	0.1682±0.0001
	Improved-GS	29.955±0.006	0.8903±0.0002	0.1560±0.0000
	Mini-Splatting	28.766±0.027	0.8716±0.0001	0.1889±0.0000
	Prune Wisely	29.329±0.020	0.8797±0.0001	0.1743±0.0000
	ReCap-GS	29.917±0.010	0.8880±0.0000	0.1589±0.0000
2M	Vanilla capped	29.145±0.010	0.8696±0.0000	0.1842±0.0002
	Taming-3DGS	29.661±0.027	0.8797±0.0002	0.1697±0.0006
	Taming+SPARE (our adaptation)	29.693±0.008	0.8796±0.0005	0.1700±0.0006
	SalientGS	29.158±0.012	0.8720±0.0001	0.1584±0.0003
	3DGS-MCMC	29.827±0.022	0.8932±0.0000	0.1520±0.0000
	Improved-GS	29.972±0.003	0.8961±0.0000	0.1446±0.0000
	Mini-Splatting	29.048±0.008	0.8830±0.0001	0.1667±0.0001
	Prune Wisely	29.287±0.029	0.8804±0.0010	0.1664±0.0002
	ReCap-GS	30.008±0.011	0.8932±0.0001	0.1431±0.0001

C.2 Effect of Initialization Capping

To quantify the effect of the deterministic initialization adaptation, we compare each initialization-capped run with its corresponding native run. Table 7 reports the PSNR difference, $\Delta\mathrm{PSNR} = \mathrm{PSNR}_{\mathrm{strict}} - \mathrm{PSNR}_{\mathrm{native}}$.

Table 7: Effect of deterministic initialization capping. Positive values indicate that the strict initialization improves PSNR relative to the corresponding native run.

Method	50k	100k	200k
Vanilla capped	-0.282	+0.003	+0.025
Taming-3DGS	+1.757	+1.627	+0.464
Taming+SPARE (our adaptation)	+1.713	+1.605	+0.432
SalientGS	-1.405	-0.557	-0.041

Table 7 includes only methods for which paired native and initialization-capped runs are available at the same requested budgets. Although 3DGS-MCMC and Improved-GS also use the deterministic initialization cap in the strict benchmark, corresponding native budget-matched pairs are not available and are therefore omitted from this comparison.

D Additional Resource Analysis

D.1 Measured Resource Consumption

Peak VRAM is measured as the maximum `nvidia-smi` memory usage over the complete method process tree, and wall time covers the same end-to-end pipeline as in Sec. 4.1. For ReCap-GS, training images remain in CPU memory and tensors required by the active camera are staged to the GPU on a dedicated CUDA stream.

Table 8 reports the exact measurements underlying the quality–resource analysis in Fig. 3. The exact operating points for all methods shown in Fig. 3 are provided in the accompanying `quality_resource_tradeoff_data.csv` file.

Table 8: Measured resource consumption of ReCap-GS and Improved-GS. Peak VRAM is reported in MiB and wall time in minutes. The final column gives the relative reduction in peak VRAM of ReCap-GS with respect to Improved-GS.

Budget	Method	Peak VRAM (MiB)	Wall time (min)	VRAM reduction
50k	ReCap-GS	2115	5.96	76.0%
	Improved-GS	8828	6.45	
100k	ReCap-GS	2464	6.64	73.1%
	Improved-GS	9210	6.80	
200k	ReCap-GS	3403	7.72	65.3%
	Improved-GS	9815	7.61	
300k	ReCap-GS	4000	8.55	61.4%
	Improved-GS	10369	8.31	
500k	ReCap-GS	4975	10.13	55.5%
	Improved-GS	11172	9.56	
1M	ReCap-GS	7379	13.75	41.3%
	Improved-GS	12571	11.53	
2M	ReCap-GS	12190	20.27	11.6%
	Improved-GS	13794	13.31	

E Additional Ablation Details

Recovery-count matching. Recovery policies use the same nominal probe schedule, recovery delay, recovery fraction, and eligibility definition. Because independent runs diverge slightly after structural interventions, realized victim counts are not exactly identical: their mean absolute relative difference is 0.136% and the maximum is 1.137%. We therefore describe these controls as schedule-matched with nearly identical realized counts rather than exactly count-matched.

Opacity redistribution. Across the seven scenes and seeds 1–2, transmittance-aware opacity redistribution improves PSNR in 11 of 14 matched scene–seed pairs, SSIM in 14 of 14, and LPIPS in 12 of 14. Its mean effect is $+0.0555$ dB PSNR, $+0.001221$ SSIM, and -0.002025 LPIPS.

MU-mean versus MU-sum. On the three diagnostic scenes and seeds 1–2, replacing MU-mean with MU-sum decreases PSNR by 0.0366 dB and SSIM by 0.001190, while increasing LPIPS by 0.000788. We therefore retain gradient averaging in the final configuration. Multi-view Update originates from Improved-GS and is treated as an adaptation rather than an original ReCap-GS contribution.

F Additional Qualitative Results

The main paper provides a multi-method qualitative comparison at a strict 100k-Gaussian budget. Here, we compare ReCap-GS and Improved-GS systematically over the low-capacity regime. For each of the seven Mip-NeRF360 scenes, the same held-out test view and two fixed crop locations are shown at 50k, 100k, and 200k Gaussians.

The RGB reconstructions are often visually close, with differences becoming subtler as capacity increases. We include all seven scenes to provide a systematic view of this regime rather than selecting only favorable examples.

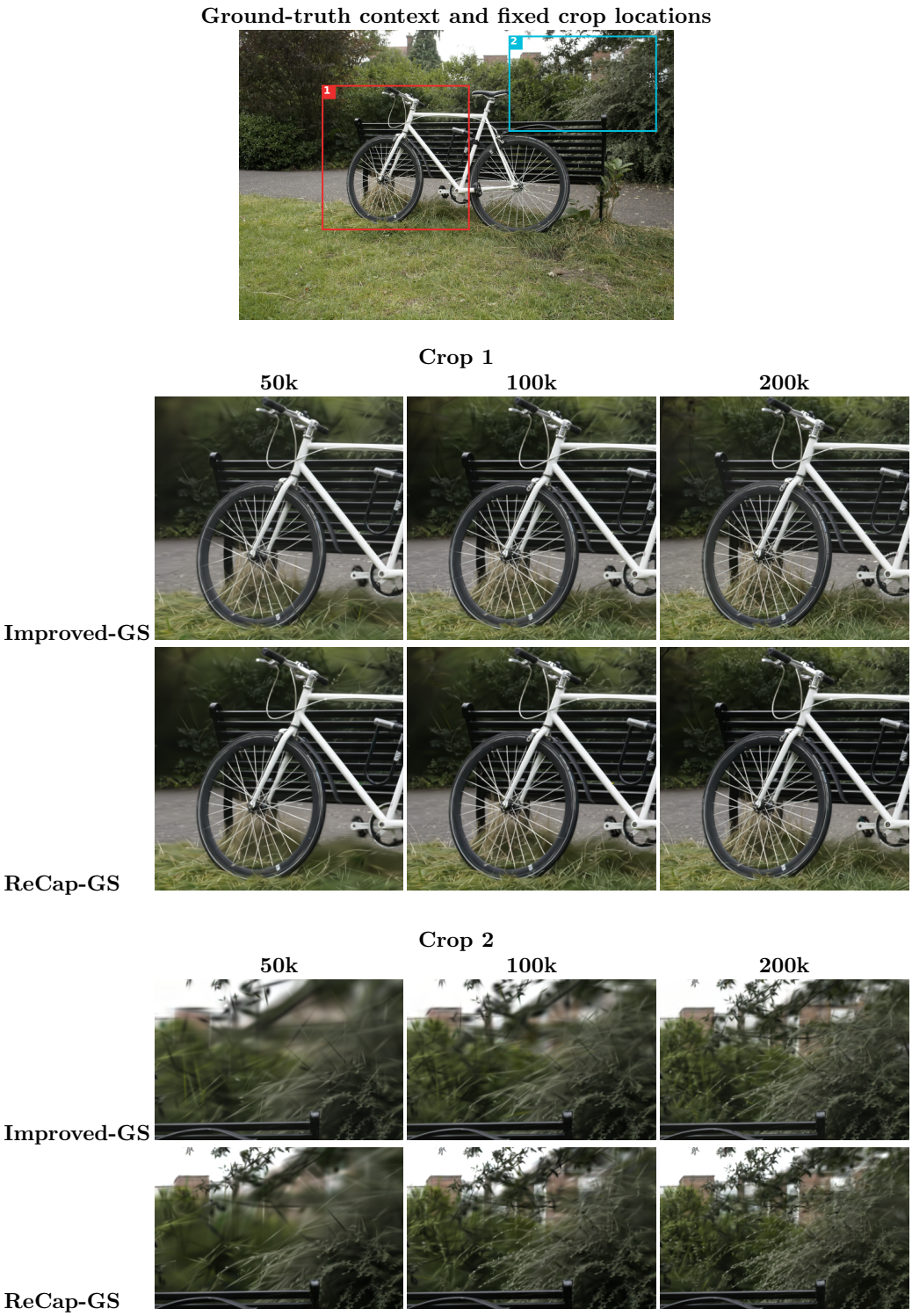

Figure 4: Low-capacity qualitative comparison on the *bicycle* scene. Improved-GS and ReCap-GS are shown at 50k, 100k, and 200k Gaussians using the same held-out view and identical crop coordinates.

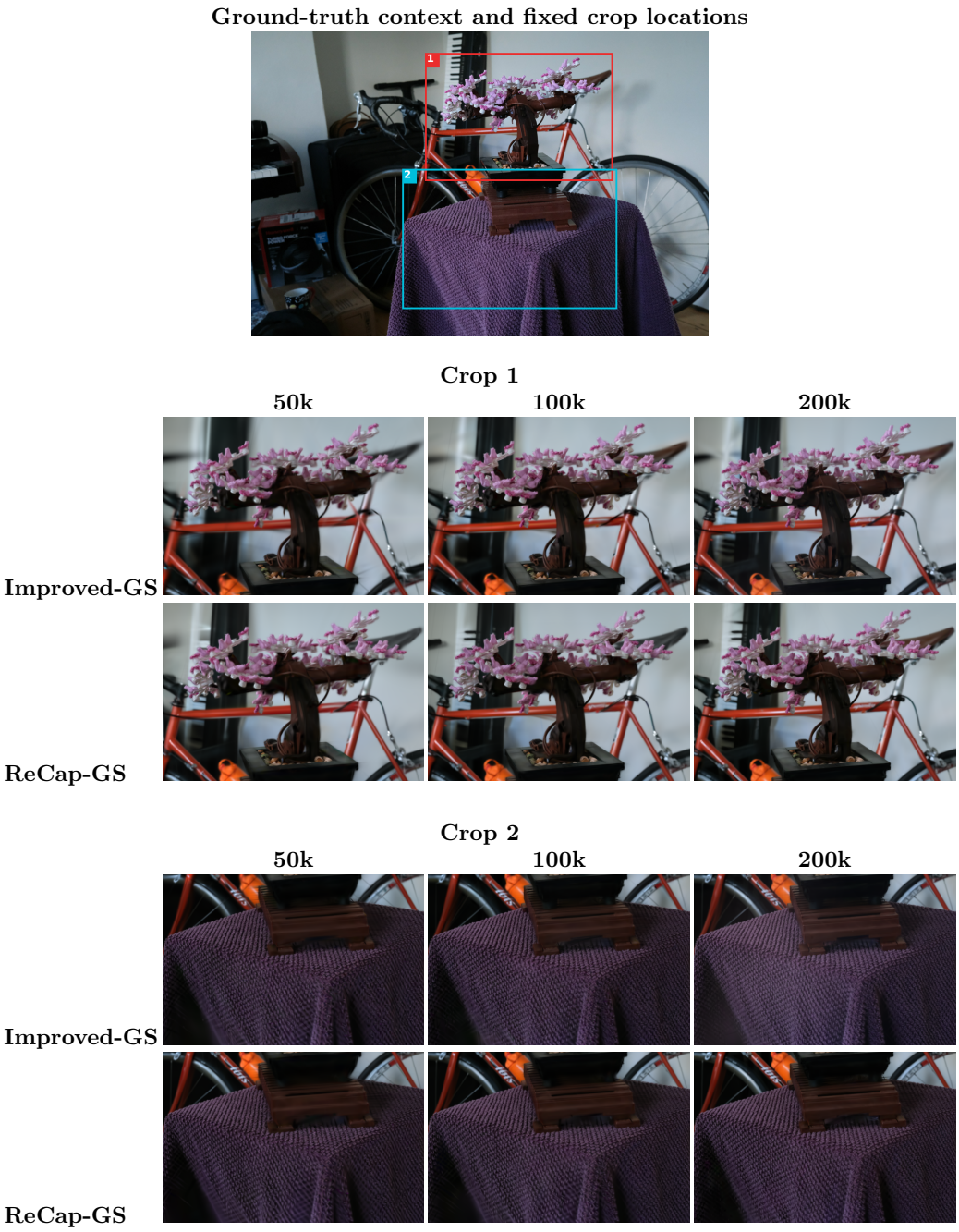

Figure 5: Low-capacity qualitative comparison on the *bonsai* scene. Improved-GS and ReCap-GS are shown at 50k, 100k, and 200k Gaussians using the same held-out view and identical crop coordinates.

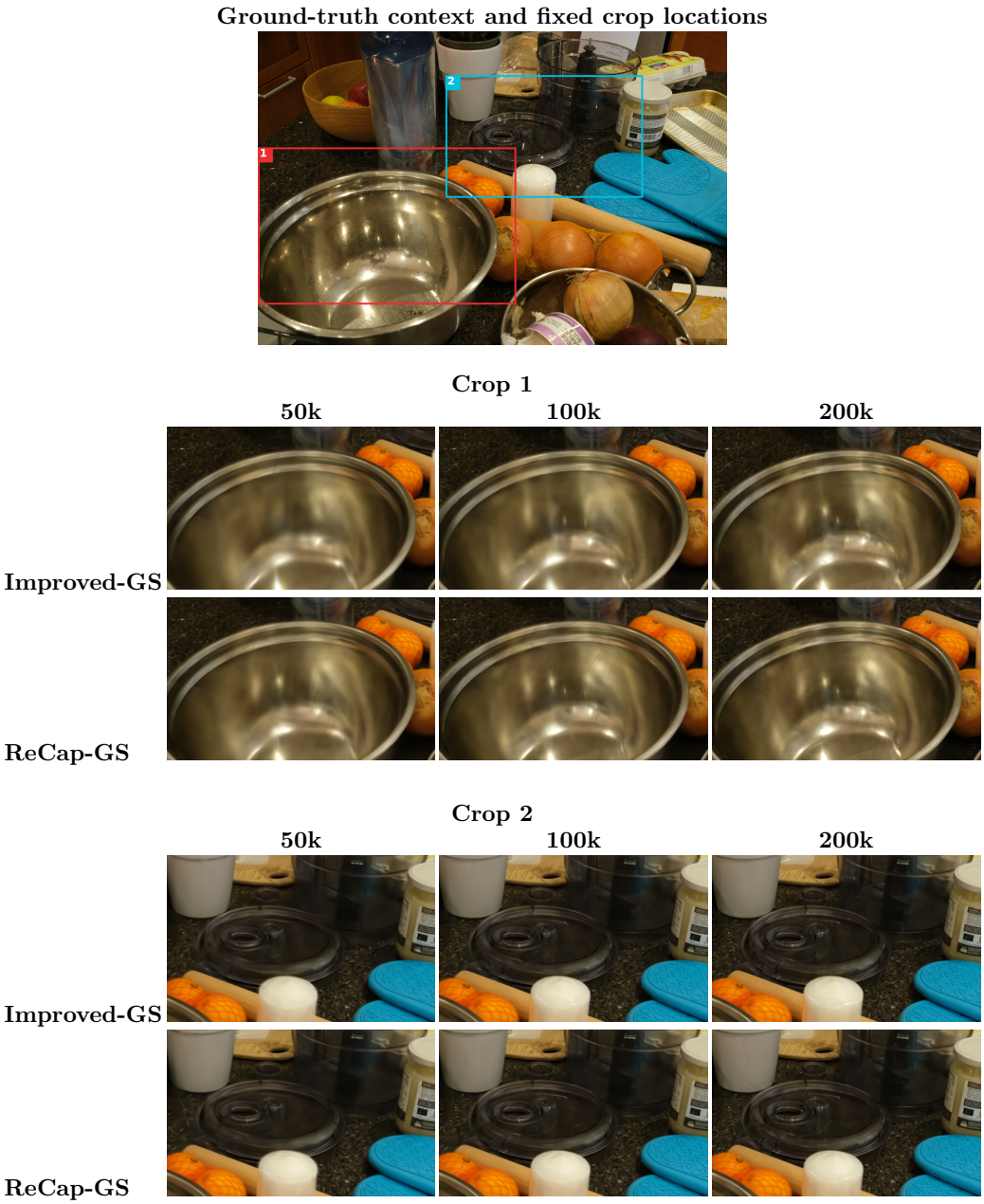

Figure 6: Low-capacity qualitative comparison on the *counter* scene. Improved-GS and ReCap-GS are shown at 50k, 100k, and 200k Gaussians using the same held-out view and identical crop coordinates.

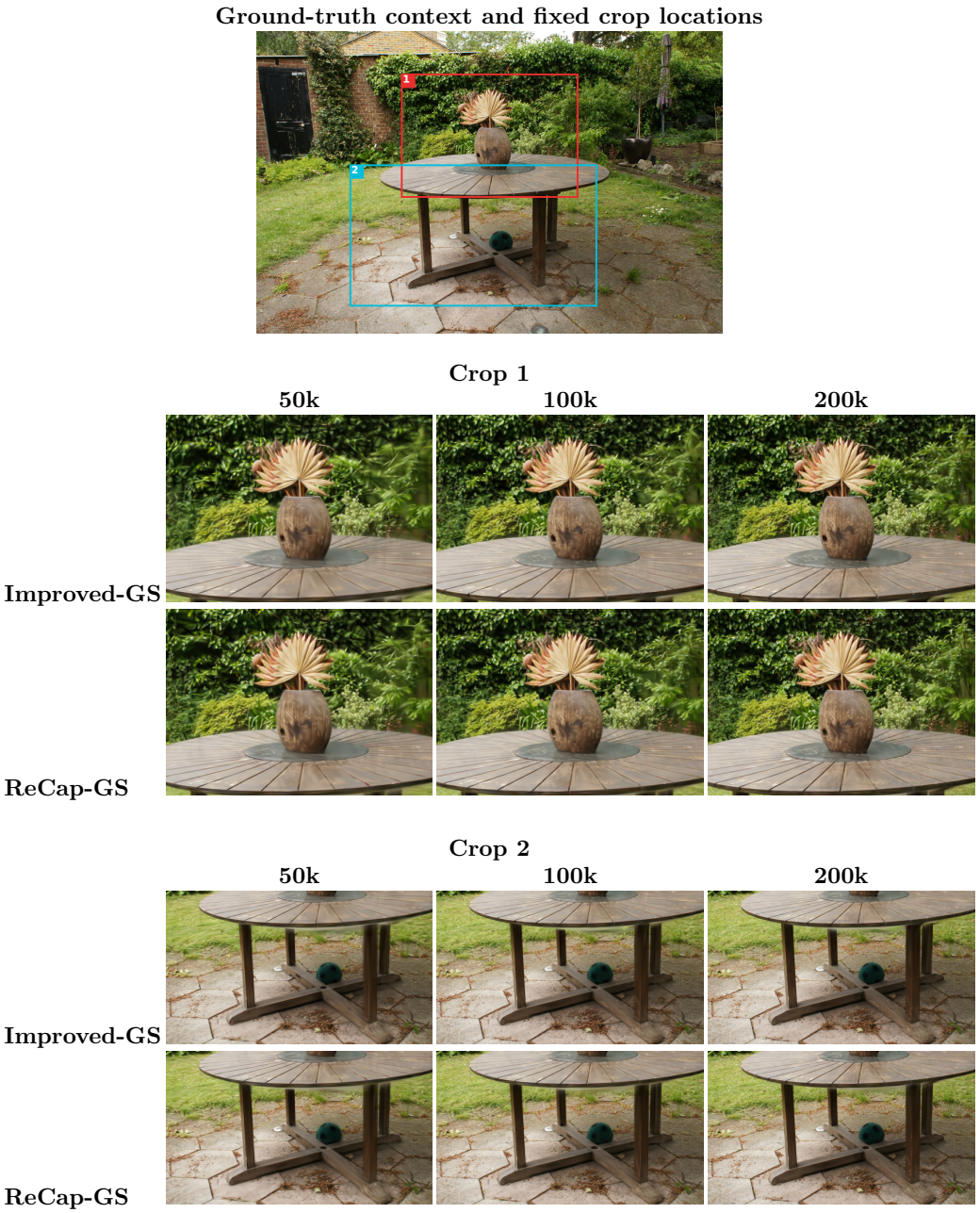

Figure 7: Low-capacity qualitative comparison on the *garden* scene. Improved-GS and ReCap-GS are shown at 50k, 100k, and 200k Gaussians using the same held-out view and identical crop coordinates.

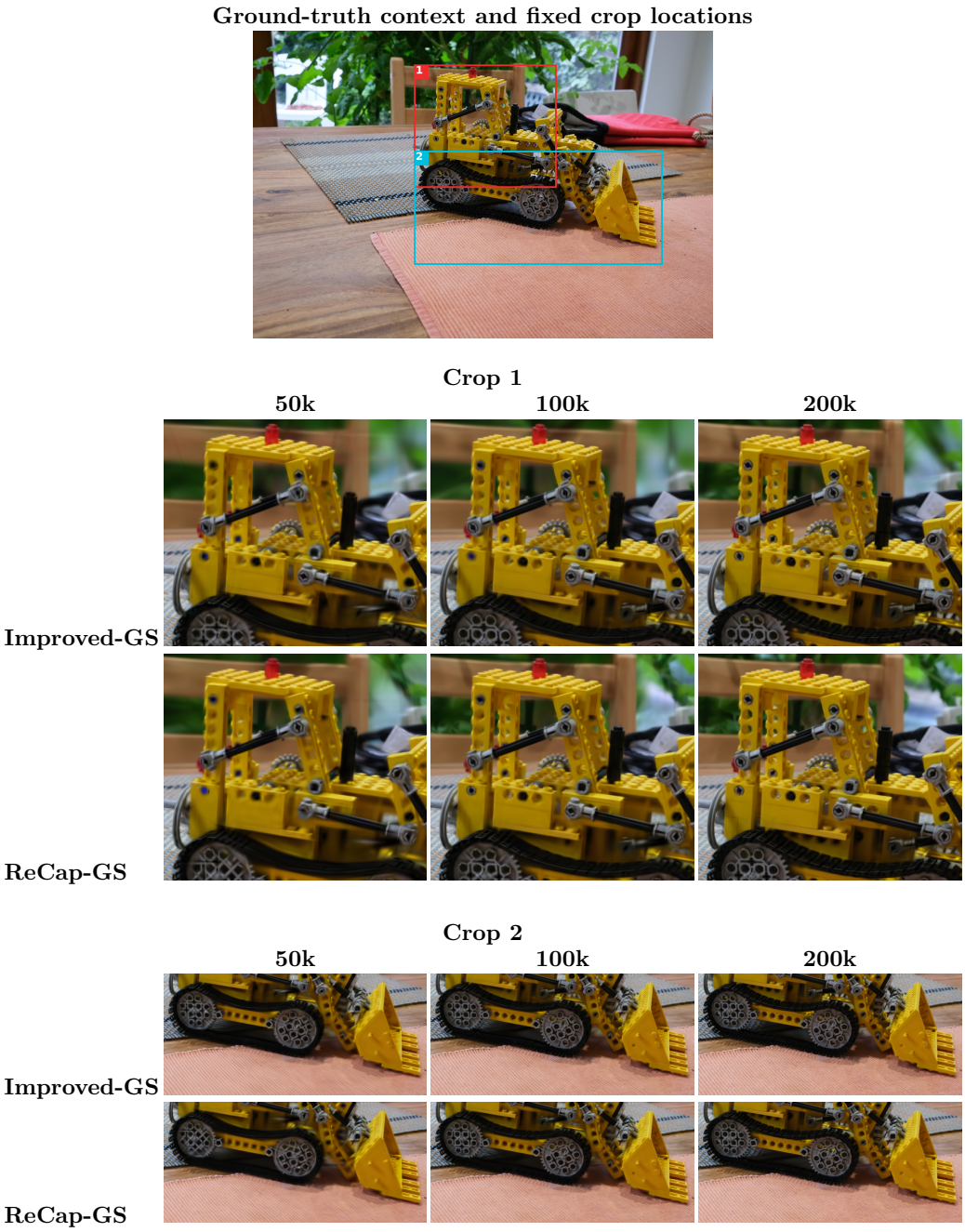

Figure 8: Low-capacity qualitative comparison on the *kitchen* scene. Improved-GS and ReCap-GS are shown at 50k, 100k, and 200k Gaussians using the same held-out view and identical crop coordinates.

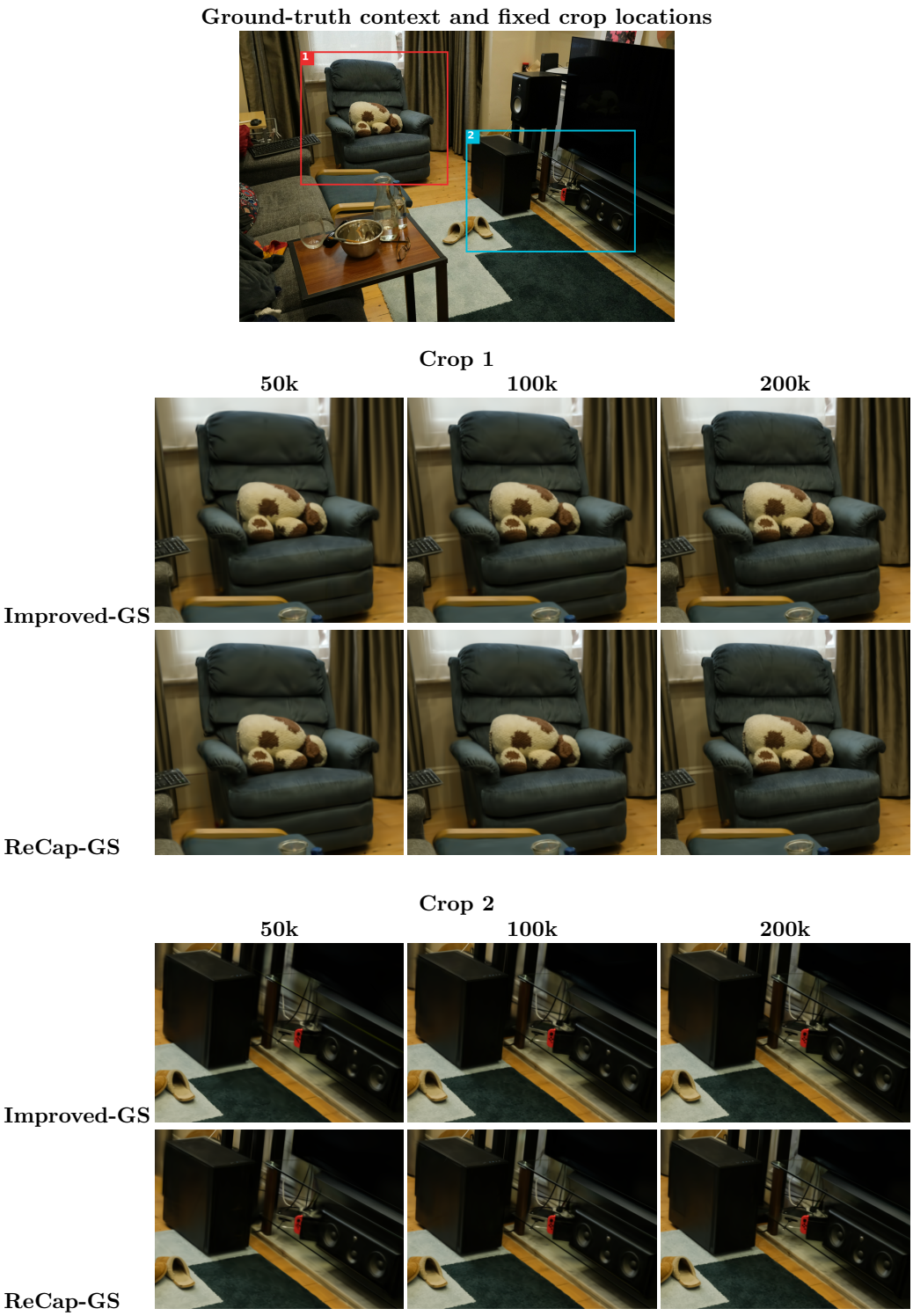

Figure 9: Low-capacity qualitative comparison on the *room* scene. Improved-GS and ReCap-GS are shown at 50k, 100k, and 200k Gaussians using the same held-out view and identical crop coordinates.

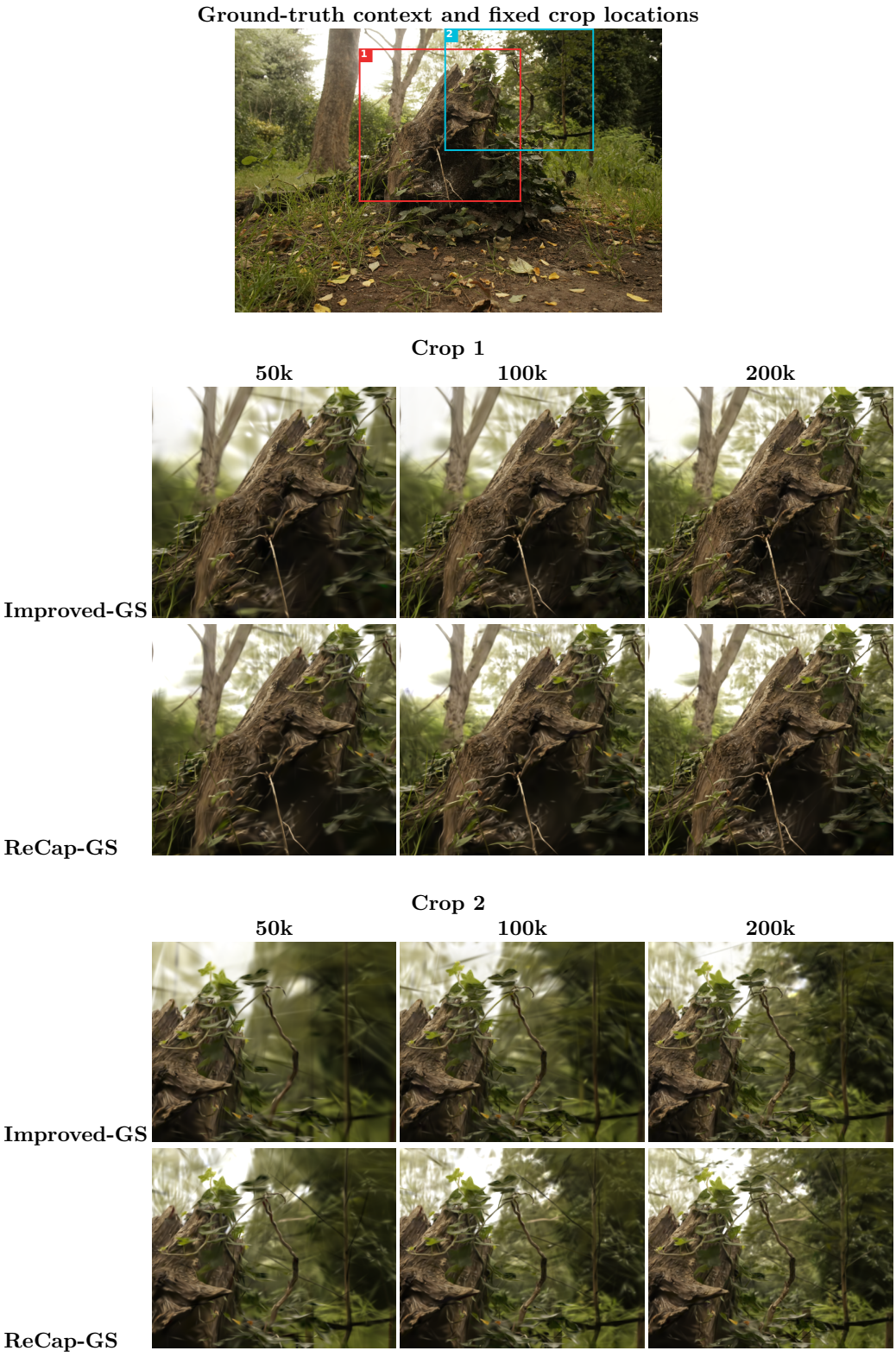

Figure 10: Low-capacity qualitative comparison on the *stump* scene. Improved-GS and ReCap-GS are shown at 50k, 100k, and 200k Gaussians using the same held-out view and identical crop coordinates.